

データサイエンスをはじめましょう
- Data Science for All -

鈴木寛 (Hiroshi Suzuki)

2023-09-22

目次

データサイエンスをはじめましょう	11
Data Science for All	11
著者について	11
コンピュータ言語について	12
言語について	13
PDF、ePub 版について	13
参考	14
第 1 章 はじめに	15
1.1 データサイエンスとは	15
1.2 Gapminder	23
1.3 「データサイエンスを始めましょう」の特徴	24
1.4 学習方法について	25
1.5 参考	26
第 2 章 学ぶ内容	27
2.1 データサイエンス入門	27
2.2 第一部 オープンデータ	27
2.3 第二部 R の基礎	27
2.4 第三部 オープンデータの活用	28
2.5 第四部 探索的データ分析 Exploratory Data Analysis	29
2.6 第五部 分析例	29
2.7 付録	29
2.8 パッケージについて	29
第 3 章 はじめてのデータサイエンス	31
3.1 データサイエンスの実際	31
3.2 R のパッケージを活用	32
3.3 練習	57
3.4 プロジェクト	57
3.5 まとめ	58
第 4 章 世界開発指標とオープンデータ	59
4.1 世界開発指標 (WDI)	59

4.2	オープンデータ (Open Data)	64
第5章	データサイエンスノートブック	67
5.1	再現性のために記録すべきこと	67
5.2	技術的なこと	68
5.3	まとめ	70
第1部	第一部オープンデータ	71
第6章	オープンデータ	73
6.1	概要	73
6.2	さまざまな機関のオープンデータ	74
6.3	オープンデータについて	75
6.4	ダッシュボード	76
6.5	API を利用したパッケージ	78
第7章	世界銀行 (World Bank)	79
7.1	概要	79
7.2	データベース	79
7.3	ダッシュボード (Dashboard)	83
7.4	API	88
7.5	Google Public Data Explorer	89
第8章	OECD (経済協力開発機構)	93
8.1	概要	93
8.2	データベース	93
8.3	OECD	93
8.4	ダッシュボード	94
8.5	API	94
第9章	United Nations (国際連合)	95
9.1	概要	95
9.2	データベース	95
9.3	ダッシュボード	95
9.4	API	95
第10章	Our World in Data	97
10.1	概要	97
10.2	レポート	98
10.3	データベース	99
10.4	ダッシュボード	99
10.5	API	100
第11章	e-Stat 日本の政府統計	101

11.1	概要	101
11.2	データベース	101
11.3	ダッシュボード	102
11.4	API	103
11.5	estatapi 利用概要	103
第 12 章	SDGs	105
12.1	SDGs とは	105
12.2	国際連合	105
12.3	世界銀行	105
12.4	OWID	106
12.5	SDG Index	107
12.6	外務省	107
第 13 章	その他のサイト	109
13.1	WIR	109
13.2	EuroStat	109
第 II 部	第二部 R の基礎	111
第 14 章	R Studio で R	113
14.1	はじめに	113
14.2	R と R Studio	113
14.3	R と R Studio のインストール	114
14.4	クラウド - Posit Cloud	120
14.5	R のその他の利用方法	122
第 15 章	R Markdown	125
15.1	再現可能性とプログラムの文書化	125
15.2	準備：パッケージのインストール	126
15.3	R Notebook	126
15.4	日本語のテンプレート	127
15.5	R Markdown いくつかの Output	127
15.6	YouTube Video - rmarkdown	128
第 16 章	base R	131
16.1	はじめに	131
16.2	プロジェクト - Project	131
16.3	コンソールで実行 - Run in Console	132
16.4	RStudio について	136
16.5	R Script 実行記録	136
16.6	パッケージ - Packages	145
16.7	練習問題 Posit Primers	146

16.8	参考文献 References	147
16.9	YouTube Video - getstarted	147
16.10	まとめと復習	147
第 17 章	Tidyverse	149
17.1	はじめに	149
17.2	あやめ (iris) のデータ	150
17.3	WDI のデータ	160
17.4	まとめ	165
第 18 章	変形 (Transform)	167
18.1	dplyr 概要	168
18.2	例から学ぶ dplyr, I	171
18.3	dplyr の参考文献	177
18.4	例から学ぶ dplyr, II	177
18.5	練習問題	202
18.6	まとめ	204
第 19 章	視覚化 (Visualize)	207
19.1	基本的なこと	207
19.2	散布図 (Scatter Plot)	210
19.3	箱ひげ図 Boxplot	214
19.4	例から学ぶ ggplot2, I	222
19.5	例から学ぶ ggplot2, II	225
19.6	コメント	243
19.7	練習	243
第 20 章	データ読み込み (Import)	245
20.1	はじめに	245
20.2	概略	245
20.3	データの整形 tidyr	248
20.4	参照: tidyr	256
20.5	二つの表の結合	257
第 21 章	数理モデル (Modeling)	263
21.1	はじめに	263
21.2	準備 (Setup)	264
21.3	線形モデル入門	265
21.4	世界開発指標 (WDI) からの例	282
21.5	数理モデルに関する参考文献	289
第 22 章	情報共有 (Communicate)	291
22.1	はじめに	291
22.2	R Markdown について	291

22.3	レポートとスライド	291
22.4	Quarto について	294
22.5	参考文献	295
第 23 章	復習	297
23.1	はじめてのデータサイエンスについて	297
23.2	WDI の実際のデータを使って	324
23.3	Posit Primers - Remaining Tutorials	324
第 III 部	第三部オープンデータの活用	325
第 24 章	オープンデータの分析	327
第 25 章	World Bank	329
25.1	World Development Indicator (WDI)	329
25.2	WDI パッケージ	334
25.3	可視化 Visualization	344
25.4	課題 Assignment	348
25.5	政府支出 (国内総生産比) Government Expenditure, (% of GDP)	349
第 26 章	UN Data (国際連合データ)	361
第 27 章	Our World in Data	363
第 28 章	世界不平等データベース	365
28.1	WIR2022	365
28.2	F2: The poorest half lags behind: Bottom 50%, middle 40% and top 10% income shares across the world in 2021	369
28.3	Pivot data from long to wide:	371
28.4	Practice: F4 and F13	371
28.5	F3: Top 10/Bottom 50 income gaps across the world, 2021	372
28.6	F3: Top 10/Bottom 50 income gaps across the world, 2021 - Original	372
28.7	Remaining Charts	379
28.8	F5: Global income inequality: T10/B50 ratio, 1820-2020	380
28.9	F9: Average annual wealth growth rate, 1995-2021 - fit curve + alpha	381
28.10	F7: Global income inequality, 1820-2020 - pivot + fit curve	382
28.11	F10: The share of wealth owned by the global 0.1% and billionaires, 2021 - pivot + fit curve	383
28.12	F6: Global income inequality: Between vs. Within country inequality (Theil index), 1820-2020 - pivot + area	385
28.13	F11: Top 1% vs bottom 50% wealth shares in Western Europe and the US, 1910-2020 - pivot name_sep + fit curve	386

28.14	F8: The rise of private versus the decline of public wealth in rich countries, 1970-2020 - rename + pivot + pivot + fit curve	389
28.15	F15: Per capita emissions across the world, 2019 - add row names + dodge	393
第 29 章	e-Stat	395
29.1	API の使い方	395
29.2	estatapi 利用概要	396
第 30 章	その他のデータベース	407
30.1	国際機関	407
30.2	世界の国の政府機関	407
30.3	ClinicalTrials.gov	408
30.4	Harvard Dataverse	408
30.5	Kagle	408
30.6	Gapminder	408
第 IV 部	PART IV EDA	413
第 31 章	探索的データ解析	415
31.1	探索的データ解析 (EDA) とは	415
31.2	探索的データ解析 (EDA) の一例	415
第 32 章	地図 Choropleth Maps	425
32.1	準備	425
32.2	Natural Earth Data	427
32.3	geodata Package	441
32.4	参考文献	448
第 V 部	PART V EXAMPLES	451
第 33 章	Example 1	453
第 34 章	参考文献	455
34.1	教科書	455
34.2	MOOCs などのオンラインコース	457
34.3	その他の参考書	458
34.4	統計関連	459
付録 A	日本語の扱いについて	461
A.1	日本語・中国語・韓国語	461
A.2	Base R でタイトルに日本語	461
A.3	列名や、データに日本語	462

A.4	kable で表示	462
A.5	ggplot でグラフを作成	463
A.6	備考:	463
A.7	参考: 日本語の表示について	464
付録 B	IT ツール	467
B.1	Git と GitHub	467
付録 C	Bookdown	479
C.1	About	479
C.2	Hello bookdown	480
C.3	Cross-references	480
C.4	Parts	481
C.5	Footnotes and citations	482
C.6	Blocks	483
C.7	Sharing your book	484
付録 D	To Do リスト	485

データサイエンスをはじめましょう

Data Science for All

データサイエンスは、広い意味をもったことばで、一口に、学び始めると言っても、さまざまな始め方があると思います。本書では、そのひとつを提案するとともに、共に学んでいきたいと願って、書き始めました。

いろいろな方々や、利用方法を想定して、導入のような内容や、本書の中心をなす、Rでのプログラミングを利用しなくても、ダッシュボードなどを利用することにより、データをさまざまな角度から見る経験をする、スクリプトや、テンプレートを使って、一部を置き換えることで、データサイエンスを経験すること、基本的なRでのプログラミングを学んで、自分で、簡単なプログラムを書いて、分析をすること、さらに、こんなときは、どうしたら良いかなど、少し詳しい説明などを含んだ部分などです。

また、順次、例を提供する、ブログのようなものも書いていきたいと思っています。この書を利用するために「データサイエンスを教えてくださいませんか」と、教える方のサポートも、書いていく予定です。

みなさんも一緒にデータサイエンスを学んでみませんか。

著者について

著者は、大学の学生の時以来、数学を学び、大学で教え、2019年春に退職。それ以来、少しずつ、データサイエンスを学んでいます。

幸運にも、2019年9月の日本数学会教育委員会主催教育シンポジウムで、「文理共通して行う数理・データサイエンス教育」という題で、話す機会が与えられ、その後、あることが契機となり、2020年度から、毎年、冬学期（12月から2月）に、大学院一般向け（分野の指定なし）の授業、「研究者のためのデータ分析（Data Analysis for Researchers）」を担当しています。複数の教員で担当しますが、基本的な部分は、わたしが教え、応用について、他の先生がおしえています。受講生は20人程度で、殆どが、外国人。それも、多国籍で、多くても一国から三人程度。英語で教えています。

この授業の関係で親しくなった経済学の先生に依頼されて、いくつかのレベルの経済学の授業で、2・3時限、世界銀行の世界開発指標などを使った分析の入門を、特別講師として教えさせていただきました。これらのコースは日本語で教えましたが、日本語で書かれ

た一般の学生向けのデータサイエンスの教科書、学習教材が不足していることも感じました。

2023年3月に京都大学数理解析研究所で開かれた数学教育の会で、話す機会が与えられ、大学などで全学生向けに提供する「数理・データサイエンス・AI」について講演し、生成系 AI の進化も踏まえて、どのようなことを学ぶ機会とすれば良いかについても、考えました。

「データサイエンスをはじめましょう」も、このような背景の下で書かれたものです。

コースで利用したものや、講演記録ににご興味のある方は、このページの下にもリンクがある、著者のホームページを参照してください。

コンピュータ言語について

データサイエンスには、コンピュータを使います。コンピュータに指示をして、データを使いやすく変形したり、計算をしてもらったり、グラフを書いてもらったりします。そのためにコンピュータとのやりとりをする言語が必要です。さまざまな言語（プログラミング言語と呼びます）が、使われますが、データサイエンスで、一番、使われているのは、Python と、R です。

「データサイエンスをはじめましょう」では、統計解析のために開発された R を使います。いずれは、python についても触れたいと思いますが、プログラミングの経験がない方も含めて、最初にデータサイエンスを学ぶには、R は最適だと考えています。また現在は、R Studio IDE (integrated development environment, 統合開発環境) で、R を簡単に使うことができます。さらに、簡単なプログラムであれば、Posit Cloud で試したり、共有することも可能です。また、再現性 (Reproducibility) や、なにを実行しているのかの説明を同時に記述すること (Literate Programming) は、非常に重要ですが、その記述も、R Markdown や、Quarto によって、可能になっています。これが、Excel や、Google Spread Sheet ではなく、R を推奨する理由でもあります。この文書も、R Markdown の一つの形式の、bookdown を利用しています。最後に、Bookdown に関連して、膨大な数の、参考書も、無償で提供されており、オンラインで読むことができることも、R をお薦めする理由です。

ただし、日本語の学習教材は、まだ十分とは言えない状況です。この文書を書き始めたのも、すこしでも、お役に立つことができればとの、気持ちが背景にあります。

もう一つ付け加えると、高校の教科書でも、一部、Python のスクリプトが使われ、大学で、全学生向けに、データサイエンスのコースを提供するときにも、Excel を使うか、Python を使うかが R と比較するとより一般的のように見えます。しかし、それは、教える側の都合が影響しているのではないのでしょうか。急いでコースを提供するため、身近な Excel から始めることにしたり、データサイエンスなら、情報科学の教員がコースをデザインせざるを得ないことから、汎用性が高く、かつデータサイエンスでも、中心的な役割を果たす Python を選択すると言ったことです。

しかし、すべての人が、データサイエンスを学ぶ必要があるならば、文系の先生、そして、純粋数学を生業（なりわい）としてきたわたしのようなものも含めて、教えることに関わることは、とても大切だと思います。教員も学ぶ必要があるからです。R は、社会科学系の分野で、最近特に使われています。分野ごとには、SPSS や、Stata や、MATLAB などが使われているかもしれませんが、これらは、有償で、大学では、使えたとしても、一般の学生が卒業後も、これらのソフトを使えることは稀です。また、研究においても、最近では、さまざまな分野で、R がより一般的になっているとも聞きます。

百聞は一見にしかずで、例を見ていただくのがよいと思いますが、R では、非常に短い命令で、対話的に、すぐ結果が得られます。テンプレートを使って、一部だけ書き換えて使うことも可能で、その人のレベルにあった利用が可能だと思います。さらに、Shiny のような、ダッシュボード形式のものも、利用可能ですし、learnr のように、対話型の、練習問題を提供することも可能です。

少しずつ紹介していきたいと思います。

言語について

ご覧の通り、本書は、日本語で書かれています。用語は、英語、あるいは、英語を追記、または、英語をカタカナにしただけのものを使用する可能性が大きいですが、説明は、極力、日本語で書いていく予定です。

しかし、基本的に、コード（プログラムの記述）には、日本語を使わないで書いていく予定です。とくに、初心者にとっては、二バイト文字で表現される日本語を含むコードの扱いは、負担になることが多いからです。最近では、コードの中で日本語を使用しても、ほとんど、問題は起きないようになっています。そうであっても、世界の人の共通言語として、プログラム言語を学んでいくときには、日本語を使わないことは意義があると思います。日本語を使わないことで、世界中の人たちから、アイディアを学び、あるときは協力し、例を提供して、作業をしていくことが可能だからです。

少し慣れてきて、日本語のデータなどを扱うときには、コードにも日本語を使う必要が生じてきますから、日本語の利用についても、追って説明していきます。APPENDIX A を参照してください。

最初は、みなさんも、変数（variable）や、オブジェクト（object）に名前をつけるときは、半角英数を使い、日本語は、使わないようにすることをお勧めします。自分には難しいと感じる時は、ローマ字を半角英数で使うのはいかがでしょうか。

PDF、ePub 版について

この文書は、PDF 版と、ePub 版も作成しています。しかし、扱いが異なるので、ある程度完成するまでは、あまり更新しない予定です。いずれ、これらも、更新したものを公開できると良いのですが。試験公開版は、下のリンクにあります。

- PDF 版
- ePub 版

参考

この電子書籍以外にも、データ・サイエンスについて幾つかの文書をインターネット上に公開しています。わたしのホームページにリストしてありますので、ご興味のあるかたは、参考にしてください。

- データサイエンスを学びませんか・データサイエンス教育

第1章

はじめに

1.1 データサイエンスとは

データサイエンスとはどのようなものでしょうか。いくつかの定義を紹介しますが、新しい分野で、非常に広い範囲の人たちが、データサイエンスに関係していることから、誰でもが受け入れられる定義することは、難しいように思います。

簡単に表現すると「データを活用するための科学」かなと、わたしは、考えています。皆さんが、これから、データサイエンスを学びながら、自分だったらどのように表現するか、考えてください。

「データを活用するための科学」には、三つのことばが含まれています。「データ」「活用するため」「科学」。一つ一つ、厳密な定義は難しいですが、データは、これから、皆さんがたくさん出会いますので、それまで置いておきましょう。「活用するため」と書きましたが、課題を見つけたり、その解決のための意思決定の根拠をさがしたりということでしょうか。分野は、さまざまですから、表現もまちまちかもしれません。最後に「科学」これも、簡単ではありませんが、わたしは、二つのことが大切だと考えています。一つは、反証可能性 (falsifiability^{*1}) です。もう一つは、再現可能性 (reproducibility^{*2}) です。反証可能性は、自分はそう信じるというような主観ではなく、正しいかどうかチェックすることができるという意味です。再現可能性は大体理解できると思いますが、データから得られる事実を、他のひとが操作をしても、同じ結果が得られるということです。そのためには、その「操作」が明確になっており、合理性も確保されていなければいけませんね。

これら二つの性質は、「活用するため」に、共通の客観的な基盤を保証するということだと思います。データサイエンスは、根拠を明確にした議論 (evidence based, fact based) のために、欠かせないことで、多様な価値観の多様なひとたちが合意して課題に立ち向かっていくためには、必要不可欠なことだと思います。そのいみでも、これから、データサイエンスは、ますます、重要度がましていくと思います。

*1 カール・ポパー (Karl Raimund Poppr, 1902-1994) が科学論において提唱した考え方]

*2 科学はすべて再現可能とは言えません。宇宙の成り立ちや、生物進化などは、再現可能ではありませんが、事実を確認しそれを根拠とした議論を考えると、可能な限り確保することが、反証可能性にもつながります。

もう少し、踏み込んで、データサイエンスについて書いてみましょう。実際に、どのような作業が重視されているのかも、含まれています。

Data Science: データ (Data) を活用して課題を発見・探求し、適切な解決策を探る意思決定のための科学 (Decision Science) で、エンピリカル (Empirical Study) すなわち、理論ではなく、実証性を特徴とする。データから得られる特徴を表示するとともに、数理モデルを適用し・機械学習などで評価し・アルゴリズムを策定する数理的思考を通して得られた結果を、可視化などによってコミュニケーションをおこない、共有し、他者の意見を聞き理解する努力をしながら、さらに課題について、あらたにデータを活用して考え、検証し、適切な解決策がもたらす新たな課題も予測しながら、調整をはかること。

上に書いたのは、ひとつの表現ですが、AI*³ に、表題の「データサイエンスとは」と聞いてみると、次のような答えをしてくれました。

ChatGPT (2023.6.3)

データサイエンス (Data Science) は、データを解析し、洞察を得るための科学的な手法やツールの開発と応用を含む、多様なスキルセットを組み合わせた領域です。データサイエンスは、統計学、機械学習、データマイニング、データ可視化などの技術や手法を利用して、大量のデータからパターンやトレンドを抽出し、意思決定に役立つ洞察を導き出すことを目指します。

データサイエンスのプロセスは、データの収集、前処理、分析、モデリング、評価、展開などのステップで構成されます。最初に、関連するデータを収集し、必要な形式に整理するために前処理を行います。次に、統計的な手法や機械学習アルゴリズムを使用してデータを分析し、予測モデルやパターンを発見します。モデルのパフォーマンスを評価し、必要に応じて改善を行い、最終的に洞察をビジネスや科学の現場に適用することが目標です。

データサイエンスは、さまざまな分野で活用されています。例えば、ビジネス分野では、マーケティングキャンペーンの最適化、顧客セグメンテーション、需要予測などに利用されます。医療分野では、病気の診断や治療計画の最適化、新薬の開発などに応用されます。また、交通や都市計画、環境保護、金融など、さまざまな領域でデータサイエンスの手法が活用されています。

データサイエンスには、数学や統計学、プログラミング、データベース管理、機械学習、ビジュアル化などの幅広いスキルが必要です。また、データの解釈やビジネス理解、コミュニケーション能力も重要な要素です。データサイエンティストやデータアナリストは、これらのスキルを駆使して、データを活用し、意思決定のサポートや新たな知見の発見に貢献します。

ちょっと日本語がぎこちない感じもするので、英語での問いと答えも書いておきましょう。

*³ どんどん進化していますから、答えを得た日付も加えておきます。一般的なことがらについてまとめて答えてくれるので、その意味で便利です。

What is datascience?

Data science is a multidisciplinary field that involves using scientific methods and tools to analyze data and extract insights. It encompasses a wide range of skills, including statistics, machine learning, data mining, and data visualization, to explore and understand complex data sets.

At its core, data science involves collecting, organizing, cleaning, and processing large volumes of data to uncover patterns, trends, and correlations. It applies various statistical techniques and algorithms to gain meaningful insights and make informed decisions. Data scientists employ exploratory data analysis, hypothesis testing, and predictive modeling to extract valuable information from data and generate actionable recommendations.

Data science finds applications in numerous industries and domains. It is widely used in business for market research, customer segmentation, demand forecasting, and personalized recommendations. In healthcare, data science aids in disease diagnosis, treatment optimization, and drug discovery. Other areas where data science is applied include transportation and urban planning, environmental conservation, finance, and social sciences.

To excel in data science, one needs a solid foundation in mathematics, statistics, programming, and data manipulation. Proficiency in programming languages such as Python or R is crucial for data manipulation and building models. Strong analytical skills, critical thinking, and the ability to communicate findings effectively are also essential for data scientists.

Data science is a rapidly evolving field, driven by advancements in technology and the increasing availability of data. It plays a crucial role in leveraging data assets to gain insights and make data-driven decisions in various industries.

学際的分野 (multidisciplinary field) という言葉も使われています。かなりの広がりをもった分野です。

1.1.1 なぜ、データサイエンスが最近注目されているのか

では、なぜ、最近、急に、データサイエンスが注目されるようになったのでしょうか。最初にも少し書きましたし、みなさんが、少しずつ学びながら、考えていただくのが一番よいと思いますが、いくつか理由があると思います。

個人的には、インターネットの普及や、さまざまなセンサーの普及にも助けられて、膨大なデータが得られるようになり、さらに、コンピュータの進化により、大きなデータであっても、それらを、扱うことができるようになったこと。それによって、複雑な問題であっても、その解決と意思決定に関して、データを使って根拠を明確にした (evidence based) 議論を支援することが可能になったことだと考えています。

付け加えておかないといけないと考えていることがもう一つあります。それは、問題解決と書きましたが、課題と向き合うという表現でもよいと思います。これまでは、因果関係が単純な課題については、根拠を示して、ある程度議論できましたが、実際の世の中の問題は、非常に多くの要素が相互に絡み合っており、それらを、調整すると、結果にも、さまざまな変化が生じるため、調整も非常に困難だということも挙げられると思います。

それだけ複雑な課題と向き合うには、データを丁寧にみる必要があると、その中で、さまざまな指標に関して評価値が高いものを選択することが可能になって来たということなのです。

そのことが、もう一歩進んで、AI (Artificial Intelligence 人工知能) を支える技術に使われています。ここで使った、ChatGPT の GPT は、Generative Pre-trained Transformer の略ですが、LLM (Large Language Model) によって、自然なことを生成できるような技術が実用段階に入っています。たとえば、「データサイエンスが最近注目されているのは...」とすると、この後に続く言葉として、評価値が高いものを、選択してくれる。それを、さらに、調整して、応答することが可能になっているのです。

少し長いですが、もう少し、踏み込んで説明していますので、ChatGPT の応答を下に挙げておきましょう。

ChatGPT (2023.6.3)

データサイエンスが最近注目されている理由は、以下の要素によるものです。

1. ビッグデータの増加: 近年、デジタル技術の進歩により、膨大な量のデータが生成されるようになりました。ウェブサイトのアクセスログ、ソーシャルメディアの投稿、センサーデータ、トランザクションデータなど、多様なソースから大量のデータが利用可能です。データサイエンスは、これらのビッグデータを活用して洞察を得ることができるため、需要が高まっています。
2. 技術の進歩: データサイエンスを支える技術の進歩も、その注目を高めています。クラウドコンピューティングの発展やオープンソースのツールとライブラリの普及により、データの収集、処理、分析が容易になりました。さらに、機械学習や深層学習のアルゴリズムの改善や高速化も進んでおり、より高度な予測モデルやパターンの抽出が可能になっています。
3. ビジネス価値の認識: データは企業にとって貴重な資産であり、データを活用することで競争優位を得ることができます。データサイエンスを活用することで、マーケットのニーズやトレンドを理解し、効果的なマーケティング戦略を立てることができます。顧客行動の予測やセグメンテーションを行い、顧客満足度や収益性を向上させることも可能です。
4. 知識と人材の需要: データサイエンスの分野での知識とスキルを持つ人材への需要が高まっています。企業や組織は、データサイエンスの専門家を雇用して、データの分析と活用を行いたいと考えています。また、データサイエンスは学問としても注目を浴びており、大学や研究機関ではデータサイエンスに関連す

るプログラムや研究が増えています。

これらの要素により、データサイエンスはビジネスや科学の分野で重要な役割を果たしており、その注目度が高まっています。以下に、その具体的な理由をいくつか挙げます。

5. 革新と競争力の追求: データサイエンスは新たなビジネスモデルや製品の開発に貢献します。データを活用して市場のトレンドや消費者のニーズを洞察し、それに基づいて新しいサービスや製品を開発することが可能です。企業はデータを駆使して競争力を高め、革新的なソリューションを提供することを求めています。
6. 問題解決と意思決定の支援: データサイエンスは複雑な問題を解決し、意思決定をサポートする上で重要なツールとなっています。データの分析により、事実に基づいた洞察が得られ、ビジネス上の課題や機会を特定することができます。データサイエンスの手法はリスク評価や予測、最適化などにも応用され、効果的な戦略立案や意思決定を支援します。
7. パーソナライズされた体験の提供: データサイエンスは顧客の行動や好みを理解し、パーソナライズされた体験を提供するための手段としても利用されます。顧客データや行動履歴を分析し、個別のニーズや関心に基づいたオファーや推奨を行うことで、顧客満足度や忠誠度を向上させることができます。
8. 社会的なインパクトと公共政策への応用: データサイエンスは社会的な課題や公共政策の分析にも役立ちます。例えば、交通渋滞の予測や都市計画の最適化、犯罪予測や防犯対策の強化などにデータサイエンスが活用されています。データ駆動のアプローチは社会のさまざまな側面において効果的な解決策を提供する可能性があります。

これらの要素により、データサイエンスはビジネスや科学の分野で重要な役割を果たしており、その注目度が高まっています。以下に、その具体的な理由をいくつか挙げます。

1. 自動化と効率化の追求: データサイエンスは、業務プロセスの自動化や効率化にも貢献します。データの分析と予測モデルの活用により、タスクの自動化や最適なリソースの割り当てが可能となります。これにより、人的な作業負担を軽減し、生産性を向上させることができます。
2. 組織内のデータ文化の普及: データサイエンスの注目度の高まりに伴い、企業や組織においてデータ文化の重要性が認識されています。データサイエンスの手法やアプローチを組織全体で活用し、データに基づいた意思決定や問題解決を行うことが求められています。組織内でのデータの収集、管理、分析の能力向上に取り組むことで、競争力を高めることができます。
3. 専門化と職業の多様化: データサイエンスの注目度の高まりに伴い、関連する職業や専門化も増えています。データサイエンティスト、データアナリスト、

機械学習エンジニアなどの役割が求められるようになりました。また、データサイエンスを支えるツールやプラットフォームの開発も進んでおり、それに関連する職業やスキルも増加しています。

総じて、データサイエンスの注目度が高まっている理由は、データの増加、技術の進歩、ビジネス価値の認識、革新と競争力の追求、問題解決と意思決定の支援、パーソナライズされた体験の提供、社会的なインパクトと公共政策への応用、自動化と効率化の追求、組織内のデータ文化の普及、専門化と職業の多様化など、さまざまな要素が重なっています。

データサイエンスは、ビジネスや科学の領域でデータの価値を最大限に引き出すための力強いツールとなっており、今後もその重要性と注目度は高まっていくことが予想されます。

1.1.2 なぜ、一人ひとりが、データサイエンスを、学ぶ必要があるのでしょうか

専門家に任せてはいけないのでしょうか。

個人的には、次のように考えています。非常に広い範囲の課題に関係している課題に、データサイエンスが使われるだけでなく、その背後にある、単純な、因果関係ではない、考え方を、理解する必要があることが挙げられると思います。

さらに、わたしが特に大切だと考えているのは、個人のコンピュータでも、十分に利用可能であること、オープンデータとか、パブリックデータと言われる、誰でも利用できるデータが、使いやすい形で、提供されているということです。

ChatGPT (2023.6.3)

一人ひとりがデータサイエンスを学ぶ必要がある理由は、以下のような点が挙げられます。

1. データリテラシーの重要性: データは現代社会においてますます重要な役割を果たしています。データに基づく意思決定や問題解決は、個人の日常生活から組織や社会の活動まで広がっています。データサイエンスを学ぶことで、データを扱うスキルや知識を身につけることができます。データを理解し、適切に分析して活用する能力は、個人の能力向上や自己啓発にも繋がります。
2. 個人の成長とキャリアの展望: データサイエンスは求人市場で高い需要があり、競争力のあるキャリアの一つとなっています。データサイエンスのスキルを持つことで、自身のキャリアの成長や就業機会の拡大が期待できます。データサイエンスの知識やスキルは、さまざまな職業や業界で役立つだけでなく、起業や独立の道も開ける可能性があります。
3. 自己啓発と問題解決能力の向上: データサイエンスは、問題解決や意思決定において非常に有用なツールです。データサイエンスを学ぶことで、情報を分析

し洞察を得る力や、パターンを見つける力、予測や最適化の能力を養うことができます。これにより、個人の問題解決能力や意思決定力が向上し、より効果的な選択や行動ができるようになります。

専門家に任せることも重要ですが、一人ひとりがデータサイエンスの基本的な概念や手法を理解することは有益です。以下にその理由を説明します。

4. データの信頼性と誤解の回避: データは私たちの日常生活やビジネスの意思決定に影響を与える重要な要素です。しかし、データの解釈や分析方法によっては誤解を招くこともあります。データサイエンスの知識を持つことで、データの信頼性を評価し、適切な解釈を行う能力が向上します。これにより、誤った結論や誤解に基づく意思決定を避けることができます。
5. 自己管理とデータプライバシーの保護: 近年、個人のデータの重要性和プライバシーの問題が注目されています。データサイエンスを学ぶことで、自身のデータの管理方法やプライバシーの保護について理解し、適切な対策を取ることができます。データの収集や利用に対して主体的に関与し、自己管理の意識を持つことが重要です。
6. クリティカルシンキングと問題解決能力の向上: データサイエンスの学習は、クリティカルシンキングや問題解決能力を向上させる助けとなります。データを扱い、分析するためには、情報を適切に評価し、論理的に考える必要があります。データサイエンスの学習は、証拠に基づく意思決定や論理的な議論のスキルを磨くのに役立ちます。

データサイエンスの専門家に相談することは重要ですが、一人ひとりが基本的なデータサイエンスの知識やスキルを身につけることで、自己啓発や問題解決能力の向上、データの信頼性の評価、プライバシーの保護などが可能となります。データサイエンスの学習は、個人の成長やキャリアの展望にもプラスの影響を与えることが期待されます。

1.1.3 AI の活用

「データサイエンスをはじめましょう」では、AI も活用していきたいと思います。クリティカルシンキングのことも、書かれていますが、どうしても、人は、バイアスのかかった考えかたをしてしまいます。それは、経験も知識も限られていますし、単純な答えを受け入れたい傾向^{*4}ということもあるかもしれません。

そのために、まず、仲間に聞いてみる。そして、AI に聞いてみる。それらを踏まえて、さらに、課題や、問いを精緻なものにしていく。それが、まず、第一に大切だと考えるからです。

さらに、最近の AI は、人間の普通の言葉を理解して、コード（コンピュータに伝える言

^{*4} 私はこれを、単純化バイアスと呼んでいます。遺伝子に組み込まれているのかもしれませんが。しかし、それに抗うのも、人間としてのたいせつな生き方のように思います。

葉)を書くこともできるようになっています。そのような支援を有効に使うことで、プログラミングの訓練を受けていない人でも、データサイエンスを学ぶことができるようになって来ています。

翻訳にも活用することが可能で、長い文章の要約もかなり上手にしてくれるようになってきます。

AIはまだ、発展途上で、ここに書いたことも、少し経つと、古くなってしまふことでしょう。しかし、少し長いスパンで考えると、AI研究の目標値として、掲げられているのは、AGI (Artificial General Intelligence)、汎用人工知能。(人間が実現可能な)あらゆる知的作業を理解・学習・実行することができる人工知能ということです。しかし、課題も多くあります。課題について、AIに聞いてみると次のような回答をしてくれました。

Q. 現在の General Artificial Intelligence の課題をまとめてください。

GPT4 (2023.8.8)

現在の一般人工知能 (General Artificial Intelligence, AGI) には、以下のような主要な課題があります。

1. 理解と推論の限界: 現在の AI は、人間のように世界を理解したり、複雑な推論を行う能力をまだ持っていません。これは、言語理解、感情理解、一般的な常識理解など、多くの問題領域に影響を与えます。
2. 教育データのバイアス: AI は学習データからパターンを学びますが、そのデータが偏っている場合、AI も偏った結果を出す可能性があります。これは、AI が人間の先入観や社会的な偏見を反映する可能性を示しています。
3. 説明可能性と透明性の欠如: AI の決定過程はしばしば「ブラックボックス」とされ、その理由は明らかにされていません。これは、AI の決定がなぜ行われたのか、どのような情報に基づいて行われたのかを理解するのを難しくしています。
4. 安全性とエチカルな問題: AI が自己学習や自己改良を行う能力が増すにつれて、その結果が予測不可能になる可能性があります。また、AI が人間の倫理観や法律を理解し、それに従うようにすることも大きな課題です。
5. リソースの要求: 大規模な AI モデルの訓練は、大量の計算リソースと電力を必要とします。これは環境に対する影響や、AI 開発の費用効率に関わる問題です。

たしかに、どれも、難しそうな課題ですね。みなさんは、これから、どのように、AI を活用し、どのように、向き合われるのでしょうか。

1.2 Gapminder

Gapminder はオラロスリング (Ola Rosling) とアンナロスリング (Anna Rosling Rönnlund) とハンスロスリング (Hans Rosling) が設立した組織です。

ハンス・ロスリングは、FACTFULNESS (ファクトフルネス) 10 の思い込みを乗り越え、データを基に世界を正しく見る習慣の著者です。

Gapminder: <https://www.gapminder.org>

一番上にテストあります。- もしかしたら、あなたの世界観は、間違っているかもしれません。(You are probably wrong about - upgrade your worldview)

Bubble Chart: [https://www.gapminder.org/tools/#\\$chart-type=bubbles&url=v1](https://www.gapminder.org/tools/#$chart-type=bubbles&url=v1)

Dollar Street: <https://www.gapminder.org/dollar-street>

Data: <https://www.gapminder.org/data/>

1.2.1 Factfulness (ファクトフルネス) Hans Rosling

実際のデータから、現実を見ていないと、非常に歪んだ世界観に毒されているかもしれません。ハンス・ロスリングは、公衆衛生が専門のスウェーデンの医師で、アフリカでも年間も働いた経験ももっている方で、ひとは、さまざまなバイアス (先入観) により、間違った根拠のもとで世界を見ているといい、統計的な資料から、息子のオラと協力して、非常にわかりやすい、バブルチャートでの時代の移り変わりを表現し、サイトに載せています。また、それだけでは、実際の生活が見えにくいという理由から、息子の奥さんのアンナさんの提案で、ダラー・ストリートというプロジェクトもしています。この程度の経済状態の暮らしがどのようなものかを、写真などで、紹介するものです。

先生から教わった知識は、先生が勉強した時代にはある程度正しかったかもしれないが、世界は変化している。自分の経験から、判断すると、その先入観から、多くの間違いを犯すというような指摘もしています。

1. 分断本能を抑えるには、大半の人がどこにいるかを探そう。
2. ネガティブ本能を抑えるには、悪いニュースのほうが広まりやすいことを覚えておこう。
3. 直線本能を抑えるには、直線もいつかは曲がることを知ろう、
4. 恐怖本能を抑えるには、リスクを計算しよう。
5. 過大視本能を抑えるには、数字を比較しよう。
6. パターン化本能を抑えるには、分類を使おう。

7. 宿命本能を抑えるには、ゆっくりとした変化でも変化していることを心に留めよう。
8. 単純化本能を抑えるには、ひとつの知識がすべてに應用できないことを覚えておこう。
9. 犯人捜し本能を抑えるには、誰かを責めても、問題は解決しないと肝に銘じよう。
10. 焦り本能を抑えるには、小さな一歩を重ねよう。

(ファクトフルネスから)

データサイエンスはこれらのいくつかを克服するひとつの方法であるように見えます。

ハンス・ロスリングは、すでに2017年2月7日に亡くなっていますが、心配している5つのリスクとして、挙げているのは、感染症の世界的な流行、金融危機、世界大戦、地球温暖化、極度の貧困。どれも示唆に富んでいるように見えます。

Youtube にたくさん、ビデオも出ていますが、二つだけリンクを載せておきます。

- How not to be ignorant about the world | Hans and Ola Rosling
– バイアスを意識しながら (日本語の字幕がついています)
- The best stats you've ever seen, Hans Rosling

1.2.2 参考文献

- 「私はこうして世界を理解できるようになった」ハンス・ロスリング、ファニー・ヘルエスタム著、青土社 (ISBN978-4-7917-7217-9, 2019.10.10) “How I learned to understand the world” by Hans Rosling with Fanny Haergestam の翻訳
- 「ファクトフルネス - 10の思い込みを乗り越え、データを基に世界を正しく見る習慣」ハンス・ロスリング、オーラ・ロスリング、アンナ・ロスリング・ロランド著、日経BP社 (ISBN978-4-8222-8960-7, 2019.1.15)

1.3 「データサイエンスを始めましょう」の特徴

1.3.1 学習者として想定しているのは

高等学校を卒業したひとたちを対象と想定して、書いていこうと思います。

1.3.2 オープン・パブリックデータの活用

データサイエンスは、広い分野ですが、ここでは、オープンデータとか、パブリックデータと言われるものを主として活用していきます。

すでに、書いたように、インターネットに繋がったコンピュータがあれば、だれでも、データサイエンスを学び、データから智を得ることが可能になって来ています。特に、世界

に目をむけると、すばらしいサイトがたくさんあり、国際機関などが、膨大なデータを提供しているので、まずは、それを活用したいと思います。

1.3.3 世界のデータをみること

日本のデータも、使っていこうと思いますが、まずは、世界の中の課題をみる必要があります。扱いやすい、世界のデータがたくさんあることも理由の一つです。

むしろ、日本の課題から目を逸（そ）らすわけではありません。世界の中の日本を意識し、日本の課題にも目を向けていきたいと思います。

1.3.4 目標としていること

ここで扱う内容は限られていますが、データサイエンスの基本を身につけることで、ここで、取り上げる、オープンデータ、パブリックデータだけでなく、さまざまな課題にデータを通して、向き合うことができると考えています。

ここまで学べばというゴールはありません。日常的に、データを通して、課題に向き合う習慣が身につけていけばと願っています。

1.4 学習方法について

インターネット上で公開していますので、さまざまな方法で学んでいただきたいと思います。個人で学ぶことも可能で、実際に、それが可能なように、書いていく予定です。

しかし、おすすめは、何人かのグループ、または、大学などで一緒に学ぶことです。それは、データサイエンスの目的でもある、問いを持ち、課題に取り組んでいくためには、さまざまな視点からの意見や、考え方が必要だからです。異なる見方をたいせつにする訓練にもなります。

ひとつのグラフから、それぞれが違うことを発見することも多くあります。それを経験しながら、共に考えていく経験が貴重だと思います。

さらに、データの背景にあることを、想像したり、情報を得るために、グループの一員として、AIに加わってもらうことも、有効だと思います。できれば、複数のAIに質問をするのが良いでしょう。

さまざまな課題に、向き合うときに、グループのメンバーや、AIは、多様な意見を言ってくれることと思います。

データを元にした議論の訓練をすることで、根拠を明確にした説明をする訓練もすることができます。そのためにも、グループで学ぶことは有益です。

コードも、AIは教えてくれますが、聞き方が適切ではないと、間違った回答が得られることもあります。それも、グループで議論することで、聞き方を工夫していくことも可能

だと思えます。

人の前で、発表する練習も、一連の学びの中で経験していくことをお勧めします。ぜひ、みなさんにあった、学びの場を協力して創って行ってください。

1.5 参考

- 対話型 AI - Chat Bot について
 - AI の使い方や例について、書いてあります。参考にしてください。
- Data Analysis for Researchers 2022
 - オープンデータを用いた、データ分析の授業のデジタルブック

第 2 章

学ぶ内容

2.1 データサイエンス入門

具体的なデータを利用して、データサイエンスとは、どのようなものかをみていきながら、ここで学ぶことの概要を紹介します。オープン・パブリックデータとしては、世界銀行のデータを使います。さまざまなデータが公開され、簡単に取得できるようになっている現状も紹介します。コードの詳細には、こだわらず、データサイエンスの実際について、雰囲気を感じていただければと思います。

2.2 第一部 オープンデータ

世界のさまざまな、パブリックデータの紹介をし、ダッシュボードと呼ばれる機能を活用して、データをみることをします。

世界銀行の世界開発指標 (WDI)、国際連合 (UN Data)、OECD、日本のデータ (e-Stat) を外観します。

ここでは、R は使わず、サイトが提供するデータを探したり、サイト内でグラフを作成したり、データを取得するには、どのような方法があるかなどを紹介したいと思います。

これらの機関内の機能を、ホームページ閲覧ソフト (Google Chrome, Edge, Safari など) を使うだけで、かなりの情報が得られることを、経験していただければと思います。

2.3 第二部 R の基礎

R の基本を学びます。R は、もともと、統計解析ソフトとして、開発されたもので、さまざまな分野の研究者によって利用され、また、それぞれの分野に必要な機能を、パッケージという形で開発して発展してきた言語です。非常に多くのひとたちが、開発に加わったために、痒いところに手が届く、多くの機能を、パッケージによって使うことができるようになりました。しかし、他方、統一性は十分ではなく、少し複雑な作業を実行するための、プログラミング言語としての機能も十分ではないという欠点も生じました。

わたしの理解では、それを一気に解決したのが、Hadley Wickham 等、その後、RStudio そして、現在の、Posit に引き継がれた、tidyverse というパッケージ群の開発です。他の研究者も、tidyverse の開発思想を受け継ぎ、発展させる形で、開発をしています。

そこで、R の起動とともに、最初に読み込まれる、Base R など、基本パッケージに、tidyverse を加えたものを基本として、極力、これらだけで、基本を学んでいきたいと思えます。実際には、他のさまざまな便利なパッケージを使うことも、有用ですが、それは、後に回して、tidyverse を中心に学んでいきます。

tidyverse により、R は、プログラミング言語としても、一つの優秀な言語となったと思えます。コーディングや、プログラミングと言われる、一つ一つのステップを構築し、それを繋げていくことを、学んでいきたいと思えます。

もう一つ追加しておくのは、R Markdown の活用です。この「データサイエンスをはじめましょう」も、R Markdown の一つの形式、bookdown を使って書いています。

データサイエンスを学ぶ上で、わたしが必要かつ不可欠と考えているのが、再現性 (Reproducibility) と、なにを執行しているのかの説明を同時に記述すること (Literate Programming) です。コードとともに、その結果を、その下に出し、かつ、そのコードの説明も加え、さらに、それによって、何がわかるかも、同時に書いていくことは、データサイエンスの核となすもので、それによって、データサイエンスの目的を達成することができると思っています。

データサイエンスでは、最後のコミュニケーションまでがひとつのまとまりです。他の人に聞いてもらうために発表したり、読んでもらうために、レポートを作成することも、一連の流れに加えることが必須だと思います。

指導して下さる方がいるときは、そのレポートをみてもらって、評価してもらったり、アドバイスを受けることは不可欠でしょう。それには、そのレポートに、コードとともに結果も書かれており、さらに、それは、何のために、そこから、何が得られるのかが書かれていることも必要です。

R Markdown の活用も、ともに学んでいきたいと思えます。

2.4 第三部 オープンデータの活用

R を使って、第一部で概観したデータを実際に分析する手法を学びます。

国際機関などの公的機関では、さまざまなデータを提供していますが、それぞれに特徴があり、データの形式や、データ取得の方法が異なります。それらを、少しずつ説明しながら、それぞれのデータを、すでに学んだことを応用しながら、分析する実際を経験していきたいと思えます。

世界銀行の、世界開発指標 (World Development Indicators) が、一番整っているので、まずは、世界開発指標から学びますが、世界銀行の他のデータや、国際連合のデータ、他の国際連合の機関が提供しているデータや、経済開発協力機構 (OECD) や、Our World

in Data、Euro Stat などと共に、日本のデータである、e-Stat の使い方も学びたいと思います。

国際機関だけではなく、他にも、オープン・パブリックデータを提供しているところがたくさんあります。少しずつその利用方法も含めて、紹介していきたいと思います。

2.5 第四部 探索的データ分析 Exploratory Data Analysis

データを分析していくには、基本的なステップがありますが、その一つ一つのステップについて、より詳しく学びます。

これまでに、紹介できなかったいくつかの手法についても、紹介していきたいと思います。

2.6 第五部 分析例

実際の分析例を加えていきたいと思います。

2.7 付録

技術的なコメントなど、幾つかのトピックについて書いていきます。

だいたい、このような構成を考えています。

2.8 パッケージについて

R を使い始めると、追加のパッケージを使う必要が生じます。本書では、特に、`tidyverse` パッケージ（群）を主として使いますが、他にも、世界銀行の、世界開発指標（WDI）を読み込むための、パッケージ `WDI` なども使います。ここには、本書で使う、パッケージのリストを上げておきます。パッケージはインストールし、さらに、使えるように読み込む必要がありますが、その情報も少し加えておきます。

2.8.1 主として利用するパッケージ

- `tidyverse`
 - 読み込み：`tidyverse`, `readxl`, `stringr`
- `WDI`

2.8.2 特定の章で利用するパッケージ

- `tidymodels`

- devtools
 - wid-r-tool: `devtools::install_github("WIDworld/wid-r-tool")`
 - owidR: `devtools::install_github("piersyork/owidR")`
- estatapi
- OECD
- eurostat
- mapproj
- geodata
- rnaturalearth
- rnaturalearthdata
- showtext

第3章

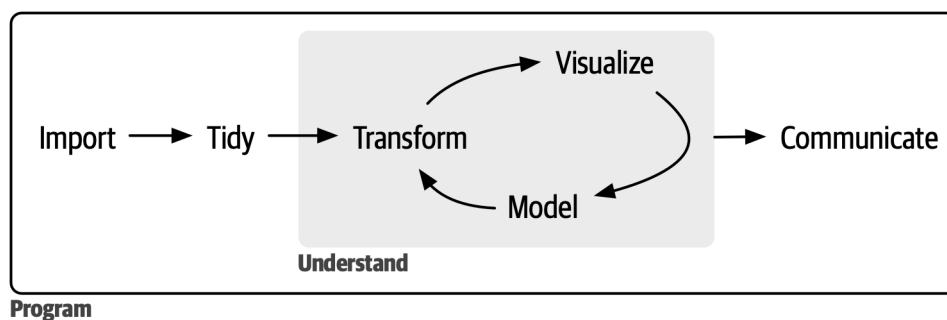
はじめてのデータサイエンス

3.1 データサイエンスの実際

データから情報を得るときには、大体次のような手順をとります。

1. 準備 Setup
2. データを取得 Import data
3. データ構造の確認 View data
4. 必要に応じて整形 Transform data
5. 視覚化 Visualize data
6. データを理解 Understand data
7. レポートなどにまとめる Communicate data

下の図は R for Data Science に掲載されている図です。よく、表現されていると思います。詳細は、少しずつ説明します。



はじめにに書きましたが、基本的には、問いをもちデータを取得し、視覚化などを通して、データを理解し、さらに問いを深めるサイクルが、データサイエンスの核だと思います。

Rを使った分析の一つの例を、見て行きます。一つ一つのコード（コンピュータ・プログラム）の簡単な説明は、加えますが、あまりそれに捉われず、「データサイエンスとは何か？」を考えながら、雰囲気を味わってください。

3.2 Rのパッケージを活用

3.2.1 準備 Setup

世界銀行 (World Bank) の、世界開発指標 (WDI: World Development Indicators) の一つの、GDP (Gross Domestic Product 国内総生産) のデータから始めます。GDP にも何種類かの尺度がありますが、次のものを見てみます。

- NY.GDP.MKTP.CD: GDP (current US\$)*¹

NY.GDP.MKTP.CD は、データコードと言われるもので、世界開発指標 (WDI) には、一つずつ決まっています。

World Development Indicators のサイトの下にある、Data Themes (テーマ) からテーマを選択し、下にスクロールすると、Code をみることができます。ちなみに、ここで利用する NY.GDP.MKTP.CD: GDP (current US\$) は、テーマ Economy (経済) の、一番上にあります。

経済用語の英語はよく知らないという方は、ブラウザ (Edge, Google Chrome, Safari など) の翻訳機能を使うのも良いでしょう。ただ、そのページの対話型の機能 (interactive function) を利用するときは、翻訳機能を OFF にする必要がある場合もありますので、注意してください。

エラーメッセージを調べるときなどに、英語のほうが情報がたくさん得られますから、言語を、英語に変更しておきます。

R には、WDI のデータを取得する R のツール (パッケージ) WDI がありますから、それを使います。また、データを取り扱うための基本的なツール (パッケージ) tidyverse を使いますので、次のコードで、これらを読み込みます。# 以下はコメント (簡単な説明を書きました) #> 以下は、コードを実行すると、表示される情報 (出力) です。以下同様です。

```
Sys.setenv(LANG = "en") # 言語を英語に
library(tidyverse)      # tidyverse パッケージを読み込みます
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats    1.0.0      v stringr    1.5.0
#> v ggplot2    3.4.3      v tibble     3.2.1
#> v lubridate  1.9.2      v tidyr      1.3.0
#> v purrr      1.0.2
```

*¹ GDP (Gross Domestic Product) とは、ある国のある期間 (通常は 1 年) における、その国で生産されたすべての最終財・サービスの市場価値の総額を指します。これは国内総生産とも呼ばれます。GDP は、その国の経済力や活力を測る指標の 1 つとして広く用いられています。WDI の GDP (Current USD) は、各国の GDP を米ドルで表したものであり、通貨の価値が異なっても比較可能な形で国際比較ができます。ただし、GDP はあくまで市場価格を基準としているため、非市場活動や自然災害などの影響を受ける場合がある点に注意が必要です。


```
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag() masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(WDI) # WDI パッケージを読み込みます
```

データを保存する場所を作成しておくことをお勧めします。保存しておくときは、このディレクトリを使います。

```
dir.create("./data")
```

3.2.2 データ取得 Import data

データを取得します。少し時間がかかります。取得したデータに、df_gdp などと、わかりやすい名前をつけます。df は data frame の略で、R で標準的なデータの形式です。

```
df_gdp <- WDI(country = "all",
              indicator = c(gdp = "NY.GDP.MKTP.CD"),
              extra = TRUE)
```

このコードで、全ての国の GDP を取得できます。GDP の値は、NY.GDP.MKTP.CD という名前の列にありますが、覚えやすいように、gdp という名前に変更しておきます。extra = TRUE とすることによって、それぞれの国についての情報などが追加されます。

3.2.3 データ構造の確認

最初の数行だけを見るには、head(df_gdp) とします。

```
head(df_gdp)
#> # A tibble: 6 x 13
#>   country iso2c iso3c year   gdp status lastupdated region
#>   <chr>   <chr> <chr> <dbl> <dbl> <lgl> <date>   <chr>
#> 1 Afghan~ AF    AFG    1963 7.51e8 NA    2023-07-25 South~
#> 2 Afghan~ AF    AFG    1962 5.47e8 NA    2023-07-25 South~
#> 3 Afghan~ AF    AFG    1961 5.49e8 NA    2023-07-25 South~
#> 4 Afghan~ AF    AFG    1960 5.38e8 NA    2023-07-25 South~
#> 5 Afghan~ AF    AFG    2003 4.54e9 NA    2023-07-25 South~
#> 6 Afghan~ AF    AFG    2002 3.85e9 NA    2023-07-25 South~
#> # i 5 more variables: capital <chr>, longitude <dbl>,
#> # latitude <dbl>, income <chr>, lending <chr>
```

データの構造を見るときには、str(df_gdp) もよく使われます。今度は、列が縦に並んで表示されます。

```

str(df_gdp)
#> spc_tbl_ [16,758 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
#> $ country      : chr [1:16758] "Afghanistan" "Afghanistan" "Afghanistan" "Afgha
#> $ iso2c        : chr [1:16758] "AF" "AF" "AF" "AF" ...
#> $ iso3c        : chr [1:16758] "AFG" "AFG" "AFG" "AFG" ...
#> $ year         : num [1:16758] 1963 1962 1961 1960 2003 ...
#> $ gdp          : num [1:16758] 7.51e+08 5.47e+08 5.49e+08 5.38e+08 4.54e+09 ...
#> $ status       : logi [1:16758] NA NA NA NA NA NA ...
#> $ lastupdated: Date[1:16758], format: "2023-07-25" ...
#> $ region       : chr [1:16758] "South Asia" "South Asia" "South Asia" "South AS
#> $ capital       : chr [1:16758] "Kabul" "Kabul" "Kabul" "Kabul" ...
#> $ longitude    : num [1:16758] 69.2 69.2 69.2 69.2 69.2 ...
#> $ latitude     : num [1:16758] 34.5 34.5 34.5 34.5 34.5 ...
#> $ income       : chr [1:16758] "Low income" "Low income" "Low income" "Low inco
#> $ lending      : chr [1:16758] "IDA" "IDA" "IDA" "IDA" ...
#> - attr(*, "spec")=
#> .. cols(
#> ..   country = col_character(),
#> ..   iso2c = col_character(),
#> ..   iso3c = col_character(),
#> ..   year = col_double(),
#> ..   gdp = col_double(),
#> ..   status = col_logical(),
#> ..   lastupdated = col_date(format = ""),
#> ..   region = col_character(),
#> ..   capital = col_character(),
#> ..   longitude = col_double(),
#> ..   latitude = col_double(),
#> ..   income = col_character(),
#> ..   lending = col_character()
#> .. )
#> - attr(*, "problems")=<externalptr>

```

概要 (`summary(df_gdp)`) からもある程度わかります。

```

summary(df_gdp)
#>   country          iso2c          iso3c
#> Length:16758      Length:16758      Length:16758
#> Class :character  Class :character  Class :character
#> Mode  :character  Mode  :character  Mode  :character
#>
#>

```

```
#>
#>
#>      year      gdp      status
#> Min.   :1960   Min.   :8.825e+06   Mode:logical
#> 1st Qu.:1975   1st Qu.:2.523e+09   NA's:16758
#> Median :1991   Median :1.843e+10
#> Mean   :1991   Mean   :1.207e+12
#> 3rd Qu.:2007   3rd Qu.:2.244e+11
#> Max.   :2022   Max.   :1.006e+14
#>      NA's   :3393
#> lastupdated      region      capital
#> Min.   :2023-07-25   Length:16758   Length:16758
#> 1st Qu.:2023-07-25   Class :character   Class :character
#> Median :2023-07-25   Mode  :character   Mode  :character
#> Mean   :2023-07-25
#> 3rd Qu.:2023-07-25
#> Max.   :2023-07-25
#>
#> longitude      latitude      income
#> Min.   : -175.22   Min.   : -41.286   Length:16758
#> 1st Qu.: -15.18   1st Qu.:  4.174   Class :character
#> Median :  19.54   Median : 17.277   Mode  :character
#> Mean   :  19.16   Mean   : 18.740
#> 3rd Qu.:  50.53   3rd Qu.: 39.715
#> Max.   : 179.09   Max.   : 64.184
#> NA's   :3528     NA's   :3528
#> lending
#> Length:16758
#> Class :character
#> Mode  :character
#>
#>
#>
#>
```

国のリストをみてみましょう。とても長いリストの中には、地域名も含まれています。

```
df_gdp |> distinct(country) |> pull()
#> [1] "Afghanistan"
#> [2] "Africa Eastern and Southern"
#> [3] "Africa Western and Central"
#> [4] "Albania"
```

```
#> [5] "Algeria"  
#> [6] "American Samoa"  
#> [7] "Andorra"  
#> [8] "Angola"  
#> [9] "Antigua and Barbuda"  
#> [10] "Arab World"  
#> [11] "Argentina"  
#> [12] "Armenia"  
#> [13] "Aruba"  
#> [14] "Australia"  
#> [15] "Austria"  
#> [16] "Azerbaijan"  
#> [17] "Bahamas, The"  
#> [18] "Bahrain"  
#> [19] "Bangladesh"  
#> [20] "Barbados"  
#> [21] "Belarus"  
#> [22] "Belgium"  
#> [23] "Belize"  
#> [24] "Benin"  
#> [25] "Bermuda"  
#> [26] "Bhutan"  
#> [27] "Bolivia"  
#> [28] "Bosnia and Herzegovina"  
#> [29] "Botswana"  
#> [30] "Brazil"  
#> [31] "British Virgin Islands"  
#> [32] "Brunei Darussalam"  
#> [33] "Bulgaria"  
#> [34] "Burkina Faso"  
#> [35] "Burundi"  
#> [36] "Cabo Verde"  
#> [37] "Cambodia"  
#> [38] "Cameroon"  
#> [39] "Canada"  
#> [40] "Caribbean small states"  
#> [41] "Cayman Islands"  
#> [42] "Central African Republic"  
#> [43] "Central Europe and the Baltics"  
#> [44] "Chad"  
#> [45] "Channel Islands"
```

```
#> [46] "Chile"  
#> [47] "China"  
#> [48] "Colombia"  
#> [49] "Comoros"  
#> [50] "Congo, Dem. Rep."  
#> [51] "Congo, Rep."  
#> [52] "Costa Rica"  
#> [53] "Cote d'Ivoire"  
#> [54] "Croatia"  
#> [55] "Cuba"  
#> [56] "Curacao"  
#> [57] "Cyprus"  
#> [58] "Czechia"  
#> [59] "Denmark"  
#> [60] "Djibouti"  
#> [61] "Dominica"  
#> [62] "Dominican Republic"  
#> [63] "Early-demographic dividend"  
#> [64] "East Asia & Pacific"  
#> [65] "East Asia & Pacific (excluding high income)"  
#> [66] "East Asia & Pacific (IDA & IBRD countries)"  
#> [67] "Ecuador"  
#> [68] "Egypt, Arab Rep."  
#> [69] "El Salvador"  
#> [70] "Equatorial Guinea"  
#> [71] "Eritrea"  
#> [72] "Estonia"  
#> [73] "Eswatini"  
#> [74] "Ethiopia"  
#> [75] "Euro area"  
#> [76] "Europe & Central Asia"  
#> [77] "Europe & Central Asia (excluding high income)"  
#> [78] "Europe & Central Asia (IDA & IBRD countries)"  
#> [79] "European Union"  
#> [80] "Faroe Islands"  
#> [81] "Fiji"  
#> [82] "Finland"  
#> [83] "Fragile and conflict affected situations"  
#> [84] "France"  
#> [85] "French Polynesia"  
#> [86] "Gabon"
```

```
#> [87] "Gambia, The"  
#> [88] "Georgia"  
#> [89] "Germany"  
#> [90] "Ghana"  
#> [91] "Gibraltar"  
#> [92] "Greece"  
#> [93] "Greenland"  
#> [94] "Grenada"  
#> [95] "Guam"  
#> [96] "Guatemala"  
#> [97] "Guinea"  
#> [98] "Guinea-Bissau"  
#> [99] "Guyana"  
#> [100] "Haiti"  
#> [101] "Heavily indebted poor countries (HIPC)"  
#> [102] "High income"  
#> [103] "Honduras"  
#> [104] "Hong Kong SAR, China"  
#> [105] "Hungary"  
#> [106] "IBRD only"  
#> [107] "Iceland"  
#> [108] "IDA & IBRD total"  
#> [109] "IDA blend"  
#> [110] "IDA only"  
#> [111] "IDA total"  
#> [112] "India"  
#> [113] "Indonesia"  
#> [114] "Iran, Islamic Rep."  
#> [115] "Iraq"  
#> [116] "Ireland"  
#> [117] "Isle of Man"  
#> [118] "Israel"  
#> [119] "Italy"  
#> [120] "Jamaica"  
#> [121] "Japan"  
#> [122] "Jordan"  
#> [123] "Kazakhstan"  
#> [124] "Kenya"  
#> [125] "Kiribati"  
#> [126] "Korea, Dem. People's Rep."  
#> [127] "Korea, Rep."
```

```
#> [128] "Kosovo"  
#> [129] "Kuwait"  
#> [130] "Kyrgyz Republic"  
#> [131] "Lao PDR"  
#> [132] "Late-demographic dividend"  
#> [133] "Latin America & Caribbean"  
#> [134] "Latin America & Caribbean (excluding high income)"  
#> [135] "Latin America & the Caribbean (IDA & IBRD countries)"  
#> [136] "Latvia"  
#> [137] "Least developed countries: UN classification"  
#> [138] "Lebanon"  
#> [139] "Lesotho"  
#> [140] "Liberia"  
#> [141] "Libya"  
#> [142] "Liechtenstein"  
#> [143] "Lithuania"  
#> [144] "Low & middle income"  
#> [145] "Low income"  
#> [146] "Lower middle income"  
#> [147] "Luxembourg"  
#> [148] "Macao SAR, China"  
#> [149] "Madagascar"  
#> [150] "Malawi"  
#> [151] "Malaysia"  
#> [152] "Maldives"  
#> [153] "Mali"  
#> [154] "Malta"  
#> [155] "Marshall Islands"  
#> [156] "Mauritania"  
#> [157] "Mauritius"  
#> [158] "Mexico"  
#> [159] "Micronesia, Fed. Sts."  
#> [160] "Middle East & North Africa"  
#> [161] "Middle East & North Africa (excluding high income)"  
#> [162] "Middle East & North Africa (IDA & IBRD countries)"  
#> [163] "Middle income"  
#> [164] "Moldova"  
#> [165] "Monaco"  
#> [166] "Mongolia"  
#> [167] "Montenegro"  
#> [168] "Morocco"
```

```
#> [169] "Mozambique"  
#> [170] "Myanmar"  
#> [171] "Namibia"  
#> [172] "Nauru"  
#> [173] "Nepal"  
#> [174] "Netherlands"  
#> [175] "New Caledonia"  
#> [176] "New Zealand"  
#> [177] "Nicaragua"  
#> [178] "Niger"  
#> [179] "Nigeria"  
#> [180] "North America"  
#> [181] "North Macedonia"  
#> [182] "Northern Mariana Islands"  
#> [183] "Norway"  
#> [184] "Not classified"  
#> [185] "OECD members"  
#> [186] "Oman"  
#> [187] "Other small states"  
#> [188] "Pacific island small states"  
#> [189] "Pakistan"  
#> [190] "Palau"  
#> [191] "Panama"  
#> [192] "Papua New Guinea"  
#> [193] "Paraguay"  
#> [194] "Peru"  
#> [195] "Philippines"  
#> [196] "Poland"  
#> [197] "Portugal"  
#> [198] "Post-demographic dividend"  
#> [199] "Pre-demographic dividend"  
#> [200] "Puerto Rico"  
#> [201] "Qatar"  
#> [202] "Romania"  
#> [203] "Russian Federation"  
#> [204] "Rwanda"  
#> [205] "Samoa"  
#> [206] "San Marino"  
#> [207] "Sao Tome and Principe"  
#> [208] "Saudi Arabia"  
#> [209] "Senegal"
```



```
#> [210] "Serbia"  
#> [211] "Seychelles"  
#> [212] "Sierra Leone"  
#> [213] "Singapore"  
#> [214] "Sint Maarten (Dutch part)"  
#> [215] "Slovak Republic"  
#> [216] "Slovenia"  
#> [217] "Small states"  
#> [218] "Solomon Islands"  
#> [219] "Somalia"  
#> [220] "South Africa"  
#> [221] "South Asia"  
#> [222] "South Asia (IDA & IBRD)"  
#> [223] "South Sudan"  
#> [224] "Spain"  
#> [225] "Sri Lanka"  
#> [226] "St. Kitts and Nevis"  
#> [227] "St. Lucia"  
#> [228] "St. Martin (French part)"  
#> [229] "St. Vincent and the Grenadines"  
#> [230] "Sub-Saharan Africa"  
#> [231] "Sub-Saharan Africa (excluding high income)"  
#> [232] "Sub-Saharan Africa (IDA & IBRD countries)"  
#> [233] "Sudan"  
#> [234] "Suriname"  
#> [235] "Sweden"  
#> [236] "Switzerland"  
#> [237] "Syrian Arab Republic"  
#> [238] "Tajikistan"  
#> [239] "Tanzania"  
#> [240] "Thailand"  
#> [241] "Timor-Leste"  
#> [242] "Togo"  
#> [243] "Tonga"  
#> [244] "Trinidad and Tobago"  
#> [245] "Tunisia"  
#> [246] "Turkiye"  
#> [247] "Turkmenistan"  
#> [248] "Turks and Caicos Islands"  
#> [249] "Tuvalu"  
#> [250] "Uganda"
```

```
#> [251] "Ukraine"
#> [252] "United Arab Emirates"
#> [253] "United Kingdom"
#> [254] "United States"
#> [255] "Upper middle income"
#> [256] "Uruguay"
#> [257] "Uzbekistan"
#> [258] "Vanuatu"
#> [259] "Venezuela, RB"
#> [260] "Vietnam"
#> [261] "Virgin Islands (U.S.)"
#> [262] "West Bank and Gaza"
#> [263] "World"
#> [264] "Yemen, Rep."
#> [265] "Zambia"
#> [266] "Zimbabwe"
```

今回は下のように、`|>`（パイプと呼びます）で繋げてコードを書きました。

```
df_gdp |> distinct(country) |> pull()
```

最初は、データ、その中の、異なる国を選択して、書き出してくださいというものです。

これは、

```
pull(distinct(df_gdp, country))
```

と同じです。どんどん、かっこの中に入れ子になって複雑になるので、一つ一つのステップを、順に書いたものが、最初のものになります。

```
df_gdp |> head()
```

```
df_gdp |> str()
```

なども可能です。かっこの中に最初に入るものが直前のもの、ここでは、データになっています。

3.2.4 必要に応じて整形 Transform data

変数が多いので、日本の部分だけ `filter` を使って選択します。 `country` が `Japan` と一致する場合のみを選択するときは、`==` を使います。数値ではないので、引用符をつけます。半角を使ってください。

```
df_gdp |> filter(country == "Japan")
#> # A tibble: 63 x 13
#>   country iso2c iso3c year      gdp status lastupdated
#>   <chr>    <chr> <chr> <dbl> <dbl> <lgl> <date>
```

```
#> 1 Japan JP JPN 2022 4.23e12 NA 2023-07-25
#> 2 Japan JP JPN 2021 5.01e12 NA 2023-07-25
#> 3 Japan JP JPN 2020 5.05e12 NA 2023-07-25
#> 4 Japan JP JPN 2019 5.12e12 NA 2023-07-25
#> 5 Japan JP JPN 2018 5.04e12 NA 2023-07-25
#> 6 Japan JP JPN 2017 4.93e12 NA 2023-07-25
#> 7 Japan JP JPN 2016 5.00e12 NA 2023-07-25
#> 8 Japan JP JPN 2015 4.44e12 NA 2023-07-25
#> 9 Japan JP JPN 2014 4.90e12 NA 2023-07-25
#> 10 Japan JP JPN 2013 5.21e12 NA 2023-07-25
#> # i 53 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> # longitude <dbl>, latitude <dbl>, income <chr>,
#> # lending <chr>
```

```
df_gdp |> filter(country == "Japan") |> head(2)
#> # A tibble: 2 x 13
#>   country iso2c iso3c year   gdp status lastupdated
#>   <chr>   <chr> <chr> <dbl> <dbl> <lgl> <date>
#> 1 Japan   JP    JPN  2022 4.23e12 NA 2023-07-25
#> 2 Japan   JP    JPN  2021 5.01e12 NA 2023-07-25
#> # i 6 more variables: region <chr>, capital <chr>,
#> # longitude <dbl>, latitude <dbl>, income <chr>,
#> # lending <chr>
```

2行目の、gdp の、 4.940878×10^{12} （この文書では、幅の都合で、 4.9×10^{12} と表示されているかもしれませんが）は、Scientific notation とされるもので、

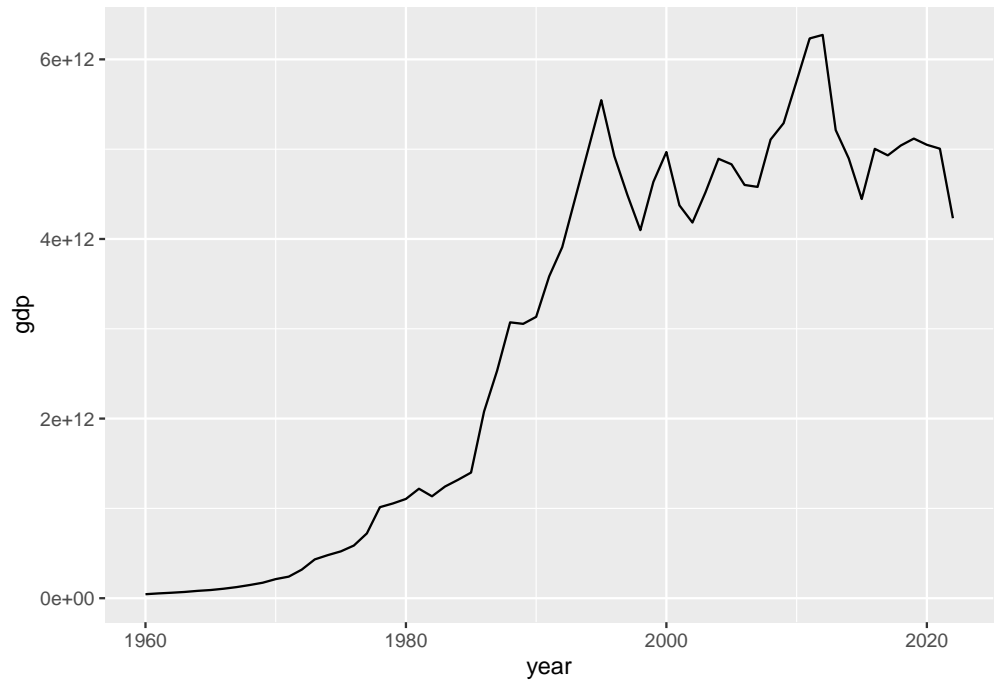
$$4.940878 \times 10^{12} = 4,940,887,800,000$$

を意味します。e+3 は千 (thousand)、e+6 は百万 (million)、e+9 は、10 億 (billion)、e+12 は、兆 (trillion) ですから、日本の、2021 年の GDP は、約 5 兆ドルとなります。

3.2.5 視覚化 data visualization

3.2.5.1 Fig 1. 日本の GDP の経年変化を表す折線グラフ (line graph)

```
df_gdp |> filter(country == "Japan") |>
  ggplot(aes(x = year, y = gdp)) + geom_line()
```



```
df_gdp |> filter(country == "Japan") |>
  ggplot(aes(x = year, y = gdp)) + geom_line()
```

日本を選択したときに、それに名前をつけた新しいオブジェクトを作り、それを使うこともできますが、名前がどんどん増えるので、パイプを使って、コードを続けて書いていく方法をとっています。

```
ggplot(aes(x = year, y = gdp)) + geom_line()
```

の部分が、グラフを描く部分で、「x 軸を、year、y 軸を、gdp として、それを、折線グラフで描いてください」というコードです。

Warning: [38;5;238mRemoved 1 row containing missing values

などと表示される場合がありますが、それは、値がない (missing または値が NA, not available) 年があることを言っています。データがない年を最初から削除してこくことも可能です。

3.2.6 データの理解 Understand data

上の折線グラフを使った、視覚化によって見えてくるものがいくつもありますね。どんなことがわかりますか。気づいたこと (observation) をあげてみましょう。

コードを描くことではなく、この部分が、データサイエンスの核の部分です。気づいたこと、疑問点を列挙してみましょう。

急激に増加しているとき、増加減少が繰り返している時、全体としては、1995 年ごろからはあまり、増加していないように見えることなどがわかりますね。それぞれのピークや、

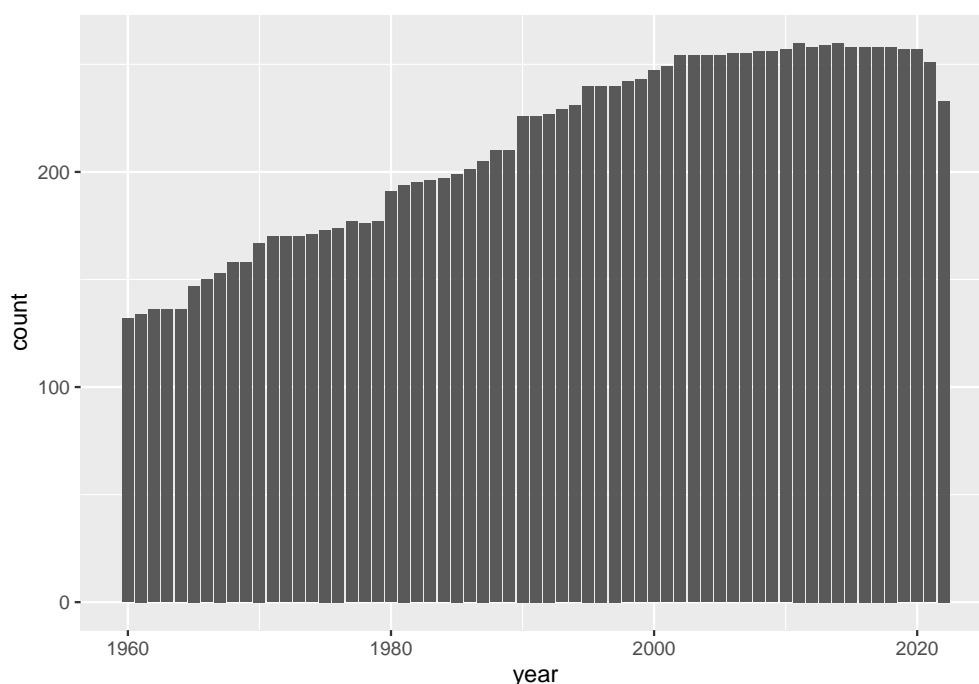
下落は、なにがあったのかも気になりませんか。これは、世界的な傾向なのでしょうか。日本だけでしょうか。他にも似た傾向の国があるのでしょうか。将来はどうなっていくのでしょうか。などなど。

3.2.7 さまざまな視覚化

3.2.7.1 Fig 2. 各年ごとのデータの数

`summary(df_gdp)` で、データ自体は、1960年から2022年までのようですが、年によって、どの程度データがあるか、調べてみます。

```
df_gdp |> drop_na(gdp) |> ggplot(aes(x = year)) + geom_bar()
```



```
df_gdp |> drop_na(gdp) |> ggplot(aes(x = year)) + geom_bar()
```

棒グラフ (bar graph) を使います。gdp の値が、欠損値 (NA: not available) のデータを削除してから、グラフを描きます。

3.2.7.2 2022年のGDPの降順での表示 (1)

最新の2022年のデータはすべてあるわけではなさそうですが、gdpの値が大きい順に並べてみましょう。

```
df_gdp |> filter(year == 2022) |> drop_na(gdp) |> arrange(desc(gdp))
#> # A tibble: 233 x 13
#>   country      iso2c iso3c year      gdp status lastupdated
#>   <chr>        <chr> <chr> <dbl>  <dbl> <lgl> <date>
#> 1 World       1W    WLD   2022  1.01e14 NA    2023-07-25
#> 2 High income XD    <NA>  2022  6.15e13 NA    2023-07-25
```

```
#> 3 OECD members OE OED 2022 5.96e13 NA 2023-07-25
#> 4 Post-demogr~ V4 PST 2022 5.60e13 NA 2023-07-25
#> 5 IDA & IBRD ~ ZT IBT 2022 4.04e13 NA 2023-07-25
#> 6 Low & middl~ XO LMY 2022 3.87e13 NA 2023-07-25
#> 7 Middle inco~ XP MIC 2022 3.82e13 NA 2023-07-25
#> 8 IBRD only XF IBD 2022 3.76e13 NA 2023-07-25
#> 9 East Asia &~ Z4 EAS 2022 3.07e13 NA 2023-07-25
#> 10 Upper middl~ XT <NA> 2022 3.01e13 NA 2023-07-25
#> # i 223 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> # longitude <dbl>, latitude <dbl>, income <chr>,
#> # lending <chr>
```

3.2.7.3 2021年のGDPの降順での表示(2)

最初に、Worldと表示され、グループや、カテゴリーのデータもあるようですから、それを、まず、削除する必要があります。regionの列を見ると、Worldなどは、Aggregatesとなっているので、そのようなものを削除すればよさそうです。数値の大きい順に並べたいので、desc降順(descending order)にします。

```
df_gdp |> filter(year == 2022, region != "Aggregates") |>
  drop_na(gdp) |> arrange(desc(gdp))
#> # A tibble: 184 x 13
#>   country      iso2c iso3c year      gdp status lastupdated
#>   <chr>        <chr> <chr> <dbl>   <dbl> <lgl> <date>
#> 1 United Stat~ US     USA  2022 2.55e13 NA 2023-07-25
#> 2 China       CN     CHN  2022 1.80e13 NA 2023-07-25
#> 3 Japan       JP     JPN  2022 4.23e12 NA 2023-07-25
#> 4 Germany     DE     DEU  2022 4.07e12 NA 2023-07-25
#> 5 India       IN     IND  2022 3.39e12 NA 2023-07-25
#> 6 United King~ GB     GBR  2022 3.07e12 NA 2023-07-25
#> 7 France      FR     FRA  2022 2.78e12 NA 2023-07-25
#> 8 Russian Fed~ RU     RUS  2022 2.24e12 NA 2023-07-25
#> 9 Canada      CA     CAN  2022 2.14e12 NA 2023-07-25
#> 10 Italy       IT     ITA  2022 2.01e12 NA 2023-07-25
#> # i 174 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> # longitude <dbl>, latitude <dbl>, income <chr>,
#> # lending <chr>
```

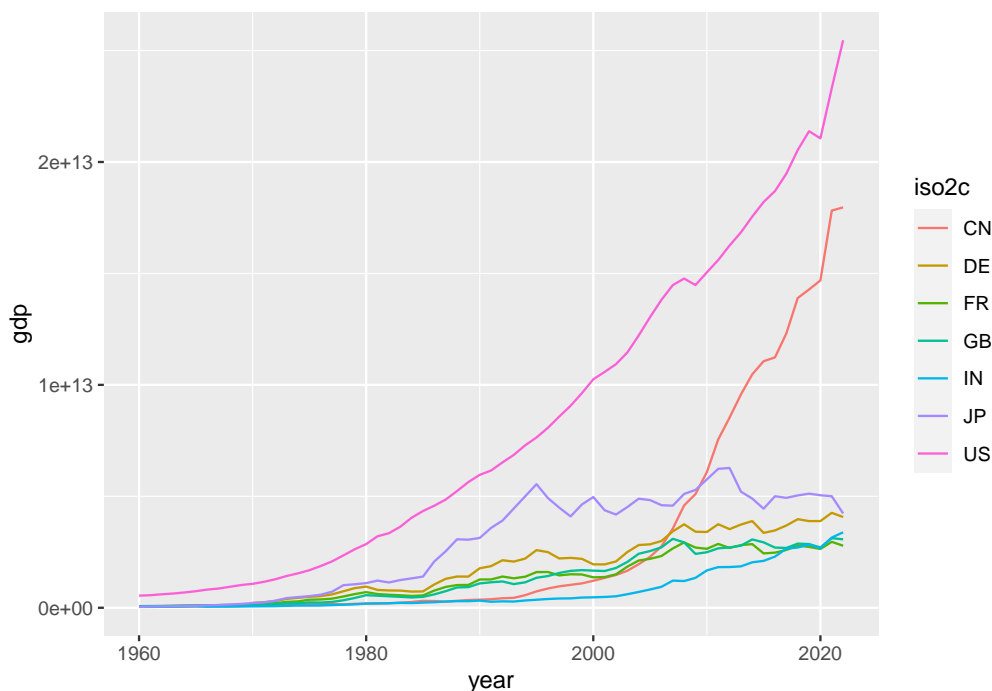
これは、グラフではありませんが、これも一つの可視化と考えられないことはありません。

上位7カ国は、United States, China, Japan, Germany, India, United Kingdom, France

であることがわかりました。このあと、Russian Federation, Canada, Italy と続き、でこまでが、2022年のGDPが2兆ドルを越している国となります。

3.2.7.4 Fig 3. 2022年時のGDP上位7カ国のGDP経年変化

```
df_gdp |> filter(iso2c %in% c("US", "CN", "JP", "DE", "IN", "GB", "FR")) |>
  ggplot(aes(x = year, y = gdp, col = iso2c)) + geom_line()
#> Warning: Removed 10 rows containing missing values
#> (`geom_line()`).
```



```
df_gdp |> filter(iso2c %in% c("US", "CN", "JP", "DE", "IN", "GB", "FR")) |>
  ggplot(aes(x = year, y = gdp, col = iso2c)) + geom_line()
```

ここでは、最初に、`filter` を使って、7カ国のデータを選択しています。そのときには、`%in%` として、国名を、combine するという意味で、`c()` とひとまとめにします。数字ではなく、文字なので、引用符で囲んでいます。この場合は、single quote でも構いませんが、半角を使ってください。

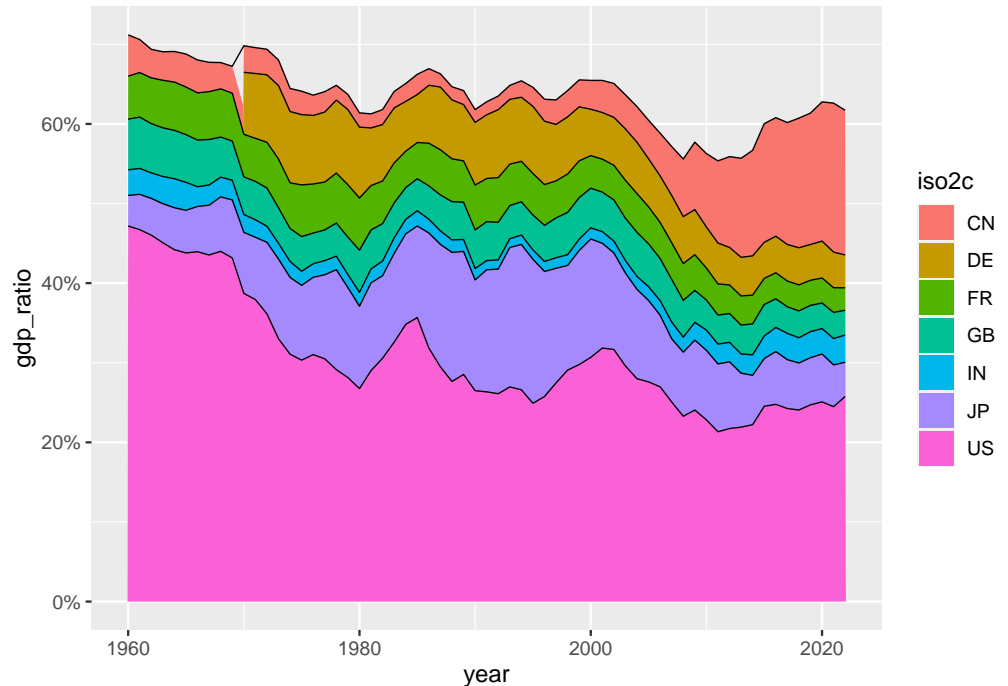
このグラフからは、どのようなことがわかりますか。気づいたことを書いてみましょう。

もう少し、このようなグラフをみてみたいというような、メモも大切です。

3.2.7.5 Fig 4. 世界のGDPにおける割合 (1)

```
df_gdp |>
  filter(region != "Aggregates") |> drop_na(gdp) |>
  group_by(year) |> mutate(gdp_ratio = gdp/sum(gdp)) |> ungroup() |>
```

```
filter(iso2c %in% c("US", "CN", "JP", "DE", "IN", "GB", "FR")) |>
ggplot(aes(x = year, y = gdp_ratio, fill = iso2c)) + geom_area() +
geom_line(col = "black", position = "stack", linewidth = 0.3) +
scale_y_continuous(labels = scales::percent_format(accuracy = 1))
```



まず、下の部分が新しいですが、ここでは、年毎にグループにして、その上で、新しい `gdp_ratio` という名前の列を追加し、その `gdp` の値を、`gdp` 合計で割っています。すなわち、世界の、GDP における割合が計算されています。

```
group_by(year) |> mutate(gdp_ratio = gdp/sum(gdp)) |> ungroup() |>
```

下の部分では、`geom_area` を使って、`fill=iso2c` により、`iso2c` ごとに、違う色を塗って、`position = "stack"` により、積み上げ型の、グラフを描き、境目がわかりやすいように、0.3 の太さの黒の線を描いてください。また、`y` 軸は、小数点以下を省いたパーセント表示に変えてください。というコードです。

```
ggplot(aes(x = year, y = gdp_ratio, fill = iso2c)) + geom_area() +
geom_line(col = "black", position = "stack", linewidth = 0.3) +
scale_y_continuous(labels = scales::percent_format(accuracy = 1))
```

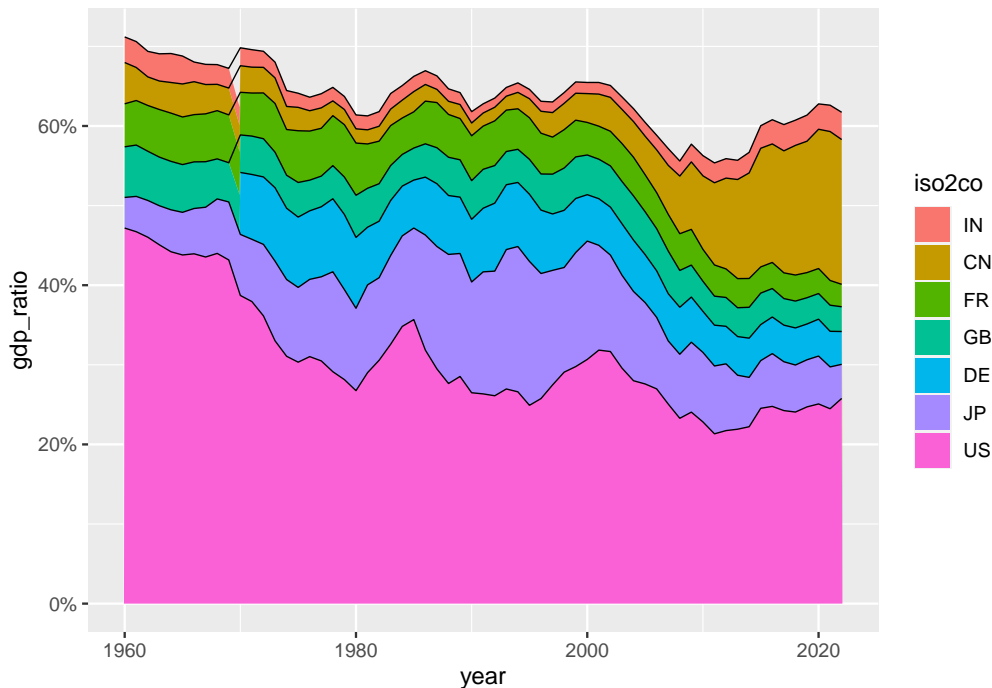
3.2.7.6 Fig 4. 世界の GDP における割合 (2)

これは、上から、`iso2c` のアルファベットの順番になっていますが、積み上げの順序を変更することもできます。

```
df_gdp |>
filter(region != "Aggregates") |> drop_na(gdp) |>
```



```
group_by(year) |> mutate(gdp_ratio = gdp/sum(gdp)) |> ungroup() |>
filter(iso2c %in% c("US", "CN", "JP", "DE", "IN", "GB", "FR")) |>
mutate(iso2co = factor(iso2c, levels = c("IN", "CN", "FR", "GB", "DE", "JP", "US"))) |>
ggplot(aes(x = year, y = gdp_ratio, fill = iso2co)) + geom_area() +
geom_line(col = "black", position = "stack", linewidth = 0.3) +
scale_y_continuous(labels = scales::percent_format(accuracy = 1))
```



これらは、世界全体の GPT における割合です。

どのようなことがわかりますか。

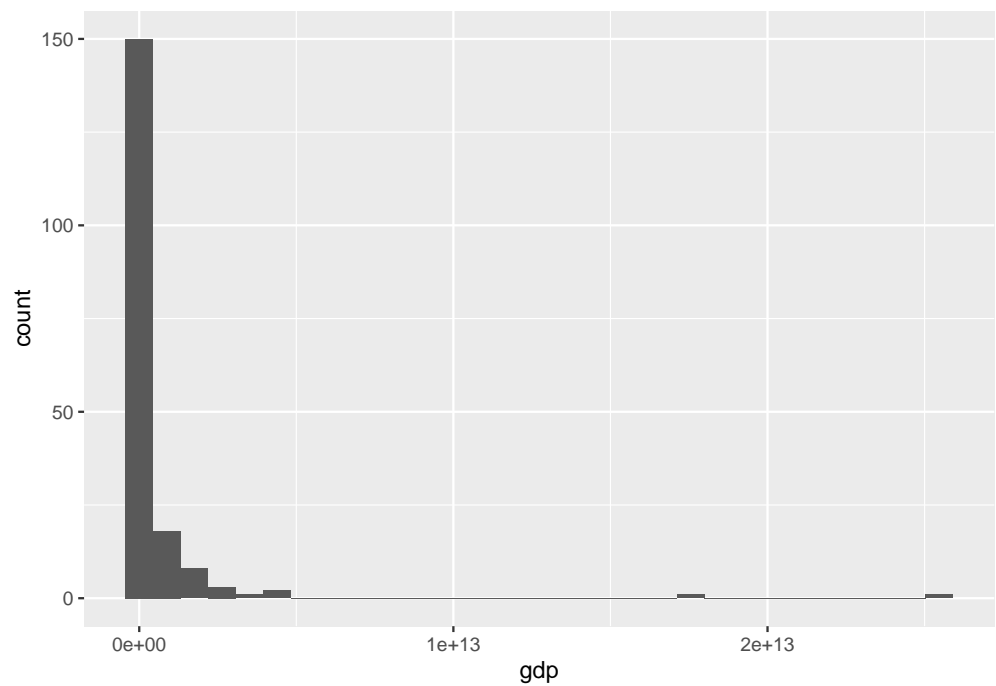
主要国で、60%~70% を占めていることがわかります。それぞれの国や、幾つかの国の影響力や、その時代による変化も、ある程度みることができるようになります。

気づいたこと、疑問に思ったことなどを、書き出してみてください。

GDP が大きな国と、小さな国があるのはわかりますが、それは、どのように分布しているのでしょうか。

3.2.7.7 Fig 5. 2022 年の世界の国の GDP の分布 (1)

```
df_gdp |> drop_na(gdp) |>
  filter(year == 2022) |> filter(region != "Aggregates") |>
  ggplot(aes(gdp)) + geom_histogram()
#> `stat_bin()` using `bins = 30`. Pick better value with
#> `binwidth`.
```

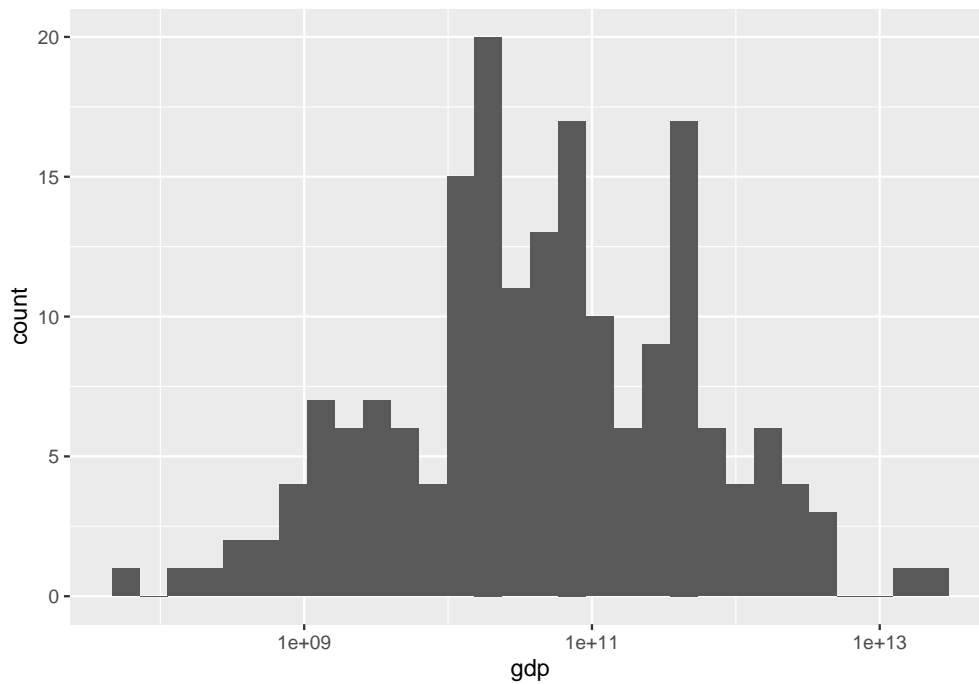


小さいところに集中していることがわかりますが、`scale_x_log10()` を加え、対数軸をとってみます。

$\log_{10}(1000) = 3$, $\log_{10}(1000000) = 6$, $\log_{10}(1000000000) = 9$ などになります。

3.2.7.8 Fig 6. 2022 年の世界の国の GDP の分布 (2)

```
df_gdp |> drop_na(gdp) |>
  filter(year == 2022) |> filter(region != "Aggregates") |>
  ggplot(aes(gdp)) + geom_histogram() + scale_x_log10()
#> `stat_bin()` using `bins = 30`. Pick better value with
#> `binwidth`.
```

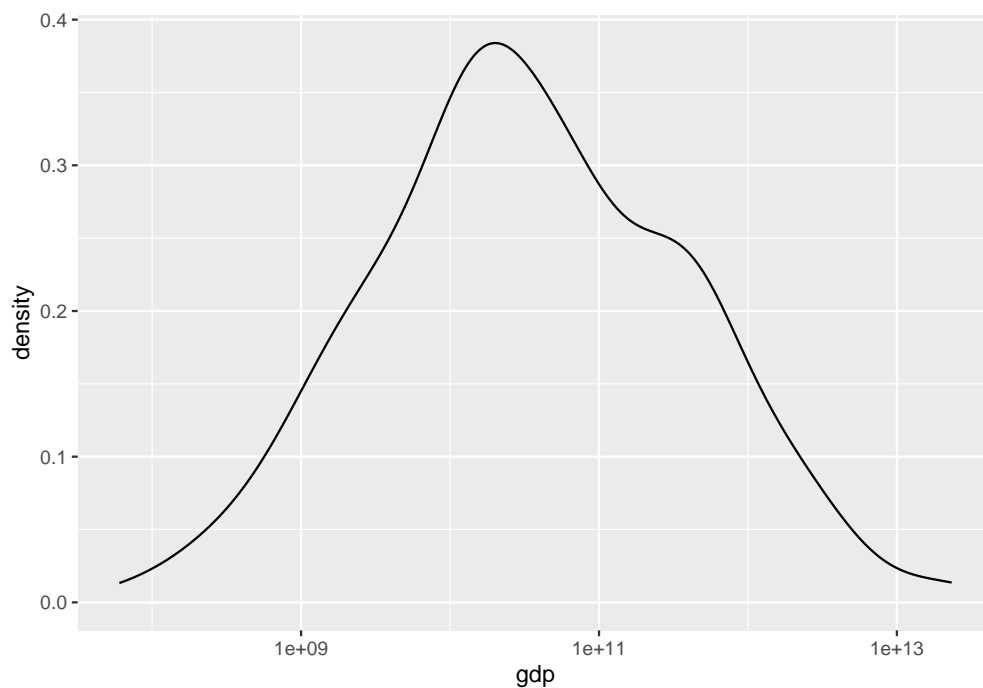


幅を変更したり、分ける個数を変更するには `binwidth = 0.5` や、`bins = 20` を、`geom_histogram()` のかっこの中に入れます。

3.2.7.9 Fig 7. 2021 年の世界の国の GDP の分布 (3)

また、密度曲線に変えるには、`geom_density` を使います。

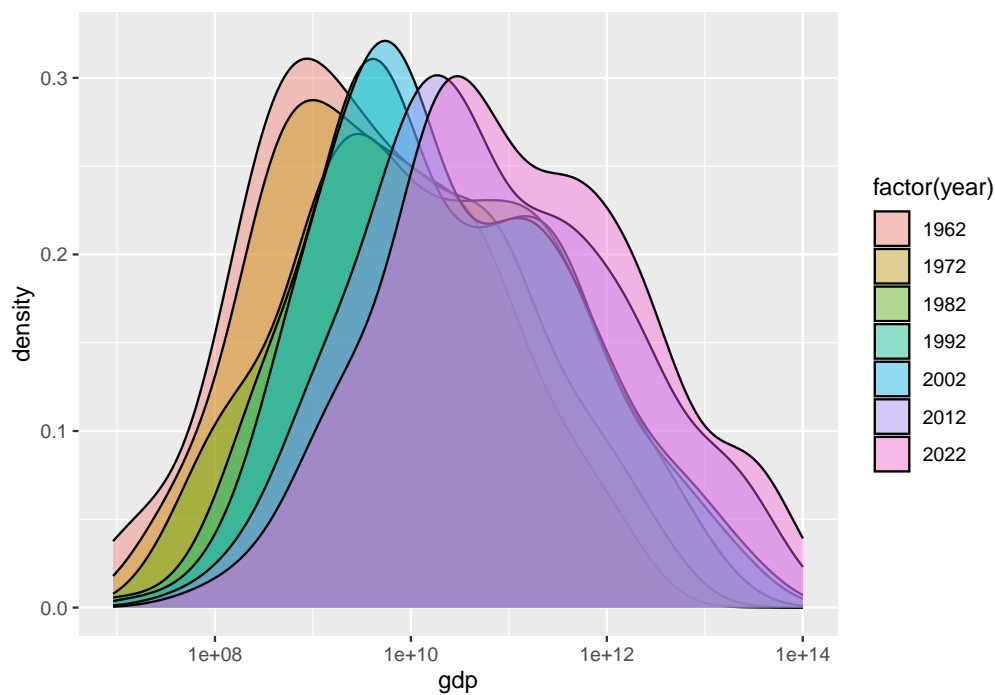
```
df_gdp |> drop_na(gdp) |>  
  filter(year == 2021) |> filter(region != "Aggregates") |>  
  ggplot(aes(gdp)) + geom_density() + scale_x_log10()
```



3.2.7.10 Fig 8. 2022年までの世界の国のGDPの分布（4）

これは、2022年のデータですが、10年ごとの、densityの変化をしてみます。alphaの値は透明度です。

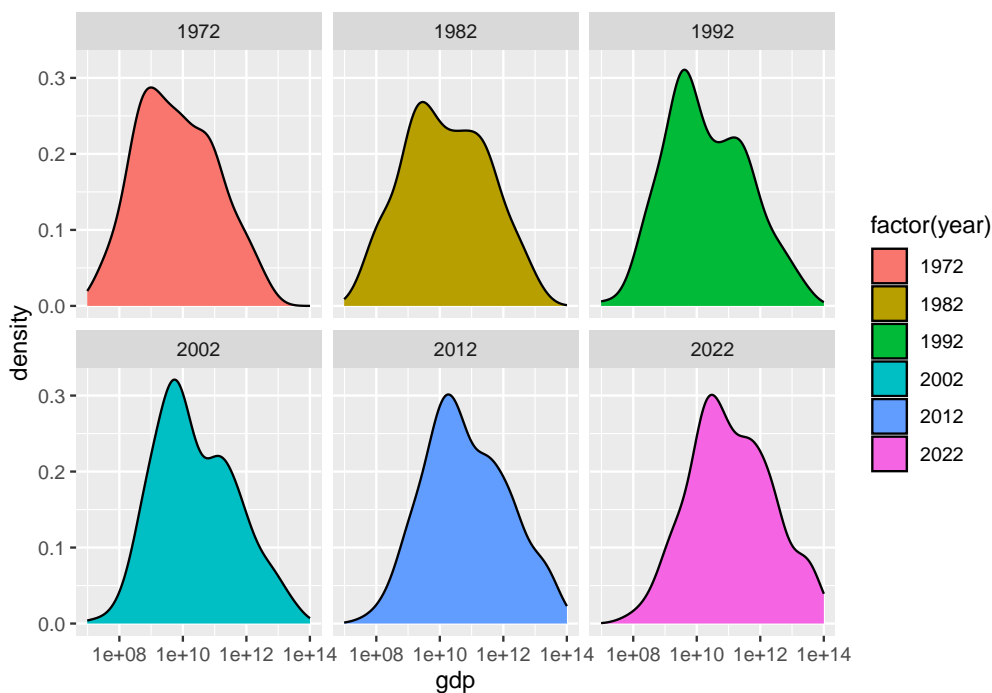
```
df_gdp |> drop_na(gdp) |>  
  filter(year %in% c(1962, 1972, 1982, 1992, 2002, 2012, 2022)) |>  
  ggplot(aes(gdp, fill = factor(year))) + geom_density(alpha = 0.4) + scale_x_log
```



3.2.7.11 Fig 9. 2022年の世界の国のGDPの分布 (5)

少しみにくいので、分けてみます。

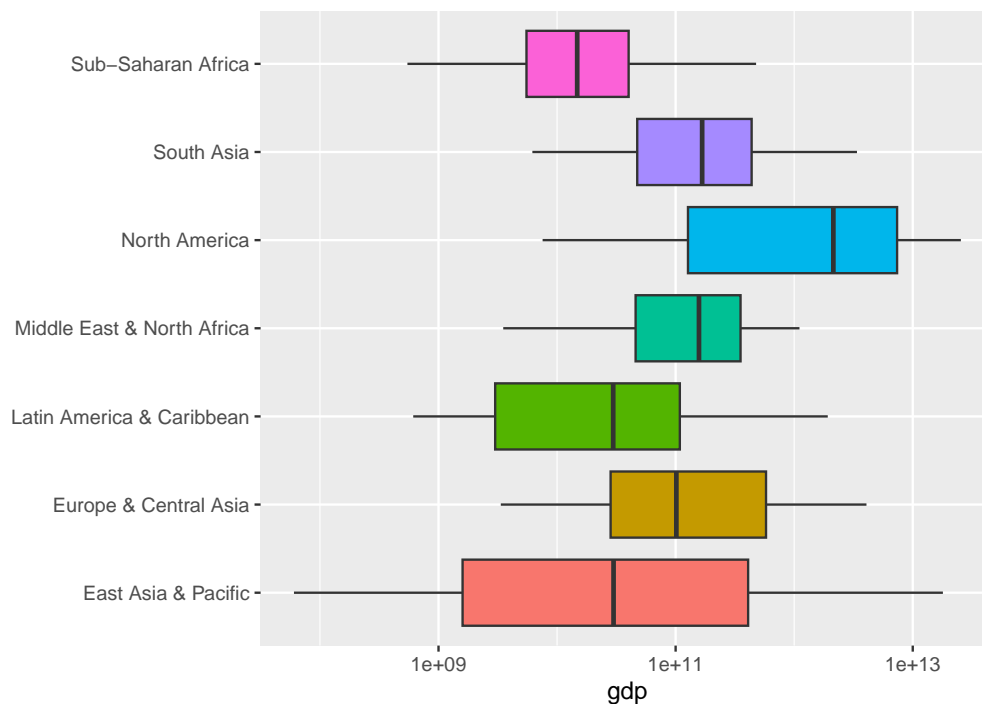
```
df_gdp |> drop_na(gdp) |>
  filter(year %in% c(1972, 1982, 1992, 2002, 2012, 2022)) |>
  ggplot(aes(gdp, fill = factor(year))) +
  geom_density() + scale_x_log10() + facet_wrap(~year)
```



3.2.7.12 Fig 10. 地域ごとのGDPの分布

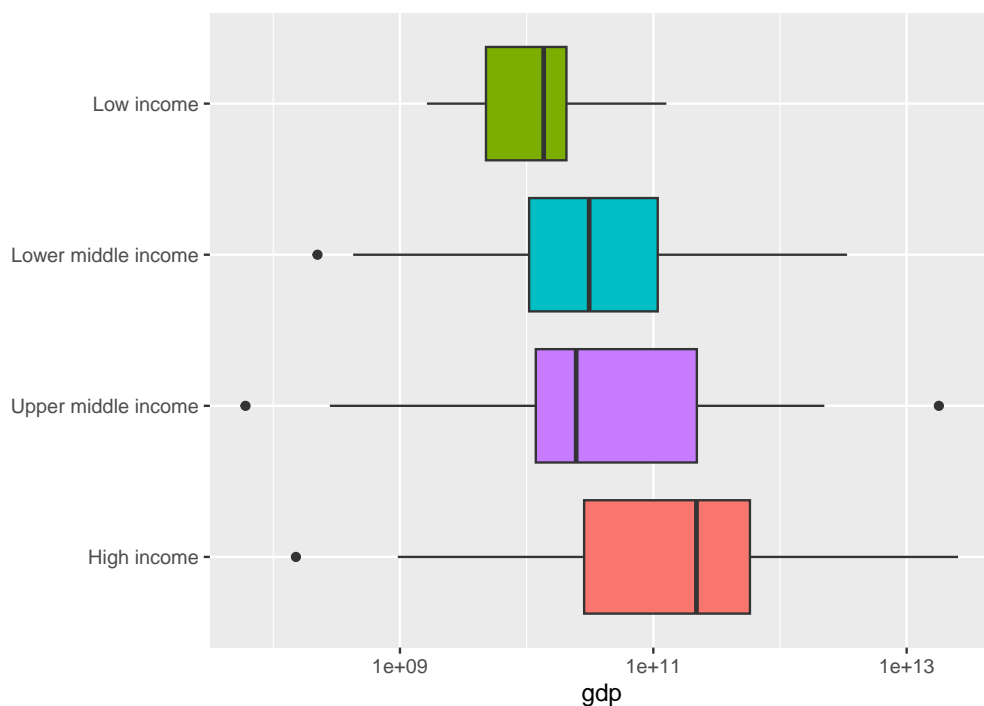
いくつかのグループごとに分布をみても可能ですが、それには、箱ひげ図 (Boxplot) が有効です。箱ひげ図では、そのグループの国を値の大きさの順にならべて、四分分し、その、真ん中の二つが箱の部分に対応しています。また、真ん中の太い線は、中央値 (median) を表しています。詳しくは、後ほど説明します。

```
df_gdp |> drop_na(gdp) |> filter(region != "Aggregates") |>
  drop_na(region) |> filter(year %in% c(2022)) |>
  ggplot(aes(gdp, region, fill = region)) +
  geom_boxplot() + scale_x_log10() + labs(y = "") +
  theme(legend.position = "none")
```



3.2.7.13 Fig 11. 収入の多寡による分類ごとの GDP 分布

```
df_gdp |> drop_na(gdp) |> filter(region != "Aggregates") |>
  drop_na(income) |> filter(year %in% c(2022)) |>
  mutate(level = factor(income, c("High income", "Upper middle income", "Lower mi
ggplot(aes(gdp, level, fill = income)) +
  geom_boxplot() + scale_x_log10() + labs(y = "") +
  theme(legend.position = "none")
```



これからも、いろいろなことがわかりますね。点は、外れ値を表しています。外れ値についても、きちんと決まっています。収入の多寡（Income Level）のグループは、GNI per Capita という、一人当たりの国民総所得（GNI を人口で割ったもの）をもとに、世界銀行が決めているものです。

3.2.7.14 世界地図の準備

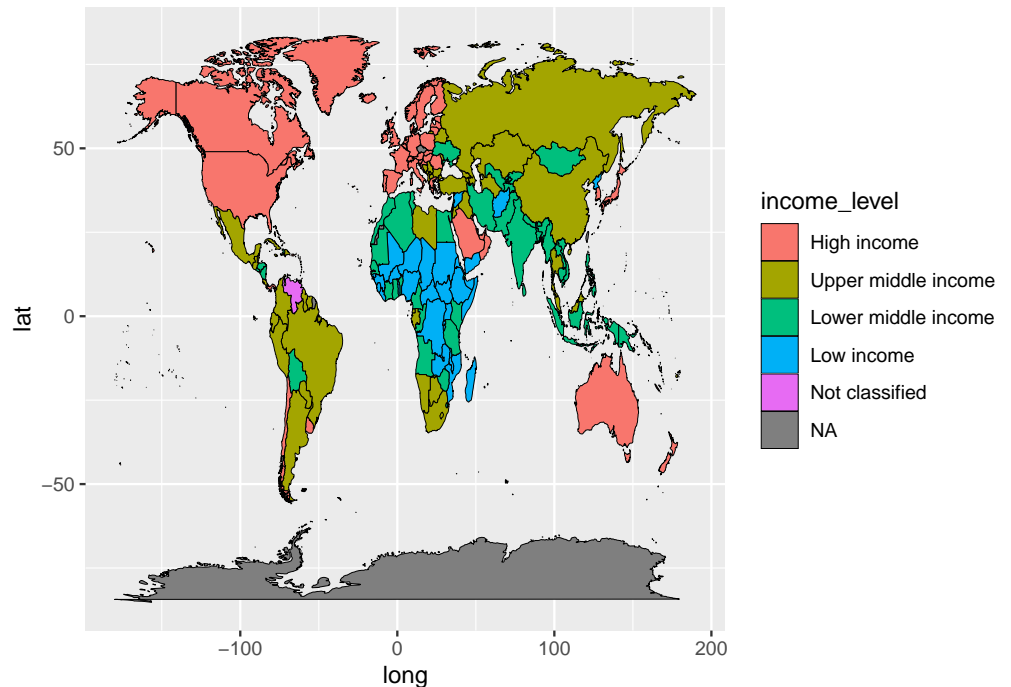
地図で、国の収入の多寡（income level）をみてみましょう。

```
library(maps)
gdp_short <- df_gdp |> filter(year == 2022, region != "Aggregates") |>
  select(iso2c, gdp, income)
map_world <- map_data('world')
map_gdp <- map_world |>
  mutate(iso2c = iso.alpha(region, n=2)) |>
  left_join(gdp_short, by = "iso2c")
head(map_gdp)
#>   long    lat group order region subregion iso2c gdp
#> 1 -69.89912 12.45200   1     1  Aruba    <NA>   AW  NA
#> 2 -69.89571 12.42300   1     2  Aruba    <NA>   AW  NA
#> 3 -69.94219 12.43853   1     3  Aruba    <NA>   AW  NA
#> 4 -70.00415 12.50049   1     4  Aruba    <NA>   AW  NA
#> 5 -70.06612 12.54697   1     5  Aruba    <NA>   AW  NA
#> 6 -70.05088 12.59707   1     6  Aruba    <NA>   AW  NA
#>   income
```

```
#> 1 High income
#> 2 High income
#> 3 High income
#> 4 High income
#> 5 High income
#> 6 High income
```

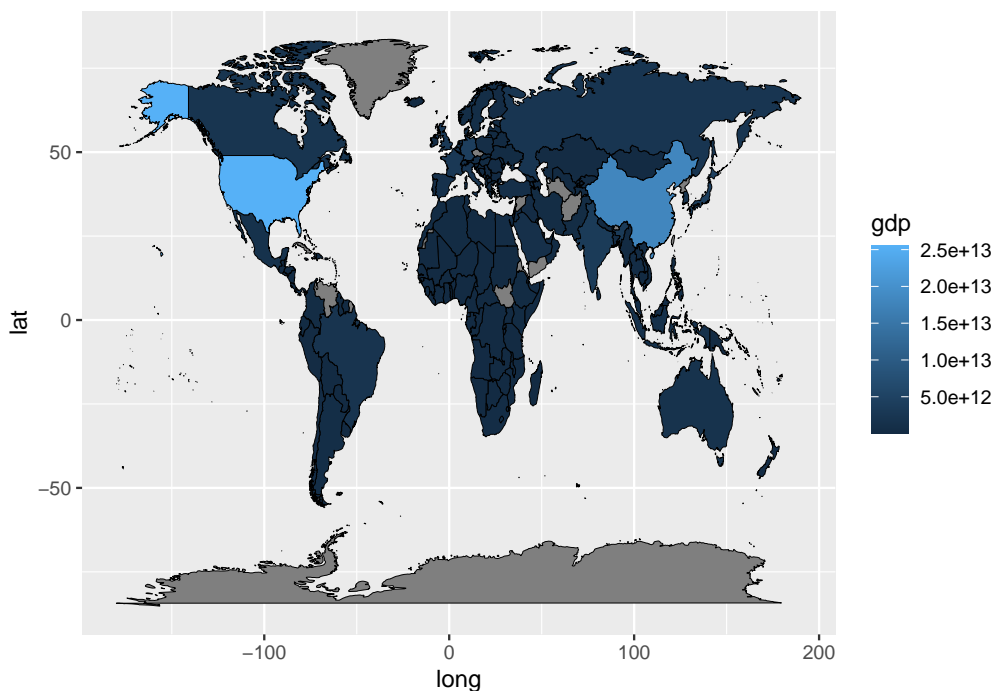
3.2.7.15 Fig 12. Income Level による色分け地図

```
map_gdp |> mutate(income_level = factor(income, levels = c("High income", "Upper
ggplot() +
  geom_map(aes(long, lat, map_id = region, fill = income_level), map = map_world,
#> Warning: Using `size` aesthetic for lines was deprecated in ggplot2
#> 3.4.0.
#> i Please use `linewidth` instead.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where
#> this warning was generated.
#> Warning in geom_map(aes(long, lat, map_id = region, fill =
#> income_level), : Ignoring unknown aesthetics: x and y
```



3.2.7.16 Fig 13. GDP による色分け地図


```
map_gdp |>
  ggplot() +
  geom_map(aes(x=long, y=lat, map_id = region, fill = gdp), map = map_world, col = "black", size = 0.1)
#> Warning in geom_map(aes(x = long, y = lat, map_id = region,
#> fill = gdp), : Ignoring unknown aesthetics: x and y
```



3.3 練習

1. それぞれのグラフから、わかったこと、問いなどを列挙してみましょう。
2. Fig 1 の Japan の部分を他の国や、グループ (World など) に変えてみてください。何がわかりますか。
3. Fig 3 の iso2c で選択する、国を変更してください。何がわかりますか。
4. Fig 4 または Fig 5 の iso2c の部分を他の国に変更してください。何がわかりますか。
5. Fig 5 または Fig 6 の、区間の幅や、数を変更してみてください。何がわかりますか。
6. Fig 7, Fig 8, Fig 9 の年を変更してみてください。何がわかりますか。
7. Fig 12, Fig 13 の年を変更してみてください。何がわかりますか。

3.4 プロジェクト

他のデータで、同様のことをしてみましょう。

1. 最初に、`gdp = "NY.GDP.MKTP.CD"` としましたが、GNI per capita, Atlas method

(current US\$): NY.GNP.PCAP.CD に変更してみましょう。

```
df_gnppcap <- WDI(country = "all",
  indicator = c(gnppcap = "NY.GNP.PCAP.CD"),
  extra = TRUE)
```

2. World Development Indicators のサイトの下にある、Data Themes (テーマ) から自分が調べたいテーマを選び、そのテーマから、データコードを取得して、同様の分析を試みてください。データがあまりない場合もありますので、ある程度データが多いものを選択することをお勧めします。

3.5 まとめ

今回は、経済指標である、GDP を使いました。データサイエンスについて少しずつ、学んでいきます。

コードの説明は、簡単にしかしていませんから、理解するのは難しいと思いますが、いくつかのことは、ご理解いただけたと思います。

- それほど、長くはない、コードで、データを見ていく。R は対話型 (interactive) のプログラミング言語とされています。
- コードに続けて、結果が表示されるので、コードと出力の対応が見やすい。また、コメントや説明も併記することができる。これは、RMarkdown という形式の中で、コードを書いていることによるものです。RMarkdown は、再現性 (reproducibility) と、プログラム・コードの内容をコンピュータにわかるようにでなく、人間にもわかるように記述する (Literate Programming) を実現しています。
- 視覚化 (Visualization) によって、わかることが色々ある。また、視覚化の方法もたくさんあり、いろいろな見方をすることで、データについての理解が深まっていく。
- 視覚化を通して、データを理解すること、問いを持ち、他の視覚化などを用いて、さらに、理解を深めることがたいせつ。
- 理解したことを元にして、さらに、そのデータ、または、他のデータを使って、新たな発見をしていく。

統計的な指標も用いますが、それらによって、新しい知識を生み出すとも表現しますが、そのような営み全体が、データサイエンスの核をなす部分だと思えます。

第 4 章

世界開発指標とオープンデータ

最初の例として紹介した、はじめてのデータサイエンスでは、世界銀行（World Bank）の世界開発指標（WDI）のひとつである、GDP（Gross Domestic Product 国内総生産）のデータを使いました。「データサイエンスをはじめましょう」では、世界開発指標をこれからも、頻繁に活用しますので、世界開発指標が、どのようなものか確認しておきましょう。

また、世界開発指標に代表されるオープンデータによって誰でも世界の課題を知り、向き合うことができるようになってきました。そのオープンデータは、どのようなものなのかも、見ていきたいと思います。

4.1 世界開発指標（WDI）

世界開発指標は、貧困撲滅と開発援助とに取り組んでいる金融機関である世界銀行が提供している、オープンデータの一つで、世界銀行は、他にも、たくさんのデータを提供しています。

世界開発指標は、英語では、World Development Indicators（WDI）と呼びますので、以下、省略形の WDI を使います。

4.1.1 概要

まず、簡単に概要を述べておきましょう。

World Development Indicators（WDI）は、世界銀行が開発に関する各国間比較可能なデータの集大成である 1400 の時系列指標（the World Bank's premier compilation of cross-country comparable data on development; 1400 time series indicators）からなっており、それらは、テーマ別にも分類されています。

また、さまざまな形式（CSV, Excel など）で、データを検索し、ダウンロードできるように整理されており、さらに、API（Application Program Interface：アプリケーションのプログラムを使ってデータを利用するための仕様）も整備されており、「データサイ

エンスをはじめましょう」で利用するコンピュータ言語 R でも、簡単に、検索やデータのダウンロードが可能なパッケージが提供されています。

4.1.2 三つの入り口

- 世界開発指標 (WDI) (World Development Indicators)
- 世界銀行オープンデータ (World Bank Open Data)
- 世界開発指標(WDI)の統計表 (World Development Indicators Statistical Tables)

三つのサイトのリンクを書きました。一つ一つみていきましょう。

まず、英語を読むことが苦手な場合は、ブラウザ (Google Chrome, Edge, Safari などのホームページ閲覧ソフト) の自動翻訳機能を使うのも良いでしょう。しかし、自動翻訳機能を利用していると、リンク先に飛べなかったり、いくつかの機能が使えないこともありますので、正常に機能しないときは、自動翻訳機能を一旦 OFF にして利用してください。

4.1.3 世界開発指標 (WDI)

まずは、このサイトがおすすめです。上の方に以下のように書かれています。

The World Development Indicators is a compilation of relevant, high-quality, and internationally comparable statistics about global development and the fight against poverty. The database contains 1,400 time series indicators for 217 economies and more than 40 country groups, with data for many indicators going back more than 50 years.

世界開発指標は、世界的な開発と、貧困との闘いに関連する、高品質で、国際的に比較可能な統計をまとめたものです。データベースには、217 の経済と 40 以上の国グループの 1,400 の時系列指標が含まれており、多くの指標のデータは 50 年以上前にさかのぼることができます。

この文章の下には、テーマ別のアイコンがあり、そこから、それぞれのテーマについてみていくことができるようになっています。さらにその下には、最近のニュースやお薦めの記事などが掲載されています。

4.1.3.1 テーマ (さまざまなトピック・課題)

1. 貧困と不平等 (Poverty and Inequality) : 貧困、繁栄、消費、所得分配
2. 人々 (People) : 人口動態、教育、労働、健康、ジェンダー
3. 環境 (Environment) : 農業、気候変動、エネルギー、生物多様性、水、衛生
4. 経済 (Economy) : 成長、経済構造、所得と貯蓄、貿易、労働生産性
5. 国家と市場 (States and Markets) : ビジネス、株式市場、軍事、通信、輸送、テクノロジー

6. グローバルリンク (Global Links) : 債務、貿易、援助への依存、難民、観光、移住

4.1.3.2 それぞれのテーマについて

一つテーマを選んで中身をみてみましょう。テーマを選んで、中身をみてみると、一番上に、最近のトピックなどが書かれ、次に、そのテーマについての概要や課題が書かれています。データの集め方や、どのように標準化するかなどについての説明もあります。

その下に、注目の指標 (Featured Indicators) として、主要な WDI の指標のリストがあり、さらにその下には、データについての説明などが書かれています。

4.1.3.3 注目の指標 (Featured Indicators)

この指標も、いくつかのグループに分けられていますが、それぞれの指標には、以下のような情報があります。

- Indicator 指標の名称
 - マウスを当てると、より詳しい概要をみることができます。
- Code 指標コード
 - R の WDI パッケージなど、API を利用して、データを取得するときに必要なコードです。
 - このコードをクリックすると、World (世界) に関する、この指標のデータをグラフが表示されます。世界に対応するデータがない場合もあります。
- Time coverage どの程度の期間のデータがあるかの概要
 - 1960 年から 10 年刻みでデータがどの程度あるかを確認することができます。
- Region coverage 地域ごとのデータの収集状況の概要
 - ラテンアメリカ、南アジア、サハラ砂漠以南のアフリカ、ヨーロッパと中央アジア、中東と北アフリカ、東アジアと太平洋諸国、北アメリカごとに、データがどの程度揃っているかを確認することができます。
- Get data データの取得
 - API、XLS、CSV、DataBank とあり、データを、取得するための情報や、Excel、コンマで区切られてテキストデータが取得可能で、また、DataBank では、World Bank のデータの、表や、グラフ、地図、各データの情報などを調べることができます。この DataBank の利用については、別の項で説明します。

4.1.3.4 備考

はじめてのデータサイエンスで利用した、GDP の指標は、テーマから経済 (Economy) を選択すると、Featured Indicator (注目の指標) の一番上にあります。まずは、GDP

について、上に書いた、Indicator、Code、Time Coverage、Region Coverage、GetDataを確認することをお勧めします。

このサイトにあるのは、基本的な指標だけです。すべての指標についての情報を得るときには、この次とその次の項目を見てください。しかし、全体では、1400 も指標があり、何をみたらよいか、かえって混乱してしまうかもしれません。まずは、このような基本的なデータで、使い方を把握してから、興味のあるデータを調べるのが良いと思います。

また、基本的な指標を調べると、その指標のページから関連した指標についても調べることができます。

WDI は、すべての年、すべての国についてのデータがあるわけではありません。前もって、期間や、地域ごとにどの程度データがあるかを見しておくことはとても有効です。

それぞれの指標についての概要や特定の国についての情報は、次の項目から見することも可能です。このページに掲載されている指標で興味をもったものについて、指標の名称や指標コードをコピー・ペーストなどで、記録しておくことをお勧めします。

4.1.4 世界銀行オープンデータ

世界銀行のオープンデータ全体にアクセスできるサイトです。

その下には、いくつかのグラフや、最近のニュースやトピックが書かれており、さらにその下には、More Resources (さらに...) といくつかの項目があります。Open Data Catalog、Data Bank、などと共に、World Development Indicators もあります。この World Development Indicators (WDI) を選択すると、最初に紹介したサイトに飛ぶことができます。また、Data Bank を選択すると、WDI の説明に登場した、ダッシュボードに飛ぶことができ、WDI だけでなく、他のデータについても、調べることができます。

上の検索窓の下に、Country (国) と Indicators (指標) と書いてあります。これらから、WDI について調べることができます。

4.1.4.1 国別サイト

Country (国別) のサイトを選択すると、国のリストが出てきます。

J のところに、Japan (日本) がありますから、選択してみてください。日本のさまざまな指標と小さなグラフが出てきます。

Indicator (指標)、Most Recent Values (直近の値)、Trend (傾向) が表示されます。指標をクリックすると、大きなグラフが出てきます。その指標のサイトですから、そこで、その指標についての他の国の状況などを確認することができます。そこでは、選択した指標と似た指標が選択できたり、もっと詳しいことをしらべるデータバンク (Data Bank) へのリンクもあります。このサイトから、データをダウンロードすることもできるようになっています。

また、国別サイトの右の方には、地域や、経済的状况などによって、グループに分けてある帯もあります。その一番下には、World (世界) もありますから、世界全体について見てみたり、6段階の収入 (GNI per Capita: 一人当たりの国民総生産) の階級に分けた階級ごとの値を見ることもできるようになっています。

4.1.4.2 指標別サイト

Indicators (指標) を選択すると、トピックに分けて、指標が並んでいます。

よく見ると、一番上に、Featured Indicators (特徴的な指標) と、All Indicators (すべての指標) とあり、最初に開いているのは、特徴的な指標の方であることがわかります。すべての指標 (All Indicators) の方を開けると、よりたくさんの指標を見ることができます。

トピックは、以下のものに分かれています。

- Agriculture & Rural Development 農業と農村開発
- Aid Effectiveness 援助の有効性
- Climate Change 気候変動
- Economy & Growth 経済と成長
- Education 教育
- Energy & Mining エネルギーと鉱業
- Environment 環境
- External Debt 対外債務
- Financial Sector 金融セクター
- Gender 性別
- Health 健康
- Infrastructure インフラ
- Poverty 貧困
- Private Sector 民間部門
- Public Sector 公共部門
- Science & Technology 科学技術
- Social Development 社会開発
- Social Protection & Labor 社会的保護と労働
- Trade 貿易
- Urban Development 都市開発

それぞれの指標を選択すると、グラフが表示される画面が出てきます。それは、上で国別のところから選択したものと同じです。

はじめてのデータサイエンスで、GDP を調べるときには、Economy and Growth のトピックにある、GDP (current US\$) の WDI コード NY.GDP.MKTP.CD を指定して、データを取得しました。この、Indicator Code (WDI コード) は、Details (詳細) を見ると、その指標の概要とともに、書いてあります。そのデータコードは、そのページの上の URL にも表示されています。

WDI という R のパッケージを使って、データを読み込みました。そのときに必要だったのが、この Indicator Code (WDI コード) でした。あとで、詳しく調べてみたい指標がありましたら、その指標名 (Indicator Name) と、WDI コード (Indicator Code) を、使えるように、メモなどに貼り付けて (Copy-Paste) おくことを、お勧めします。

4.1.5 世界開発指標 (WDI) の統計表

三つ目の入り口の説明をしましょう。英語名は、World Development Indicators Statistical Tables となっています。WDI の一番下の、Useful Resources (資料) の中にもありますし、それぞれの、テーマの中にもリンクがあります。

世界開発指標 (WDI) の統計表のリンクを開くと、WDI が7つのグループに分けられています。

1. World View 世界の姿
2. Poverty and Shared Prosperity 貧困と富の分配
3. People 人々
4. Environment 環境
5. Economy 経済
6. States and Market 国と市場
7. Global Links 世界の繋がり

WDI のテーマとほぼ一致しています。それぞれのテーマを選択すると、その説明とともに、さらに項目に分かれており、そのデータを見たり、データを PDF または Excel 形式で、ダウンロードして利用できるようになっています。

4.1.6 課題

世界開発指標 (WDI) の、データで、調べてみたいデータコードをいくつか見つけて、書き出してください。あとで利用しやすいように、データ名と、WDI コードをコピーして、メモ帳などに貼り付けておくと良いでしょう。さらに、そのコードに関する情報やリンクを、一緒に記録としておくと、あとで便利です。

4.2 オープンデータ (Open Data)

世界開発指標 (WDI) のように、公開されているデータを、オープンデータと呼びます。世界銀行などの、国際機関や、国際的に活動する非営利団体、それぞれの国の公的機関が、膨大なデータを公開 (オープンに) し、誰でも利用できるようになってきています。

パブリックデータ (Public Data) とも呼ばれますが、正確な定義があるわけではなく、各

機関によって、少しずつ考え方が違う面もあります。しかし、そのリーダーとでもいうべき、世界銀行は、オープンデータについて、厳密に定義をしています。

世界銀行のオープンデータ (Open Data) の定義を見てみましょう。

4.2.1 オープンデータの定義 (Open Data Defined)

オープンデータという言葉は、厳密な意味を持っています。データまたはコンテンツは、出所が明示されオープンという性質が維持されれば、誰でも自由に利用、再利用、再配布できるものを言います。

1. データは法的にオープンでなければなりません。つまり、パブリックドメインに置かれ、最小限の制限で自由に使用できなければなりません。
2. データは技術的にオープンでなければなりません。つまり、誰でも自由に使える一般的なソフトウェアツールを使ってデータにアクセスし、機械で読み取ることが可能な電子フォーマットで提供されていなければなりません。パスワードやファイアウォールによる制限を受けずに、公共のサーバーで、だれでもアクセスできなければなりません。また、オープンデータを見つけやすくするために、さまざまな組織がオープンデータカタログを作成し管理して必要があります。

上の定義で使われている、一つ一つの言葉について、詳細は述べませんが、最初に、はじめてのデータサイエンスで、例を示し、この章で、WDI を例にとって説明してきましたので、みなさんも、基本的な部分は、ご理解いただいたと思います。

4.2.2 オープンデータの意義

一旦、足を止めて、みなさんに考えていただきたいと思います。

1. オープンデータは誰に対してオープン (公開) になっているのでしょうか。
2. 公的 (Public) データというとき、公的とはどのような意味でしょうか。
3. なぜ、膨大な公的データのオープン化が進み、たいせつにされているのでしょうか。

4.2.3 オープンデータの利活用

さまざまな国際機関では、データを、Excel 形式や、CSV (Comma Separated Values) 形式などで、提供する以外に、ダッシュボード形式で、グラフを生成するなどして、データの可視化をある程度できるようにしています。さらに、コンピュータのアプリケーションでデータを直接取得できるように、API (Application Program Interface) を提供しています。

世界開発指標 (WDI) は、この章で説明してきたように、世界の状況やそれぞれの国についての、1400 余のさまざまな指標についてのデータを、長期間にわたって (1960 年ごろ

から) 提供している基本的なデータベースで、最初に調べてみることをお勧めします。さまざまな課題を理解するためにも、たいせつだと思います。

さらに、WDI は、この膨大な指標についてのデータすべてを統一した形式で提供しているという特徴があります。したがって、複数の指標についての関係を調べることも、容易にできます。実は、データサイエンスを始めると、違ったデータを一つにまとめて分析することは、技術的にそれなりに困難があります。それが、WDI では、ほとんどないのです。そのために、非常に使いやすいデータベースになっています。

データを、ダウンロードして、読み込むことが必要になりますが、API が整備されていると、分析ソフト（「データサイエンスをはじめましょう」では R を中心的に使いますが）に直接、取り込むことができるので、データの出所も明確で、他の人も同じように取得できれば、データ自体の、Reproducibility（再現可能性）も担保できます。はじめてのデータサイエンスでも、すべてのコード（コンピュータプログラム）が書かれていたと思います。基本的には、これを、たとえば、コピー、ペーストすれば、皆さんのコンピュータでも同じ結果が得られるということです。これについては、まだ、みなさんは、実感が持てないかもしれませんね。これから、少しずつ、説明していきます。

プログラミング言語 R を使い始める前に、データに慣れることも大切だと思いますから、次には、さまざまな、オープンデータについて、簡単な説明と共に、付随している、ダッシュボードの利用の仕方も説明しながら、みていきたいと思います。

第 5 章

データサイエンスノートブック

データサイエンスの記録について書きます。また、そのために、必要なツールについても、少しだけ説明します。

この「データサイエンスをはじめましょう」では、R で、自分でコードを書き、R Markdown や、Quarto に記録し、データサイエンスを進めていくことを目指しています。しかし、第一部では、まず、データを見ることに慣れるために、それぞれのサイトが提供する、ダッシュボードを使ってデータを見ていくことを、紹介します。

そのときにも、ノートを作成し、記録をとっていくことは、とても大切です。その説明を少しだけ書きます。

5.1 再現性のために記録すべきこと

根拠を明確にする (evidence based) ことが、データサイエンスにおいて、必須であることは、すでに、書きました。これから、データを見て行きますが、そのときに、基本的な情報を、記録しておくことをお勧めします。それが、今後のためにも有用ですし、その習慣をつけることが大切だからです。いくつか、記録すべき項目を書いておきます。

1. 日付：そのデータを調べた日付を書いておきます。サイトの内容が変更になる場合もあります。
2. データ名：もし、そのデータを特定するデータコードがあれば、それも記録しておきましょう。
3. データリンク：インターネット上のアドレスです。ブラウザ（Google Chrome、Edge、Safari などのホームページ閲覧ソフト）の上の窓に、URL (Universal Resource Locator) が表示されますから、それを記録しておきましょう。データ自体の URL を取得できる場合もありますが、そのデータが置かれている、ページ (Website) の URL だけが、取得でき、ダウンロードボタンでダウンロードする形式になっている場合もあります。その場合、右クリックや、Ctrl+Click で、データの、URL が取得できる場合もありますが、できない場合もあります。

- データをダウンロードした時は、そのファイル名と、ダウンロードした日付も忘れずにかいておくことをお勧めします。
4. メタデータ：また説明しますが、データには、データについてのデータ（これをメタデータと言います）が付いていることが多いです。最初からすべて記録する必要はありません。上の、データリンクがあれば、必要な時に、戻ることができます。しかし、データの定義や、変数の定義、データの収集方法などによっては、自分が求めているものではなかったり、データ自体がオリジナルデータではない場合もあります。最初から、詳細を記録する必要はありませんが、注意をはらうことを習慣にしておくことは大切です。
 5. ダッシュボードを設定したパラメーター（何を意味するかは少しずつ説明して行きます）：ていねいに、記録するのは、大変ですが、再現には、パラメーター情報は、必要です。場合によっては、そのリンクが提供され、そのリンクを使って、同じものが再現できる場合もあります。あるいは、埋め込み (embed) するための iframe link というものが提供されている場合もあります。もし、それがあれば、記録しておいてください。実際には、HTML 文書に埋め込んでも、そのままでは表示できない場合もありますが、設定値が含まれていますので、少し慣れれば、再現することも可能です。
 6. コメント：そのデータからわかったこと、疑問点、さらに知りたいことなど。少しでも書いてあると、あとで、とても便利です。このようなものが、データサイエンスの核でもありますから、ぜひ、記録しておいてください。

5.2 技術的なこと

5.2.1 ブラウザーの言語と様々な翻訳機能

1. ブラウザーの言語：実は、コンピュータのシステム言語が関係しますが、ほとんどの場合、システムの言語は変更せずに、ブラウザの言語を変更できるようになっています。
 - Windows でも、Mac でも、Google Chrome が使えますから、Google Chrome で説明すると、「Google Account を管理」から、設定できます。
 - Google Public Data Explorer と検索してみてください。すると、日本語の場合には、ほとんど出てきませんが、英語だと膨大なデータがあります。このサイトの場合には、右上に言語とでますから、言語を英語に変更すると、たくさんのデータを見ることができます。検索エンジンの言語によって、表示されるものが、大きく変化しますから、わたしは、Google のアカウントを複数使って、それぞれで、違う言語設定にしています。言語を変更できるようになると、検索の世界がとても広がります。
2. ブラウザーの翻訳機能：最近のブラウザには、翻訳機能が付属しており、簡単に切り替えることができます。しかし、ブラウザによって、設定方法が異なります

ので、調べて、いつでも使えるようにしておいてください。必要に応じて、翻訳機能の ON/OFF ができるととても便利です。

3. DeepL などのアプリの翻訳機能：最近、ブラウザの翻訳機能の質も向上しているのですが、わたしは、DeepL も併用しています。サイト上でも、翻訳ができますし、アプリをダウンロードして、設定を確認すると、翻訳したい箇所を選択すれば、すぐ翻訳してくれるショートカット機能があります。
4. ChatGPT など AI の翻訳機能：最近、LLM (Large Language Model) の発達で、様々な AI による、翻訳の質も非常に向上しています。ここでは、ChatGPT と書きましたが、他にも様々な AI で、翻訳が可能です。長い文章は、字数を指定して要約をしてもらうことも可能ですから、慣れると世界が広がって行きます。

5.2.2 画像

1. グラフのダウンロード：グラフ (graph, chart とも言います) は、画像になっていますから、あとで、利用する場合は、リンクを取得して、そのリンクで同じものを開くことができる場合もありますが、ダウンロードして保存しておくほうが安全です。ダウンロード方法が書いてあったり、右クリックまたは、Ctrl+ クリックで、ダウンロードできる場合が多いと思います。また、ダウンロードしたあとに、ノートに貼り付けておくことができれば、そのほうがあとで利用するとき便利です。取得したサイトの URL や、取得した日付も記録しておくことをお勧めします。
2. 画面収録 (Screen Capture)：ダウンロードできない場合、その方法が見つからない場合は、画面収録も一つの方法です。Windows, Mac によって方法が異なりますから、あらかじめ調べておくとういでしょう。

5.2.3 データファイル

データファイルは、CSV (comma separated values カンマで区切られたテキストデータ) などのテキストデータ、Excel ファイル、または、これらを、圧縮したり、いくつかのファイルをまとめて圧縮したりしてある場合があります。以下、少しだけ、注意点や、確認すべき点を書いておきます。

1. 圧縮されている場合の解凍方法を確認しておくこと。Windows か Mac でも違いますから、解凍方法を確認しておいてください。
2. CSV が一般的ですが、他にも、区切り文字が、スペースだったり、TAB だったり、縦棒だったり、様々な形式があります。R を使うようになれば、どの形式であっても、読み込めますし、変換することも可能ですが、二種類の問題が一般的です。
 - 上に書いた区切り文字の違い
 - Encoding の問題。こちらは、日本語などを含むファイルではよく起こる問題です。いわゆる文字化けが生じて中身が読めない場合があります。

- すべての対策を書くことはできませんが、区切り文字の違いは、Excel の機能でも、解決できます。Microsoft Office も高額なソフトですから、持っていないという場合は、機能限定ですが、Online 版は無料で使えますから、試してみてください。Google Spreadsheet でよみこいむことができる場合もあります。

5.3 まとめ

本書で紹介する、R を使い始めれば、統一した方法で解決できる課題もありますが、記録を取るということは、基本的ですから、簡単に書きました。

上のような記録を何に書くかということは書きませんでした。基本的には、記録を取ることが大切で、何に記録することはあまり、問題ではありません。しかし、リンクを貼り付けて、すぐ開くことができたり、画像を貼り付けたりが、できると便利でしょう。

使い慣れたものを使ってください。できれば、どこでも使えるような Cloud 型のサービスがお勧めです。他の方におすすめを聞くのも良いかもしれません。

次からは、実際に、オープンデータのサイトに行って、データを見ることを経験して行きたいと思います。

第1部

第一部オープンデータ

第 6 章

オープンデータ

6.1 概要

すでに、オープンデータについて、説明をし、その例として、世界銀行の世界開発指標 (World Development Indicators (WDI)) の説明をし、その指標の一つである、GDP (Gross Domestic Product 国内総生産) を例にとって、はじめてのデータサイエンスというタイトルで、データサイエンスの実際を見ました。

第一部では、世界銀行や、経済協力開発機構 (OECD)、国際連合 (United Nations) などの、国際機関のオープンデータや、日本のデータとして、政府機関統計 (e-Stat) の紹介をします。

さらに、実際に、それぞれのサイトのオープンデータを、ダッシュボード (dashboard) 呼ばれる対話型 (Interactive) の機能の使い方を紹介しながら、データを見ていきます。

また、実際のデータの取得 (ダウンロード) にも触れ、それぞれの期間が提供する API (アプリケーションプログラムインターフェース) の活用についても、簡単に触れていきます。

第二部で、R を使ったデータサイエンスについて学びますが、その例においても、いくつかのオープンデータを用いる機会がありますので、第一部は、その準備としての位置付けです。オープンデータの分析のより詳しい説明は、第三部で行います。

R を使ったデータサイエンスを、早く学びたい方は、この第一部をスキップして、第二部から読んでくださって構いません。しかし、みなさんの中には、R を使うところまでできるかは自信がないけれど、実際のオープンデータを見て、データサイエンスではどのようなことを考えるのかを体験してみたいという方もおられると思い、第一部を書いています。

また、実際に、オープンデータを活用して、ある課題について調べるときには、基本的なオープンデータについての知識は有用です。どこに、そのようなデータがあるかを見つけることがたいせつであるとともに、データサイエンスはなんと言っても、理論ではなく、実証的なものですから、実際のデータに触れながら、学んでいくのがたいせつだと考えているからです。

第一部の学びを通して、こんなふうに、データがたくさん、公開されているのか、と、そんな感触を持っていただければと思います。

6.2 さまざまな機関のオープンデータ

すでに少しだけ紹介した世界銀行以外にも、多くの機関がデータを提供しています。いずれも、使いやすくなってきています。少しずつ、いくつかのデータベースに、アクセスして、できれば、API の利用の仕方も、習得して、データを調べることに少しずつ慣れていただければと思います。

下のリストは、私が個人的に、何度か使ったことのある、データベースです。「データサイエンスをはじめましょう」でも、少しずつ消化していきます。他にも、たくさんのデータベースがありますので、ぜひ、調べてみてください。

- 国際連合 UNdata
- 経済協力開発機構 OECD data、
- 世界格差データベース (World Inequality Database) WID、
- 欧州連合の統計局 Eurostat、
- データで見る私たちの世界 Our World in Data

6.2.1 日本のデータ

日本では、それぞれの政府機関でデータを膨大なデータを公開していますが、次のサイトで、それらをまとめて公開しています。

- e-Stat：政府統計の総合窓口 - 統計で見る日本
- ダッシュボード：対話型形式でのデータの視覚化

6.2.2 持続的開発目標 (SDGs) データ

持続的開発目標 (Sustainable Development Goals (SDGs)) は、皆さんもご存知かと思います。2015 年に国際連合で定められ、2030 年までの達成目標が掲げられています。様々なサイトがありますから、ご存知かと思いますが、SDGs は、17 の分野に分かれており、それぞれにさらに細かい分類があり、その一つ一つに、数値目標が掲げられています。数値目標だけに、依存することは、十分ではないかもしれませんが、少なくとも、その数値目標の達成を検証するために、データが公開されています。

自分が興味をもっている課題に絞って、各サイトでどのようなデータが提供されているかをみていくのもよいと思います。

- 世界銀行：[https://databank.worldbank.org/source/sustainable-development-goals-\(sdgs\)](https://databank.worldbank.org/source/sustainable-development-goals-(sdgs))
- 国連：<https://unstats.un.org/sdgs/dataportal/database>

- データで見る私たちの世界：Our World in Data SGD Tracker: <https://sdg-tracker.org/>
- SDG Index: <https://dashboards.sdgindex.org/map>

6.3 オープンデータについて

オープンデータについては、すでに、世界銀行の定義をご紹介しました。パブリックデータをいう言葉も使われます。それらの違いなども含めて、AI に聞いてみましょう。

Q. オープンデータ (Open Data) とか、パブリックデータ (Public Data) と言われるものは、どのようなものですか。定義のようなものと、いくつかの例、さらに、重要視されてきている背景を説明してください。

GPT4. (2023.8.15)

オープンデータ (Open Data) とは、誰でも自由に利用、再配布、再利用することができ、制限がない、または限定的な制限だけが存在するデータを指します。オープンデータは一般的には無償で提供され、機械可読形式（たとえば CSV、JSON など）で利用可能であり、再利用のためのライセンスが明確にされています。

一方、**パブリックデータ (Public Data)** は公共の手段で取得されたデータ、または公共の利益のために公開されるべきデータを指します。これは政府や公共機関、研究機関などが収集したデータで、しばしば公共の利益のために公開されます。

オープンデータとパブリックデータの主な違いは、オープンデータが再利用と再配布の観点から「オープン」であることを強調するのに対し、パブリックデータは単に公開されているだけで、再利用や再配布については必ずしも明示されていないことです。

例えば、以下のようなものがあります：

1. 気象データ：天気予報、気温、風向、風速、降水量など
2. 交通データ：交通渋滞、駐車場の利用状況、公共交通機関の運行状況など
3. 統計データ：人口、雇用、健康、教育などのデータ
4. 環境データ：汚染レベル、森林破壊、気候変動などのデータ
5. 地理データ：地図、土地利用、地質学的情報など

オープンデータとパブリックデータが重要視されてきた背景には、以下の要素があります：

1. **透明性と説明責任**：データが公開されることで、政府や企業の行動についての透明性が高まり、説明責任が強化されます。
2. **イノベーションと経済的価値**：データが自由に利用できることで、新たなビジネスモデルや製品、サービスの開発が促進され、経済的価値が創出されます。

3. **市民参加:** データへのアクセスが提供されることで、市民が社会的・政治的な決定プロセスに参加する機会が増えます。
4. **社会的・環境的課題の解決:** 大量のデータを解析することで、社会的・環境的な課題をより良く理解し、解決策を見つけることが可能になります。

以上のように、オープンデータとパブリックデータは、透明性、説明責任、イノベーション、市民参加、社会的・環境的課題の解決などの観点から重要性を増しています。

6.3.1 オープンデータの未来

みなさんは、どのように考えますか。まだ、発展途上で、特に、パブリックデータについては、国によって、考え方も一定ではないように見受けられますが、これから、どのような世界を目指していくのか、データサイエンスは、どのような目的のために必要なのか。みなさんは、なんのために勉強しているのか、上の一つ一つの項目をもとに、周囲のみなさん、友人などと、ディスカッションをしながら、考えていただければと思います。

いずれもたいせつな項目だと思いますが、項目3の市民参加は、公開されているデータをわたしたち市民が利用して初めて可能になることです。項目4の、社会的・環境的課題の解決には、私たち市民も責任をもって、取り組むべきだということが基盤にあるのだと思います。みなさんは、どう考えられますか。

6.4 ダッシュボード

はじめて、オープンデータを利用するときには、対話型のダッシュボードは、助けとなります。ダッシュボードとはどのようなものか、AIに説明してもらいましょう。

Q. データを対話的にみたり視覚化したりする、ダッシュボード (dashboard) について説明してください。

ChatGPT (2023.08.15)

ダッシュボード (dashboard) は、情報やデータを一元化し、対話的に表示・分析・視覚化できるインタラクティブなユーザーインターフェースです。主にビジネスや組織内で使用され、重要な指標や情報をリアルタイムに監視・分析するために利用されます。

ダッシュボードは、さまざまなデータソースからの情報を収集し、それを一つの場所にまとめて表示します。これにより、膨大なデータを迅速に把握し、傾向やパターンを特定することが可能になります。例えば、売上データ、顧客の統計情報、ウェブ解析データ、ソーシャルメディアの指標など、さまざまなデータソースからの情報を一つのダッシュボードに表示することができます。

ダッシュボードは、視覚化ツールを使用してデータをグラフやチャート、表などの形式で表示します。これにより、データを直感的に理解しやすくし、視覚的なパターンや相関関係を把握することができます。例えば、売上の推移を折れ線グラフで表示したり、地域別

の売上を地図上に表示したりすることができます。

ダッシュボードは、利用者が必要な情報に素早くアクセスできるように設計されています。ユーザーは、ダッシュボード上の要素をクリックしたり、フィルタを適用したり、グラフをズームしたりすることで、詳細な情報を取得したり特定の視点でデータを分析したりすることができます。

また、ダッシュボードはリアルタイムな情報の監視にも利用されます。データソースが更新されるたびに、ダッシュボード上の情報も自動的に更新されます。これにより、組織内の関係者が最新の情報にアクセスし、迅速な意思決定を行うことができます。

ダッシュボードは、情報の可視化と分析によって、効率的なデータ駆動型の意思決定を支援する役割を果たします。組織内のさまざまなレベルの利用者が、ビジネスのパフォーマンスを把握し、問題や機会を特定し、戦略やアクションプランを立案するために活用します。

Q. 例や、ダッシュボードのサイトを教えてください。

ChatGPT (2023.8.15)

もちろんです！以下にいくつかのダッシュボードの例と、一般的に使用されるいくつかのダッシュボードサイトをご紹介します。

ダッシュボードの例:

1. Google Analytics: ウェブサイトのトラフィックやユーザー行動などの情報をリアルタイムに監視・視覚化するためのダッシュボードです。
2. Salesforce: セールスや顧客関連のデータを統合し、ビジネスのパフォーマンスを追跡するためのダッシュボードです。
3. Tableau: データの視覚化と分析に特化したダッシュボード作成ツールであり、さまざまな業界で利用されています。

ダッシュボードのサイト:

1. Klipfolio (<https://www.klipfolio.com/>)
2. Geckoboard (<https://www.geckoboard.com/>)
3. DashThis (<https://dashthis.com/>)
4. Databox (<https://databox.com/>)
5. Microsoft Power BI (<https://powerbi.microsoft.com/>)

これらのサイトでは、ダッシュボード作成ツールやサービスを利用して、カスタムのダッシュボードを作成することができます。それぞれのサイトには、詳細な情報やデモが提供されている場合もありますので、ぜひご覧ください。

上で、ダッシュボードのサイトとあるのは、これらのサイトのツールを使って、ダッシュボードを作ることができるというものです。少し、ダッシュボードの利用に慣れてきたら、挑戦してみてもいいのではないでしょうか。

6.5 API を利用したパッケージ

それぞれのオープンデータを検索したり、読み込んだりするための R のパッケージの紹介をします。上に書いたように、第二部で利用するためです。詳しい使い方は、第三部で紹介します。

基本的には、以下の項目について紹介します。

- R の (公式) パッケージサイト：パッケージも含め、R の管理をしている CRAN の公式サイトの情報です。この下の情報も、ほとんどの場合、このリンク先に掲載されています。
 - 資料 (Materials)：README (はじめにお読みください) などで、基本情報が書かれています。
 - マニュアル (Manual)：利用者用説明書です。パッケージで利用可能な関数 (命令、データ) などの情報がすべて書かれてあります。それぞれの関数 (命令) に関しては、R Studio 内の Help (ヘルプ検索窓) から利用可能です。
 - 使い方の例 (Vignette)：開発者のサイト (GitHub (バージョン管理システムの支援サイト) など) にある場合もありますが、最近は、公式パッケージサイトに、Vignette として使い方の例が掲載されている場合が増えてきています。すべて理解しようとせず、使い方の例からまずはみてみるのが有効です
-

第 7 章

世界銀行 (World Bank)

7.1 概要

世界銀行は、貧困削減と持続的成長の実現に向けて、途上国政府に対し融資、技術協力、政策助言を提供する国際開発金融機関です。2030 年までに極度の貧困をなくし、各国の下位 40 パーセントの人々の所得を引き上げて繁栄の共有を促進するという 2 つの目標を掲げています。

世界銀行のサイトを見てみると、下の方にはグラフも出ており、さまざまな統計データを提供することに力を入れていることがわかります。

すでに、世界開発指標については、簡単に説明しました。ここでは、世界銀行のサイトの中のデータについて、見ていくとともに、ダッシュボード (dashboard) の活用、データの取得方法や、API (Application Program Interface) を用いて、R でデータを検索したり、取得するパッケージの紹介を簡単にします。

7.2 データベース

7.2.1 三つのサイト

- 世界銀行オープンデータ
- データカタログ
- 世界開発指標 (WDI)

一つ一つみていきましょう。

英語を読むことが苦手な場合は、ブラウザ (Google Chrome, Edge, Safari などのホームページ閲覧ソフト) の自動翻訳機能を使うのも良いでしょう。しかし、自動翻訳機能を利用していると、リンク先に飛べなかったり、いくつかの機能が使えないこともありますので、正常に機能しないときは、自動翻訳機能を一旦 OFF にして利用してください。

7.2.2 世界銀行オープンデータ

世界銀行のオープンデータ全体にアクセスできるサイトです。上の検索窓の下に、Country (国) と Indicators (指標) と書いてあります。

その下には、いくつかのグラフや、最近のニュースやトピックが書かれており、さらにその下には、More Resources (さらに...) といくつかの項目があります。Open Data Catalog、Data Bank、などと共に、World Development Indicators もあります。この World Development Indicators (WDI) を選択すると、上の三つのサイトの三番目に飛びます。また、Data Bank を選択すると、三つのサイトの二番目に飛びます。

最初に書いた、Country (国) と Indicators (指標) から説明しましょう。

7.2.2.1 国別サイト

Country (国別) のサイトを選択すると、国のリストが出てきます。

J のところに、Japan (日本) がありますから、選択してみてください。日本のさまざまな指標とグラフが出てきます。

Indicator (指標)、Most Recent Values (直近の値)、Trend (傾向) が表示されます。指標をクリックすると、大きなグラフが出てきます。その指標のサイトですから、そこで、その指標についての他の国の状況などを確認することができます。そこでは、選択した指標と似た指標が選択できたり、もっと詳しいことを調べるデータバンク (Data Bank) へのリンクもあります。このサイトから、データをダウンロードすることもできるようになっています。

また、国別サイトの右の方には、地域など、グループに分けてある帯もあります。その一番下には、World (世界) もありますから、世界全体について見てみたり、収入の階級に分けたサイトの指標を見ることもできるようになっています。

7.2.2.2 指標別サイト

Indicators (指標) を選択すると、トピックに分けて、指標が並んでいます。

よく見ると、一番上に、Featured Indicators (特徴的な指標) と、All Indicators (すべての指標) とあり、最初に開いているのは、特徴的な指標の方であることがわかります。すべての指標の方を開けると、よりたくさんの指標を見ることができます。

トピックは、以下のものに分かれています。

- Agriculture & Rural Development 農業と農村開発
- Aid Effectiveness 援助の有効性
- Climate Change 気候変動
- Economy & Growth 経済と成長

- Education 教育
- Energy & Mining エネルギーと鉱業
- Environment 環境
- External Debt 対外債務
- Financial Sector 金融セクター
- Gender 性別
- Health 健康
- Infrastructure インフラ
- Poverty 貧困
- Private Sector 民間部門
- Public Sector 公共部門
- Science & Technology 科学技術
- Social Development 社会開発
- Social Protection & Labor 社会的保護と労働
- Trade 貿易
- Urban Development 都市開発

それぞれの指標を選択すると、グラフが表示される画面が出てきます。それは、上で国別のところから選択したものと同じです。

GDP を調べるときには、NY.GDP.MKTP.CD という、コードを指定して、データを取得しました。この、Indicator Code (WDI コード) は、Details (詳細) を見ると、その指標の概要とともに、書いてあります。そのデータコードは、そのページの上の URL にも表示されています。

はじめてのデータサイエンスでは、WDI という R のパッケージを使って、データを読み込みました。そのときに必要だったのが、この Indicator Code でした。あとで、詳しく調べてみたい指標がありましたら、その、WDI コード (Indicator Code) を、あとで、使えるように、メモなどに貼り付けて (Copy-Paste) おくことを、お勧めします。

7.2.3 オープンデータカタログ (Open Data Catalog)

The Data Catalog is designed to make World Bank's development data easy to find, download, use, and share. It includes data from the World Bank's microdata, finances and energy data platforms, as well as datasets from the open data catalog. There are different ways to access and download datasets.

データカタログは、世界銀行で編纂した開発に関するデータを簡単に検索、ダウンロード、使用、共有できるように設計されています。これには、世界銀行のマイクロデータ、財務、エネルギーデータプラットフォームからのデータ、およびオープンデータカタログからのデータセットが含まれています。データセットにアクセスしてダウンロードするには、さまざまな方法があります。

世界銀行 (World Bank) で編纂したり、他の機関から提供を受けたデータがリストされ

ています。

一番上には、Search Box (検索窓) があり、その下には、Featured (特徴的な、またはお薦め) とあり、いくつものトピックが並んでいます。右に、スクロールするとさらにいくつものトピックを見ることができます。その中にも、上で述べた世界開発指標 (WDI) もありますし、Covid-19 (コロナウイルス感染症) 関連のデータもあります。

それぞれの、トピックに、関連のデータがリストされています。

7.2.4 世界銀行 (World Bank)

このページの最初にも書きましたが、簡単にまとめておきましょう。

- 世界銀行 (World Bank) : <https://www.worldbank.org>
- 世界銀行について (Who we are) :
 - 極度の貧困状態の削減 (To end extreme poverty) : 2030 年までに、極度の貧困状態にある世界人口の割合を 3% に削減する。By reducing the share of the global population that lives in extreme poverty to 3 percent by 2030.
 - 繁栄を共に享受 (To promote shared prosperity) : すべての国の最貧困層の 40% の人々の所得を増加させることによって共栄を促進。By increasing the incomes of the poorest 40 percent of people in every country.
- 世界銀行オープンデータ (World Bank Open Data) : <https://data.worldbank.org>
 - Data Bank, World Development Indicators, etc.

日本について : <https://data.worldbank.org/country/japan?view=chart>

7.2.5 世界開発指標 (World Development Indicator)

すでに紹介しましたが、簡単にまとめておきます。

- World Development Indicators (WDI) : 世界銀行が開発に関する各国間比較可能なデータの集大成である 1400 の時系列指標 (the World Bank's premier compilation of cross-country comparable data on development; 1400 time series indicators)
 - テーマ別 (Themes) : 貧困と格差、人間、環境、経済、国家と市場、グローバルリンク集 (Poverty and Inequality, People, Environment, Economy, States and Markets, Global Links)
 - オープンデータとデータバンク (Open Data & DataBank) : Explore data, Query database
 - すべてのデータおよびメタデータを Excel または CSV 形式で、一括してダウンロードすることもできるようになっています。Bulk Download:
 - コンピュータを使って読み込む場合のデータの仕様が書かれています。API (Application Program Interface) Documentation

1400 ものデータがありますから、すべてのデータやメタデータをダウンロードすれば、すぐ、データを調べることができるわけではありません。基本的なことをおく必要がありますね。上にもリンクのある、データカタログから、世界開発指標 (World Development Indicators) を選択すると、次のサイトにリンクがついています。

このページからは、Databank (ダッシュボード) へのリンクなどの他、Excel ファイルや、CSV ファイルで、メタデータを取得することもできるようになっています。WDI 全部のリストもここで見るができます。このファイルから探すのが最適とは言えませんが、そのようなファイルを持つておくことは便利です。

7.3 ダッシュボード (Dashboard)

世界銀行のダッシュボードには二種類あります。一つは、それぞれの指標についてのダッシュボード、もう一つは、データバンク (DataBank) です。

7.3.1 World Bank アカウント

ダッシュボードは、リンクにアクセスすれば、アカウントなどを作成せず、すぐに使えます。しかし、ダッシュボードを使って、グラフを作成したり、自分用の、データを作成したりした場合には、その結果を、保存をしたり、リンク (iframe 形式) を、文書に埋め込んだりする必要がある場合があります。その場合には、一般用アカウントを作成する必要があります。このサイトの Sign Up から、アカウントを作成してください。

7.3.2 指標毎のダッシュボード

はじめてのデータサイエンスで使った、GDP (Current US\$)、データコード NY.GDP.MKTP.CD についてみましょう。

いくつかの方法があります。

1. 世界銀行オープンデータ から、指標 (Indicator) を選択し、その中の Economy & Growth (経済と成長) の中から、GDP (current US\$) を選択すると、ダッシュボードが現れ、世界の GDP の推移のグラフが表示されます。
2. 世界銀行オープンデータ から、国 (Country) を選択し、例えば、J から、Japan を選択し、Economics (経済) の指標のGDP (current US\$) を選択すると、ダッシュボードが現れ、日本の GDP の推移のグラフが表示されます。指標によっては、上の、Theme (テーマ) や、Topic (トピック) から選ぶ必要がある場合もあります。
3. 世界開発指標 (WDI) の Data Theme (テーマ) の中の、ECONOMY (経済) の中から、GDP (current US\$) を選択すると、ダッシュボードが現れ、世界の GDP の推移のグラフが表示されます。
4. もし、WDI コード (この場合は、NY.GDP.MKTP.CD) を知っていれば、このコードを、世界銀行オープンデータの検索窓に入れて検索すると、上のダッシュボードのページが表示されます。

最初は、選択した指標について、世界か、日本など選択した国の、折れ線グラフ (Line Graph) が表示されていると思います。そして、下の方に、国のリストがあり、その一番下には、地域のリストが続いています*1。また、Line (折れ線グラフ) と書いた右には、Bar (棒グラフ) と、Map (地図) とあります。

その右には、Also Show (追加) とあり、Aggregates (総合)、Same Region (同じ地域)、Similar Values (近い値)、Highest Values (最高値)、Lowest Values (最小値) を表示することもできます。表示しないときは、None を選択します。

その右には、Share (共有) と、Details (詳細) があります。詳細には、その指標についての、詳しい説明があります。すべてを理解することはできないかもしれませんが、翻訳機能も使って、確認しておくことをお勧めします。詳細には、データコードも書かれています。記録しておく、次に同じ指標のデータを探すときに便利です。共有からは、Web Page や、SNS に埋め込む、iframe link を取得することができます。

右の方の帯には、関連する指標がリストされ、さらに、Download (ダウンロード)、Data Bank (データバンク)、WDI Tables (統計表) へのリンクがあります。

7.3.2.1 使い方

具体例としては、GDP (Current US\$) の世界 (World) のグラフが表示されているとします。上の検索窓に GDP (Current US\$) と入っていると思います。

- 国や地域の追加
 - 英語で入力しますから、グラフの下の、国や地域名のところから、追加したい、国や地域名を選んで、書き出しておいてください。一つずつ追加するときは、コピーするのも良いかもしれません。
 - 日本をグラフに追加するときは、Japan ですから、検索窓に、Japan と入れて少し待ちます。すると、窓の下に、Japan が表示されますから、それを選択 (クリック) します。すると、日本のグラフに変わります。India も加えてください。次に、検索窓に、United と入力すると、United Kingdom, United States, United Arab Emirates が表示されますから、United Kingdom を選択してください。Japan, India, United Kingdom の三つの国のグラフが同時に表示されます。これによって、いくつかの国のその指標における経年変化を比較することができます。
 - いろいろな国や地域を加えてみてください。たとえば、GDP を指標として、United States を加えると、United States の GDP の値が大きいので、他の国のグラフは下の方に重なり合うようになります。Afghanistan を加えると、ある年以前のデータがなかったり、値がとても小さいために、X 軸に張り付くようになってしまったりします。みやすいグラフを表示するには、どのような

*1 アルファベットの昇順になっていますが、そのリストの Country (国) の右にある 型のマークを押すと に変わり、降順になります。

ものの比較をするかも重要になってくるのがわかると思います。

- 追加した国や地域を消すのは、単にその国名の、右に表示される x マークを選択すれば良いですし、Delete キーでも消去できます。
- 実は、Japan など一つの国を加えてからあとは、下の国名などのリストから、追加したい国を選択すると、追加されていきます。検索窓に入れるよりも簡単だと思います。

- Also Show の活用

- 上に書いたように、Aggregates (総合)、Same Region (同じ地域)、Similar Values (近い値)、Highest Values (最高値)、Lowest Values (最小値) を追加できます。たとえば、Japan (日本) だけを残しておいて、Same Region を選択すると、いくつかの国の値が、薄く表示されます。地域名をみると、これは、East Asia & Pacific (東アジアと太平洋地域) であることが推測されます。
- そこで、East Asia & Pacific を選択すると、その地域の総計の値が追加されます。

7.3.2.2 データダウンロード

R のパッケージを用いたデータのダウンロードについてはあとから述べますが、指標毎のダッシュボードの右の帯のダウンロードからも、データをダウンロードすることができます。

CSV (Comma Separated Values)、XML (Extensible Markup Language)、EXCEL (Microsoft Excel Spreadsheet) と三つの形式でデータを取得できます。それぞれ、R などでも読み込むことができます。ただし、CSV と、XML は、Zip 形式の圧縮ファイルになっています。EXCEL は、三つのシート (Data, Metadata - Countries, Metadata - Indicators) が一つの、ブックになっています。Metadata (メタデータ) は、データのデータで、データについての情報が収められています。

7.3.3 データバンク (DataBank)

上で説明した指標毎のダッシュボードの右の帯からも、データバンクのダッシュボードが開きますが、世界銀行オープンデータの下にある、データバンクを選択すると、リストが表示されます。ここで、ある程度選択してから、ダッシュボードを表示させることもできます。一番上に、World Development Indicators (世界開発指標) がありますから、選択してください。あとから、他のものに変更することも可能です。

表が表示され、左の方に、Variables (変数)、Layout (レイアウト)、Styles (形式)、Save (保存)、Share (共有)、Embed (埋め込み) とあり、右上には、Table (表)、Chart (グラフ)、Map (地図)、Metadata (メタデータ) とあり、その下には、Preview (表示) などとなっていると思います。

膨大なデータを選択し、形式を整えて、ダウンロードしたり、グラフを表示したりすることができます。

7.3.3.1 例1

まずは、一つ目の例として、GDP (Current US\$) の値を大きい方から国順に並べてみましょう。

1. Variable の Database で、World Development Indicators を選択します。
2. Countries では、上に、All、Countries、Aggregates とありますから、Countries を選択します。
3. 一番左のチェックボックス Select All (すべて選択) を選びます。これで国が全部選択されました。現在ですと、Selected 217 と出ています。
4. Series は、いくつか選択されている可能性がありますから、X マークを選択して、まずは、全部選択を解除し、次に、GDP (current US\$) にチェックを入れます。Selected 1 となっていることを確認してください。
5. Time の一番左のチェックボックスから、Select All にしてください。現在では、Selected 63 と表示されました。
6. 右の方に、Selections have been modified ... (選択が変更されました ...) と出ますので、Apply Changes (変更を適用) を選びます。
7. 上の Layout タブを選択し、Time を Column (列)、Country を Row (行)、Series を Page (ページに指定します)

ここまでで、Table に、GDP (current US\$) についての表が表示されていることを確認してください。

これを書いている時点では、2022 年が最も新しいデータで、2022 の年のところをクリックすると、最初は、国名のアルファベット順になっていると思いますが、それが、その指標の値の、昇順、降順に変更できます。ここでは、GDP が大きな値の順に並べたいので、降順にします。

10 位までの国は、United States, China, Japan, Germany, India, United Kingdom, France, Russian Federation, Canada, Italy となっているかと思います。

リンク

7.3.3.2 例2

2022 年の値で、10 位までの国を選択して、折れ線グラフや、棒グラフなどを書いてみましょう。

1. Variables の Countries の X で選択を解除し、上の 10 カ国を選んでください。

2. 右の方に、Selections have been modified ... (選択が変更されました ...) と出ますので、Apply Changes (変更を適用) を選びます。すると表が 10 カ国のものになります。
3. 上の、Styles を選択し、Chat Type & Style で、Line (折れ線グラフ) を選択すると、しばらくして、10 カ国の、折れ線グラフを表示させることができます。色も変更することも可能です。
4. Chat Type & Style で Horizontal Bar を選択し、Layout で、

リンク

以下では、変数 (Variables) の選び方と、グラフ (Chart) について簡単に説明します。

7.3.3.3 変数 (Variables)

Database、Country、Series、Time とあり、それぞれの右に、Available と Selected とあります。

どのデータベースについて、国を選択し、系列を選び、期間を選択するという形式になっています。

Database の左の三角印を選択すると、データベース名が確認できます。現在は、World Development Indicators が選択されています。ここで、他のデータベースに変更することも可能ですが、まずは、そのままにしましょう。

次は、Countries (国) です。All、Countries、Aggregates と一番上にあります。国だけを表示するか、地域やグループを表示するか、すべてを表示するかを選択できます。

簡単のために、まずは、Countries (国) を選択しましょう。

国がいくつか選択されている場合もありますから、X マークをクリックして、すべて選択を消去し、国を選んでみましょう。GDP のところで経験したように、GDP の多い方から、United States、China、Japan、Germany、India、United Kingdom、France を選択してみましょう。

7.3.3.4 GDP per capita (constant 2015 US\$)

実質 GDP (2015 年を基準にしたもの) を、総人口で割った値。アメリカ合衆国、英国、ドイツ、フランス、日本、中国、日本、ロシア、ウクライナの 2021 年における比較棒グラフ - リンク

年次変化を示す折線グラフ -

7.3.3.5 Central government debt, total (% of GDP)

2020 年の政府の負債 (GDP の百分率) - リンク

政府の負債 (GDP の百分率) の年次変化を示す折線グラフ

7.3.3.6 CO2 emissions (metric tons per capita)

CO2 排出量 (1 人あたりのメートルトン) - リンク

CO2 排出量 (1 人あたりのメートルトン) の年次変化の折線グラフ

7.3.3.7 Military expenditure (% of GDP)

2021 年の軍事費 (GDP の%) - リンク

軍事費 (GDP の%) の年次変化

7.3.3.8 Military expenditure (current USD)

2021 年の軍事費 (現在の米ドル)

軍事費の年次変化

7.3.3.9 Proportion of seats held by women in national parliaments (%)

2021 年、国会で女性が占める議席の割合 (%) - リンク

国会で女性が占める議席の割合 (%) の年次変化

7.3.4 世界のさまざまな課題から見る

7.4 API

世界銀行 (World Bank) の API を利用した R のパッケージを二つ紹介します。

7.4.1 WDI

Search and download data from over 40 databases hosted by the World Bank, including the World Development Indicators ('WDI'), International Debt Statistics, Doing Business, Human Capital Index, and Sub-national Poverty indicators.

世界開発指標 (「WDI」)、国際債務統計、Doing Business、人的資本指数、準国家貧困指標など、世界銀行が主催する 40 以上のデータベースからデータを検索してダウンロードします。

- R のパッケージサイト : <https://CRAN.R-project.org/package=WDI>
- 資料 (Materials) : <https://cran.r-project.org/web/packages/WDI/readme/README.html>
- マニュアル (Manual) : <https://cran.r-project.org/web/packages/WDI/WDI.pdf>

- 使い方の例：<https://vincentarelbundock.github.io/WDI/>

7.4.2 wbstats

Programmatic Access to Data and Statistics from the World Bank API

世界銀行 API からのデータと統計へのプログラムによるアクセス

- R のパッケージサイト：<https://CRAN.R-project.org/package=wbstats>
- 資料 (Materials)：README
- マニュアル (Manual)：<https://cran.r-project.org/web/packages/wbstats/wbstats.pdf>
- 使い方の例 (Vignette)：<https://cran.r-project.org/web/packages/wbstats/vignettes/wbstats.html>

7.5 Google Public Data Explorer

Google のパブリックデータ探索 (Public Data Explorer) サイトの紹介とともに、世界開発指標をこれを使って見てみたいと思います。

Google で Public Data を検索すると、おそらく、次のサイトが見つかると思います。

<https://www.google.com/publicdata/directory?hl=ja&dl=ja#!>

これは、日本語サイトで、2023 年 9 月現在では、データの提供元の数 が 7 となっています。また、

右上に、言語とありますから、それで English (United States) を選択すると下のリンクに飛びます。ここには、45 と書かれています。

https://www.google.co.jp/publicdata/directory?hl=en_US&dl=en_US#!

英語版で使うことをお勧めします。例えば、上で見た World Development Indicators (世界開発指標) は、英語版だけでなく、日本語版にもありますが、中身を見てみると、国によってデータがなかったり、少し古いデータまでしかなかったりなどあるようです。どの指標の、どのデータとすべてを挙げることはできませんが、英語版を使った方が安全だと思います。指標について、英語で意味するものがよくわからない時は、ブラウザの翻訳機能を使って見当をつけるのも良いでしょう。

英語版には、上に例が出ていると思います。自動的にスライドしますが、一番最初は、世界開発指標で、私が確認したときは、Living Longer with Fewer Children (子供の数が少ないと長生き) という表題になっています。このグラフをクリックしてください。

7.5.1 例1 WDI: Living Longer with Fewer Children

グラフをクリックすると、このようなページが表示されると思います。

左の帯には、Public Data、World Development Indicators と書かれ、いろいろな項目が並んでいます。また、よく見ると、以下のように書かれています。

X 軸：Life Expectancy (生まれた時点の平均寿命)

Y 軸：Fertility Rate (出生率)

色：Region (地域)

サイズ：Population (人口)

下：矢印と 2017 の数

と出ているかと思いますが、何が書いてあるか確認してください。違っていても構いません。

これで大体理解できたと思いますが、これが、WDI のデータを元にして、2017 年時点での、平均寿命と、出生率の散布図で、丸の大きさで、人口を表し、色で地域を表しています。

下の矢印を押すと、1960 から始まって、どのように変化したかを見ることができます。Gapminder のところで、書いた、バブルチャートと言われるもので、ハンス・ロスリングが Google に管理を依頼したと言われています。

左には、Compare by という項目があります。それを開くと、Region、Lending Type、Income Level とあります。Region に Color と書いてあると思いますが、Income Level の右のプルダウンメニューを押し、一番下にある、Color by This を選択すると、右上の、凡例 (Legend) と言われるものが、Income Level に変わると思います。動かしてみるとわかりますが、右下に、High Income がまとまっていると思います。日本は、見つけられますか。

Country List のところの、日本にチェックを入れると、Japan と表示されますから、すぐ見つけられます。マウス (ポインタ) を丸に近づけると国をみることもできます。High Income でも、Fertility Rate が 2 に近い国もあることがわかります。どんな国がそうになっていますか。

この辺りでやめておきます。

右上には、折れ線グラフと、棒グラフと、地球のマークと、散布図のマークがあります。現在は、散布図が使われています。その右には、ギアマークと、リンクのようなマークがあります。

ギアマークで、X 軸や、Y 軸を対数にしたり、リンクから、このグラフのリンクを取得することもできます。リンクは二種類ありますが、上が、通常のリンク、下は、iframe リンクと言われるものです。

7.5.2 例 2 GDP per Cap vs CO2 per Cap in Log-Log

今度は、日本語でも、英語でも良いですが、一人当たりの GDP を X 軸に、一人当たりの CO2 排出量を Y 軸にとり、対数にして、表示してみましょう。

X 軸：1 人あたり GDP (実質値: 2010 年基準、米ドル表示) - 対数

Y 軸：1 人あたりの CO2 排出量 - 対数

サイズ：人口

色：地域

リンク

7.5.3 まとめ

例から始めましたので、二つの WDI の指標を使い、さらに、人口や、地域など、他の指標も、一つのグラフに含めたものを見てきました。最初にすごいものから始めてしまいましたが、大雑把には、次のようなものになっています。

- 折れ線グラフ (line graph)：一つの指標の時系列での変化。
 - いくつかの国や地域などについての値を用いて、色で区別して表示することも可能です。
- 棒グラフ (bar graph)：一つの指標をいくつかの項目 (国や、地域など) を表示
 - 特定の年の、特定の指標の値を、いくつかの国や地域について表示
- 色付き世界地図 (choropleth map)：カテゴリごとに、色を変えて、地図上に表示
 - 個人の収入の多寡などのグループ (income level) ごとに色を変えて、地図上に表示することなどが可能です。
- 散布図 (scatter plot)：二つの指標の関係性を表示
 - さらに、点などの大きさにも他の指標の情報を加えたり、色などで、カテゴリごとの情報を加えることも可能です。

最後に一番最初に挙げた、Living Longer with Fewer Children で何種類の情報が表示されているか見てみましょう。同時に、数値か、カテゴリ (グループ) かも書いておきます。

1. Life Expectancy (生まれた時点の平均寿命)：数値 - X 軸
2. Fertility Rate (出生率)：数値 - Y 軸
3. Country (国)：カテゴリ：点

4. Population (人口) : 数値 : 点の大きさ

5. Region (地域) : カテゴリー : 点の色

6. Year (年) : (離散的 : とびとびの) 数値 : 一枚ごとのスライド

ほかにはありますか。普通は二つの指標しか表せないように思いますが、ここでは、6つの情報が入っていますね。

いろいろと調べてみませんか。

なかなか素晴らしいですね。ただ、2014年にプロジェクトがスタートしてから、2017年、2020年と更新されていますが、更新の頻度はあまり高くないように見えます。例としては、十分機能していると思いますが。

第 8 章

OECD（経済協力開発機構）

8.1 概要

OECD（経済協力開発機構）はヨーロッパ諸国を中心に日・米を含め 38 ヶ国の先進国が加盟する国際機関です。OECD は国際マクロ経済動向、貿易、開発援助といった分野に加え、最近では持続可能な開発、ガバナンスといった新たな分野についても加盟国間の分析・検討を行っています。

https://www.meti.go.jp/policy/trade_policy/oecd/index.html

8.2 データベース

8.3 OECD

OECD Data: <https://data.oecd.org/>

Topic と Country に分かれて、調べられ、検索も可能です。

8.3.1 テーマ別

- Agriculture
 - Agricultural output
 - Agricultural policy
 - Fisheries
 - Sustainable agriculture
- Development
 - Development resource flows
 - Official development assistance (ODA)

- Economy
- Education
- Energy
- Environment
- Finance
- Government
- Health
- Innovation and Technology
- Jobs
- Society

8.3.2 例

OECD Chart: Inflation (CPI), Total / Food / Total less food, less energy, Annual growth rate (%), Monthly, Jul 2023

OECD Chart: Quarterly GDP, Total, Percentage change, previous period, Quarterly, Q1 2023

8.4 ダッシュボード

OECD.Stat <https://stats.oecd.org/Index.aspx>

8.5 API

第 9 章

United Nations (国際連合)

9.1 概要

9.2 データベース

UNdata: <https://data.un.org>

膨大なリストがあります。さまざまな機関からデータが提供されていることもあり、WDI のように、規格が統一されてはいません。

9.3 ダッシュボード

いくつかのトピックについては、ダッシュボードがあります。

9.4 API

一部のデータのための API があります。

第 10 章

Our World in Data

10.1 概要

サイト：<https://ourworldindata.org>

様々なトピックについてデータからのレポートがあり、そのデータも見ることができるようになっています。さらに、使いやすい API も提供されています。

一番上の帯の Browse by Topic からトピック探すことができます。

- Population and Demographic Change 人口動態
 - **Population Growth**
 - **Age Structure**
 - **Gender Ratio**
 - **Life Expectancy**
 - **Child and Infant Mortality**
 - **Fertility Rate**
 - **Urbanization**
 - **Migration**
- Health 健康
- Energy and Environment エネルギーと環境
- Food and Agriculture 食料と農業
- Poverty and Economic Development 貧困と開発経済
- Education and Knowledge 教育と知識
- Innovation and Technological Change 技術開発とイノベーション

- Living Conditions, Community and Wellbeing 生活環境、共同体と幸せ
- Human Rights and Democracy 人権と民主主義
 - **LGBT+ Rights**
 - **Women’s Rights**
 - **Child Labor**
 - **Human Rights**
 - **Democracy**
 - **Corruption**
- Violence and War 暴力と戦争
 - **Biological and Chemical Weapons**
 - **War and Peace**
 - **Military Personnel and Spending**
 - **Terrorism**
 - **Nuclear Weapons**
 - **Violence against children and children’s rights**
 - **Homicides**

これら以外にも、コロナウイルス感染症や、持続的開発目標（SDGs）についてのサイトなどもあります。

10.2 レポート

10.2.1 Population Growth

人口において、日本は、1950年に何番目で、2021年には何番目かわかりますか。上のダッシュボードからもわかります。

10.2.2 War and Peace

1945年以降のデータが主ですが、世界の紛争などの死者数などを知ることができます。古いデータも少しだけあります。

<https://ourworldindata.org/war-and-peace#the-decline-of-wars-between-great-powers>

10.3 データベース

owid: <https://ourworldindata.org/>

10.4 ダッシュボード

10.4.1 使い方

1. Chart グラフ

- いくつかの国（変更可能な場合もあります）についての年次変化を表す折れ線グラフや棒グラフなど

2. Choropleth 色付き地図

- 右上のプルダウンメニューからは、地域（アフリカ、北アメリカ、南アメリカ、アジア、ヨーロッパ、オセアニア）を選択して拡大することができます。
- それぞれの国にマウスまたはポインターを重ねると、その国の状況を見ることができます。

3. Table 表

- データの一部が表となっています。降順・昇順などの並べ替えも可能です。

4. Sources 出典

- 変数の簡単な定義とともに、World Bank などもとのデータの出典へのリンクも含まれています。

5. Download ダウンロード

- グラフやデータをダウンロードすることができるようになっています。

6. Share 共有

- 共有リンクや、ホームページに埋め込む iframe リンクなどが付いています。

10.4.2 Covid-19 Data Explorer

リンク

日本を加えてあります。

10.4.3 Causes of Death

死亡原因についての分析のページです。

リンク<https://ourworldindata.org/causes-of-death>

10.4.4 Causes of death, World, 2019

10.4.5 Share of Deaths from major causes of Japan, 2020

10.5 API

- パッケージの公式サイト: <https://CRAN.R-project.org/package=owidR>
 - はじめにお読みください (ReadMe) : <https://cran.r-project.org/web/packages/owidR/readme/README.html>
 - マニュアル (Manual) : <https://cran.r-project.org/web/packages/owidR/owidR.pdf>
 - 使い方の例 (Vignette) : Create and Analyse a Dataset
- `owidR`: Import Data from Our World in Data

第 11 章

e-Stat 日本の政府統計

11.1 概要

基本的なデータベースは、「社会・人口統計体系」から見ることができます。一覧表示も可能ですが、まずは、リンクから階層形式で見ると、都道府県別データと、市町村別データに分かれています。また、それぞれが、基礎データと社会統計指標に分かれ、さらに、それぞれが、

11.2 データベース

- e-Stat：政府統計の総合窓口 - 統計で見る日本
- ダッシュボード：対話型形式でのデータの視覚化

たくさんの統計が掲載されています。ダッシュボードは、そのうちのいくつかを見やすいように、グラフによって視覚化されています。

まず、e-Stat 政府統計の総合窓口に行くと、右上の若葉マークに使い方が書かれていますから、それを読むのがお勧めです。ここには、詳細は書きません。

まず「すべて」（政府統計一覧）では、すべての統計情報を見ることができ、検索もできます。左の帯に、データベースと、ファイルとありますが。基本的に、API を使って、データを読み込むことができるようになっているのが、データベースです。

その下には、統計分類（大分類）として、トピックごとに分類されたものを調べるものがあります。これは、トップページの「分野」と大体対応しています。その下には、組織で調べる、統計の種類で絞り込み、政府統計名で絞り込み、提供周期で絞り込み、調査年で絞り込み、調査月で絞り込み、50 音で絞り込み、統計表フォーマットで絞り込み、集計地域区分で絞り込みと続きます。

まずは、統計の種類から、基幹統計を見てみましょう。ここからも見るすることができます。全部で 61 のデータベース、64 のファイルと書かれて、国勢調査や、人口推計、労働力調査、家計調査、学校基本調査などが並んでいます。

11.2.1 家計調査

家計調査をみてみましょう。家計調査についての説明がありますが、その下の、二人以上の世帯、年次を見てみます。費目分類 010 品目分類 (2020 年改定) (総数: 金額) の DB (データベース) をみてみましょう。

表が現れます。

左の方にある、表示項目選択を開きます。

時間軸: すべて解除して、最新の 2022 のものだけ選択します。

左の方にある、レイアウト設定を開きます。

ページ上部: 表章項目、世帯区分、時間軸 (年次)

列: 費目分類

行: 地域分類

確定させてもとに戻ると、それぞれの項目のそれぞれの地域の平均消費額がわかります。一番上には全国平均が載っています。

たとえば、二人以上の世帯の、この調査における平均世帯人数は、2.91 人、消費支出 3,490,383 円、牛乳の支出 15,001 円など出ています。

11.2.1.1 補足

統計表としてではありませんが、家計調査からのランキングのようなものも、公開されています。ご興味のあるかたはみてください。

家計調査 (二人以上の世帯) 品目別都道府県庁所在市及び政令指定都市 (※) ランキング (2020 年 (令和 2 年) ~ 2022 年 (令和 4 年) 平均)

ただし、このランキングの下になっているデータを探そうとすると、なかなか大変です。

11.3 ダッシュボード

何種類かのグラフを見ることができると同時に、右の方に、時系列表とあり、これから、データをダウンロードできるようにもなっています。

人口ピラミッド

人口ピラミッドを見てみましょう。1920 年からの人口ピラミッドを見ることができます。また、日本の全体だけではなく、都道府県、市町村、世界の国々についても見るができます。ピラミッドの形が変わっていく様子、戦争などで、急激に変化する様子なども、見るができます。

また、それぞれの個々のグラフのデータをダウンロードすることができます。全体のデータを一括ダウンロードできると良いと思うのですが。

11.4 API

11.5 estatapi 利用概要

CRAN estatapi URL: <https://CRAN.R-project.org/package=estatapi>

11.5.1 アプリケーション ID の設定

```
appId <- " " # 私のものは、英数40文字
```


第 12 章

SDGs

持続的開発目標（Sustainable Development Goals）関連の情報がまとまっているサイトの紹介とともに、そのダッシュボードの使い方などデータ関係を中心に書いていきます。

12.1 SDGs とは

持続可能な開発目標（SDGs：Sustainable Development Goals）とは、2001年に策定されたミレニアム開発目標（MDGs）の後継として、2015年9月の国連サミットで加盟国の全会一致で採択された「持続可能な開発のための2030アジェンダ」に記載された、2030年までに持続可能でよりよい世界を目指す国際目標です。17のゴール・169のターゲットから構成され、地球上の「誰一人取り残さない（leave no one behind）」ことを誓っています。SDGsは発展途上国のみならず、先進国自身が取り組むユニバーサル（普遍的）なものであり、日本としても積極的に取り組んでいます。（外務省ホームページ）

12.2 国際連合

17 Goals to Transform Our World: <https://www.un.org/sustainabledevelopment/>

SDGs HOME: <https://unstats.un.org/sdgs>

国連: <https://unstats.un.org/sdgs/dataportal/database>

12.3 世界銀行

世界銀行のダッシュボードとして紹介したデータバンク（DataBank）には、データベースとして、Sustainable Development Goals (SDGs) があります。下のリンクでは、そのデータベースがすでに選択されてあります。

SDGs DataBank: [https://databank.worldbank.org/source/sustainable-development-goals-\(sdgs\)](https://databank.worldbank.org/source/sustainable-development-goals-(sdgs))

全部で 408 の項目 (Series) に分かれて、データを見ることができます。

Data Topics of SDGs: <https://datatopics.worldbank.org/sdgatlas/>

17 の各項目についての状況がまとめられています。

12.4 OWID

データで見る私たちの世界：Our World in Data SGD Tracker: <https://sdg-tracker.org/>

17 のそれぞれの項目について

- 概要
- 数値目標と指標
- 簡単な解説など
- Chart 視覚化のためのダッシュボード

がっています。ダッシュボードについて少し書いておきます。

1. Chart グラフ

- いくつかの国についての年次変化を表す折れ線グラフなどがあります。
- Add Country or Region から国や地域を選択することができるようになっています。

2. Choropleth 色付き地図

- 右上のプルダウンメニューからは、地域（アフリカ、北アメリカ、南アメリカ、アジア、ヨーロッパ、オセアニア）を選択して拡大することができます。
- それぞれの国にマウスまたはポインターを重ねると、その国の状況を見ることができます。

3. Table 表

- データの一部が表となっています。降順・昇順などの並べ替えも可能です。

4. Sources 出典

- 変数の簡単な定義とともに、World Bank などもとのデータの出典へのリンクも含まれています。

5. Download ダウンロード

- グラフやデータをダウンロードすることができるようになっています。

6. Share 共有

- 共有リンクや、ホームページに埋め込む iframe リンクなどが付いています。

12.4.1 例：SDG1: **End poverty in all its forms everywhere**

リンク

12.4.2 例：SDG5: **Achieve gender equality and empower all women and girls**

リンク

12.5 SDG Index

SDG Report: <https://dashboards.sdgindex.org>

SDG Index: <https://dashboards.sdgindex.org/map>

Sustainable Development Report (SDGs 報告) を毎年出しています。17 の項目それぞれについての達成度を国別にも確認できるようになっており、改善されている、変化なし、悪化しているなどの状況も掲載されています。

報告全体を読むこともできますが、リンク上の地図で日本を選択すると、日本の状況を見ることができます。SDG1 から SDG17 までの項目について詳しくみていくことも可能です。

指標をどのように決めているかも気になりますが、それを理解するコースも公開しています。

SDGAcademyX: <https://www.edx.org/school/sdgacademyx>

- Measuring Sustainable Development

12.6 外務省

リンク：<https://www.mofa.go.jp/mofaj/gaiko/oda/sdgs/statistics/index.html>

17 項目についての説明があり、少し資料もついています。

第 13 章

その他のサイト

13.1 WIR

13.1.1 概要

13.1.2 データベース

13.1.3 ダッシュボード

13.1.4 API

13.2 EuroStat

13.2.1 概要

13.2.2 データベース

eurostat: <https://ec.europa.eu/eurostat>

13.2.3 ダッシュボード

13.2.4 API

第 II 部

第二部 R の基礎

第 14 章

R Studio で R

いよいよ、プログラミング言語 R を使って、データサイエンスを始めます。RStudio で R をすでに、使い始めておられる方も、ざっと読んでおかれることをお勧めします。R Markdown についてと、Posit Cloud というクラウドサービスの利用方法についても、述べてあります。

14.1 はじめに

この章では、R と R Studio をインストールし、正常に動作しているかを、確認することを目標とします。実際の、使い方は、次の章以下で、述べます。

また、Posit Cloud など、R Studio 以外で、R を使う方法も、最後に述べます。特に、Posit Cloud は、一定の制限はあるものの、基本的には、R Studio で R を使うのと、同様のことが可能な、Cloud 環境ですので、R と R Studio のインストールに成功した場合も、利用方法を確認してください。

確認の段階で、エラーが出てしまった場合の解決方法も、下に書きますが、まずは、Posit Cloud など、他の方法で R を使えるようにして、時間のあるときに、解決を試みてください。

注：この部分を書いている、2023年では、ほとんど、インストールについては、ほとんど問題がないようですが、Windows ユーザの、OneDrive 問題は、まだ、一部残っているようですので、確認していただくことをお勧めします。

14.2 R と R Studio

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS.
<https://www.r-project.org>

R は、無償で提供されている、統計解析ともに、データサイエンスにおいてたいせつな可視化のためのグラフを描写する環境でもあります。Windows、MacOS や、

Linux で利用することが可能です。

RStudio is an integrated development environment (IDE) for R and Python. It includes a console, syntax-highlighting editor that supports direct code execution, and tools for plotting, history, debugging, and workspace management. RStudio is available in open source and commercial editions and runs on the desktop (Windows, Mac, and Linux). <https://posit.co/products/open-source/rstudio/>

RStudio は、R と Python のための、総合開発環境 (IDE) です。RStudio には、プログラムを実行したり、制御やジョブ管理のための、コンソール (console)、コードを書いたり、実行したりする、文書の編集をする、エディター (Editor) とともに、グラフを表示したり、履歴や、プログラムを修正するなどのための、さまざまなツールが付属しています。RStudio はオープンソースで提供され、Windows、Mac および、Linux で利用可能で、有償版のサービスと無償版を提供しています。

R は、統計解析のためのシステムで、R Studio は、R (および Python) を利用するための、総合開発環境です。そこで、「R Studio で R を利用する」という表現をします。

14.3 R と R Studio のインストール

R と R Studio をインストールします。

両方とも、インストールすることが必要です。

今後のために、動作確認を、RNotebook を使って行います。

注意

以下は、自分のコンピュータ、すなわち、管理者権限があるコンピュータにインストールすることを前提に書きます。

家族で、コンピュータを共有していて、管理者権限がない場合は、管理者権限のあるかたに、R と RStudio をインストールしてもらってください。そうすれば、そのコンピュータの他のアカウントでも利用することができます。

ある制約がかかるとメッセージが出ますが、管理者権限なしで、インストールすることも可能です。特別の事情があり、管理者権限なしで、インストールする場合は、こちらを参照してください。

14.3.1 R のインストール

<https://cloud.r-project.org>

上のリンクから、Windows、macOS または、Linux を選択して、インストールしてください。

Windows の場合は、base - install R for the first time を選択してください。

macOS の場合は、M1, M2 など、最近の Apple Silicon の CPU で動くコンピュータか、以前の、Intel の CPU で動くものか、選択してください。Mac の左上の、りんごマークの、このコンピュータについてから、確認できます。

不明の場合は、「R のインストール」と検索してみてください。

14.3.2 R Studio のインストール

<http://www.rstudio.com/download>

上のリンクから、Windows 10/11 または、macOS 11+ を選択してください。これら以外の、古いシステムのコンピュータの場合は、下のサイトから、探してください。

<https://docs.posit.co/previous-versions/>

不明の場合は、「RStudio のインストール」と検索してみてください。

14.3.3 動作確認

動作確認のための、一連の流れを書き、その下に、問題が起こった場合の対処を書きます。

R Studio はこれまでも使っていたが、R Notebook は使ったことがないという方も、同様の確認をしてください。

1. RStudio を立ち上げます。
2. 上のメニューの File から、New Project を選択します。
3. Project の名前を (test とか、firstproject など) つけて、Create ボタンを押します。
4. 上のメニューの File から、New File の R Notebook を選択します。
 - Package が必要なので、Install するかと聞かれますから、インストールを選んでください*1。
5. File から、Save as を選んで、名前を (test0, rnotebook など) とつけ、Preview ボタンを押すと、内容が現れます。
6. Preview ボタンの右の、三角印を押すと、Knit PDF と現れますから、それを選択
 - すでに、TeX というシステムを使っておられる方は、PDF が作成されますが、それ以外の方は、なにやらメッセージが出て、TeX システムが無いと出ると言います。
 - 左下の、Console タブに、`tinytex::install_tinytex()` をコピーして、入力し、Enter または、Return キーを押して実行します。
 - もう一度、Knit PDF を (今度は、Preview ボタンが、Knit となっていると思います。その右の三角から Knit PDF を選択し) 押し実行します。

*1 次のステップ5でインストールする場合もあるかもしれません。

この一連の作業で、最後に、PDF が現れれば、動作確認終了です。クラウド Posit Cloud の項目に進んでください。

ステップ 5 まで終了すれば、PDF を作成しない限り、問題はありませんので、あとは、時間のあるときに、対応することも可能です。

14.3.4 トラブル・シューティング (1)

動作確認のステップ 4 (またはステップ 5 で要求されるパッケージのインストール時点) または、6 でエラーが出る場合があります。

- 動作確認のステップ 4 で、パッケージをたくさんインストールしますが、この段階で、エラーが出る場合があります。そのときは、まず、RStudio の上のメニューの、Tools の一番下の、Global Options を開き、左のメニューから、Packages を選び、Primary CRAN Repository を Change として、CRAN mirror から、Japan: The Institute of Statistical Mathematics, Tokyo を選択してください*2。
 - 解決しない場合は、下の、**解決しない場合は**に進んでください。
- 動作確認のステップ 4 では、問題なく、パッケージのインストールが完了したが、6 で、エラーが出た場合には、Windows の日本語アカウント名の問題の可能性が高いと思います。OneDrive の問題の可能性もないとは言えませんが、次を試してみてください。
 1. TinyTeX のアンインストール : `tinytex::uninstall_tinytex()` をコンソールで実行
 2. Windows の自分のアカウントからサインアウトし、サインインしなおす
 3. TinyTeX をディレクトリを指定してインストール : `tinytex::install_tinytex(dir = "C:/myTinyTeX")` をコンソールで実行
 4. サインアウトし、サインインしなおす
 5. 上の動作確認のステップ 6、R Notebook から knit PDF を利用して、PDF を作成する。
 - 解決しない場合は、下の、**解決しない場合は**に進んでください。
- **解決しない場合は**、下の囲みの中を読んでください。Windows の、OneDrive 関連の問題の可能性が高いかと思えます。
 - その後に書いてあるように、R と RStudio を一度、アンインストールします。(この作業はしなくても、おそらく問題ないと思いますが、安全のためにアンインストールします。)
 - もう一つ別の管理者権限のあるアカウントを (半角ローマ字名で) 下の指示に従って作成して、そのアカウントにサインインして、上の、1 から 6 を試して

*2 理由は不明ですが、特定のミラーサイトが理由ではなく、https を能動的に設定することで解決が図れるのかなと思います。

ください。

- 問題なく、1 から 6 が完了したら、しばらく、そちらのアカウントを利用してください。それ以降については、また下に書きます。
- 別のアカウントでも、問題が解決しない場合は、コンピュータに詳しい方に相談してください。わたしのホームページにある、メールアドレスから、わたしに相談するときは、次の三つの情報をコピー・ペーストして、教えてください。
 - 上の手順のどこでどのような問題が生じたか、そのときの、エラーメッセージ
 - `Sys.getenv()` の出力
 - Windows の場合、コマンド・プロンプトから、`systeminfo` と入力したときの出力

注意：Windows における既知の不具合について

`Sys.getenv('HOME')` と `Sys.getenv('R_LIBS_USER')` をそれぞれ、コピーして、RStudio 左下の窓枠の Console タブに、ペーストして、エンターして、???? や、カタカナ、漢字や、OneDrive という文字列が現れるか確認してください。`Sys.getenv()` とすれば、すべての環境変数を確認することもできます*3。

Windows の日本語システムで、アカウント名に日本語を使っておられる方、または、OneDrive を使っていて、Documents (書類) ディレクトリーのすべてをバックアップしておられる方は、ファイルを作成したり、パッケージをインストールするときに、問題が発生する可能性があります。

カタカナや漢字のユーザー名を使っている場合は、RMarkdown で、文書を作成し、PDF で出力するときには、問題が起こります*4。解決方法は上に書いたように、一つの解決方法として、TinyTeX を、Home とは異なるところにインストールしています*5。

OneDrive で、ドキュメントフォルダ全体のバックアップをとっている場合も、問題が起こる可能性があります。一般的には書けませんので、困ったときは、下に書く方法を参照してください*6。

14.3.4.1 トラブル・シューティング (2)

トラブル・シューティング (1) では解決できない場合の対処法を書きます。わたしは、MacOS では、動作確認の 1 から 6 で、問題が起こったと聞いたことがないので、以下では主として、Windows に限って書きます。

*3 エクスプローラからユーザ (C:\Users) を見ると漢字になっているかどうかの確認ができます。

*4 PDF の作成には、内部で TeX というプログラムを使っていますが、この TeX システムは、カタカナや漢字が入っているディレクトリには、インストールすることができません。

*5 別のアカウントを作成して、そこに、TeXLive をインストールする方法もあります。リンク を参照。

*6 パッケージのインストールや、R Markdown で文書を生成するときには、一時的なファイルを作成して、問題がある場合はレポートする仕組みになっています。そして、問題なくその作業が終了すると、それらのファイルを消去しています。詳細は不明ですが、この作業で、OneDrive でバックアップする作業が入ると、エラーが起きるのではないかと思います。

実際には、さまざまな理由がありますから、以下の方法では対処できないかもしれません。まず、エラーメッセージを丁寧にみるのが大切ですが、簡単には理解できない場合が多いかと思います。そこで、まず、試してみる方法を以下に書きます。

■14.3.4.1.1 R および RStudio のアンインストール どのようなアプリも、本来はインストールと同時に、アンインストール（アプリを削除）する方法を知っておくことは大切です。

- **Windows:** 設定から、アプリを選び、アンインストール（削除）したいものを選択して、実行。
- **Mac:** アプリケーションフォルダーから、アプリをみつけ、ゴミ箱へ。

備考：環境変数の Path について

上記の方法で、残ってしまうものがあります。それは、環境変数 Path の設定です。アンインストールなどをしたときは、一旦、サインアウトをして、もう一度、サインインしてから確認してください。

Mac は、システムのベースにあるのは、Linux などと同じ、Unix なので、アプリケーションの、ユーティリティ (Utilities) から、Terminal を立ち上げ、`echo $PATH` とすると、確認できます。RStudio の左下の窓枠にも Terminal があり、そこからでも、同じように確認できます。

Windows の場合は、対応するものには、コマンド・プロンプト (Command Prompt) と、パワーシェル (PowerShell) があります。初期設定^{*7}では、RStudio の左下の窓枠には、コマンド・プロンプトが出ていると思います。そこで、`echo %PATH%` とします。PowerShell の場合は `echo $Env:Path` とすれば、表示できます。

Windows の場合、「設定」から、「環境変数の編集」を検索して、選択すると、Path を見ることができます。上の枠が、ユーザの環境変数、下の枠が、システム環境変数です。ユーザの環境変数の中の、Path（他には、TEMP, TMP もあるかと思います。）をコピーしておくといいでしょう。Path をダブルクリックして、編集することも可能です。ただし、十分理解しないで、Path を変更すると、問題が起こる可能性がありますので、注意してください。

■14.3.4.1.2 別のアカウントを作成してインストール Windows に、もう一つ、アカウントを作成します。

1. [スタート] > [設定] > [アカウント] の順に選択し、[ファミリーとその他のユーザー] を選択し、[アカウントの追加] から作成します。

^{*7} Tools から Global Option を選択して変更することも可能です。

2. どのようにサインインしますかと出ますから、下にある、「このユーザーのサインイン情報がありません」を選択します。[次へ]
3. つぎに、アカウントを作成しようとしてますから、特に、なにも入力せず、「Microsoft アカウントを持たないユーザーを追加する」を選択します。[次へ]
4. アカウント設定、名前とパスワード、名前は、ローマ字と数字の半角で作成。パスワードは2回入れる。アカウントの名前と、パスワードは記録しておくこと。
5. その下の三つの質問とその答えを記入します。これも、おぼられるものにするか、記録すること。[次へ]
6. アカウントが作成されるが、そのアカウントを選択すると、「アカウントの種類の変更」が選択できるので、「管理者」に変更。[OK] をクリックして終了。

アカウント名をローマ字にします。このアカウントから、インストールしてください。元の日本語アカウントとファイルを共有したいときは、C ドライブのユーザーにあるパブリック（共有）ディレクトリーにコピーしてください。

■14.3.4.1.3 このあとの使い方について 他のアカウントに、R と RStudio をインストールしても、自分のアカウントでも、R と RStudio を利用することが可能です。しかし、漢字やカタカナのアカウント名や、OneDrive の問題は、解決しません。アカウント名を修正することは可能ですが、間違えると、コンピュータ自体が動かなくなりますから、ここでは、説明しませんが、興味のあるかたは、矢内勇生さんのスライドを参照してください。

OneDrive は、どのように使われているかによって、対応が変わりますので、一般的な対応方法は書けません。OneDrive は、Microsoft が提供するクラウドサービスで、5GB までは、無料で使えることもあり、コンピュータを使い始めるときに*⁸、利用を促されます。5GB を超える場合は、課金されます。十分な容量を有料で確保していると、すべて、OneDrive にバックアップすることになり、Home も、OneDrive の管理下に置かれて、上に述べたような、ファイルを書いたり消去したりを頻繁に行うプログラムの場合には、問題が起こるようです。

ただし、バックアップを取ることは、とても有効なことです。設定を変える場合は、十分検討してから行ってください。

個人的には、簡単設定で設定される、デスクトップと、ドキュメントと、フォトのうち、ドキュメントについては、OneDrive バックアップを停止し、ドキュメントの中に、バックアップ用のフォルダーを作成し、その中のもののみ、バックアップするのが良いのではないかと思います。それには、ドキュメントのバックアップを外し、OneDrive の中に、フォルダーを作成して、そこに、バックアップするファイルを移動させます。しかし、使い

*⁸ Microsoft アカウントを設定すると

方などによって、状況が異なりますので、コンピュータに詳しいかたに相談することをお勧めします。

その上で、RStudio のプロジェクトを作成するときは、OneDrive でバックアップをとっていないフォルダーに作成します。

詳細は、下のリンクなどを参照してください。

参照リンク

14.4 クラウド - Posit Cloud

Posit (RStudio) Cloud は、誰でもオンラインでデータサイエンスを行い、共有し、教え、学ぶことができる、軽量でクラウドベースのソリューションです。2022 年 11 月に、会社名が、RStudio から Posit に変更になったので、Posit Cloud となっていますが、まだ、RStudio Cloud と表示されている箇所もあります。

14.4.1 クラウドサービス How to Start Posit Cloud

まず、サインアップして、使ってください。一ヶ月の利用時間の限度など、設定されていますが、どこからでも、インターネットにつながっていれば使えるので、わたしは、いつかアカウントを持って、活用しています。

1. Posit Cloud にアクセスします。Go to <https://posit.cloud/>
2. Get Started または、右上から、Sign UP を選択します。Sign Up: top right
3. Email address or Google account
4. New Project: Project Name

特徴：制限など (Key Features)

- プロジェクト数の上限は 50。Up to 50 projects total
- 共有スペースは一つのみ (5 人までのメンバーが 10 個のプロジェクトまで共有できます) 1 shared space (5 members and 10 projects max)
- コンピュータ時間としては月間最大 25 時間*⁹。25 compute hours per month
- 各プロジェクトについて最大 1GB の RAM (Read Access Memory)。Up to 1 GB RAM per project
- 各プロジェクトについて 1 CPU。Up to 1 CPU per project
- 背後で実行する作業は 1 時間が上限。Up to 1 hour background execution time

大学で課題などに取り組むと、まず、月間 25 時間の制限に引っかかり、次のようなメッセージが表示されます。

Your account exceeded its compute hour limit. You can continue to open projects in your account until *such and such time*, or until you have used at total of 40 compute

*⁹ 右の i マークを押すと詳細な条件を確認できます。

hours. After that you will be unable to open projects in space owned by your account until your next usage period begins on *such and such day*.

わたしは、このようなメッセージが出たら、重要度にもよりますが、まずは、プロジェクトをダウンロードし、いつでも、自分のコンピュータで使えるようにしてから、他の、アカウントに引っ越して、作業を続けています。その方法を追記しておきます。

TIPs :

プロジェクトをダウンロード :

プロジェクトのダウンロード : 自分のコンピュータに R と R Studio が使える場合は、必要なプロジェクトをダウンロードします。ダウンロードするには、左上の 3 本線から、自分の Workspace にもどり、プロジェクトの右についている、ダウンロードボタンを押します。これは、プロジェクトとして、RStudio から開くことができます。

If you have installed R Studio and R on your computer, From the three lines on the top left, go back to your workspace. Then there is a download button on the right of your project. You can open it on your computer.

他のアカウントとの共有 :

1. 異なる電子メールアドレスで別の Posit Cloud アカウントを作成します。(create another Posit Cloud account with another email address.)
2. 使っていたプロジェクトを開き、右上のギアマークから、Change Access を、You から、everyone に変更します。(In your current project, open the gear mark on the top right, and change access from You to everyone.)
3. 使っていたプロジェクトのアカウント名の隣にある、三つの点が丸で囲まれているものを選び、Share Link を、別のアカウントを作成したアドレスに送ります。(In your current project, next to your account name, there is a circle with three dots. Share the link and send the email to another account.)
4. 使っていた、元のアカウントからログアウトし、新しいアカウントを開き、送られてきたアドレスをクリック、またはコードを入れると、使っていたプロジェクトが開き、Temporary Copy と出ます。編集して使いたい時は、それを Permanent Copy にすると、新しいアカウントで編集し、使うことができます。(Then, you can get an access code to your current project. Log out from the existing account, log in to your new account, and then use the link. Then you can see the temporary file containing everything in your old account and use it for a permanent project.)

14.4.2 PositCloud Shared Project

わたしのプロジェクトを一つ共有してありますので、興味のある方はご覧ください、

1. まず、自分の、PositCloud アカウントにログインする。
2. リンク をクリック

3. 上に、Temporary Copy と出ています。このままでの、右下の、窓枠から、Files を選び、開いて中身を見たり、Code Chunk を実行したり、Preview や、knit をすることも可能です。Try Again と表示されますが、それをクリックすると、表示されます。R Markdown などの詳細は、R Markdown の章をみてください。
4. 上に、Save a Permanent Copy とあります。これを、クリックすると、自分の、プロジェクトとすることができ、編集も可能になります。

14.4.3 Posit Primers and Cheat Sheet

Posit Primers の左の帯（表示されていない場合は、左上の 3 本線をクリックしてください）をみると、いろいろな機能があることがわかります。特に、Primers は、基本的な練習をするプログラムで、あとからも、紹介しますが、答えを入力している形の練習問題になっています。とても有用です。

Cheat Sheet は、いくつもの、機能の一覧が、早見表の形式で載っています。使い方などをぼえておくことはできないので、わたしも、よく使っています。

14.5 R のその他の利用方法

R Studio または、RStudio Cloud (Posit Cloud) 以外で、R を使われる方のために、少しだけ書いておきます。個人的には、Google colab と、Cocalc を利用しています。

Google colab は、Google アカウントの作成、Cocalc は、Cocalc アカウントの作成、または、Google アカウントか、GitHub アカウントのリンクが必要です。

Google アカウントをお持ちの方は多いと思うので、Google colab について、最低限のことのみ、書いておきます。

14.5.1 Google colab で R

基本的に、python 開発環境として構築されているものですが、R でも使うことができます。

1. Google アカウントにログインします。
2. ここ をクリックして起動します。
3. 一番上に、ノートブック名が Untitled0.ipynb などと表示されますから、適当に変更します。
4. +コード、+テキスト とあり、最初のコードの 1 行が表示されていますから、たとえば、head(cars) と入れて、左の三角を押します。すると、最初だけ少し時間がかかりますが、その下に結果がでます。
5. 次に、上や、最後の行の直下に、表示される、+コード、+テキストをクリックして、あたらしい、コード・チャンクか、テキスト・チャンクを書き入れていきます。
6. tidyverse などは、すでにインストールされていますが、使いたいと

きは、`library(tidyverse)` とし、インストールされていないときは、`install.packages("WDI")` などとします。

ノートを、保存、印刷、ダウンロードなど可能です。

フォルダーを作成して、外部ファイルを読み込んだり、書き出したりすることも可能です。

14.5.1.1 参考

- How to use R in Google Colab:

14.5.2 CoCalc で R と RMarkdown

Cocalc リンク

無償版は、スピードが遅いですが、個人的には、有用なクラウドサービスとして、活用しています。グループでプロジェクトを共有することも簡単にできます。

1. 最初は、上のリンクの下にある、SignUp から、アカウントを作成します。
 - Email アドレス、または、Google などのアカウントにリンクさせて、アカウントを作成することが可能です。
2. Create a New Project から、プロジェクトを作成します。
3. 使い方
 - New から、Sage Worksheet を選択して、`%default_mode r` と最初書けば、その後、R のコマンドを実行できます。
 - New から、RMarkdown を選択すると、RMarkdown 文書を作成することができます。
 - 他にも、TeX や、Python Notebook (Jupyter Notebook) など、さまざまな道具が提供されています。
 - Maxima と呼ばれる、Maathematica の Free のバージョンから改良した、SageMath からスタートしたサイトです。

第 15 章

R Markdown

データサイエンスを実際に行っていくときには、再現性のためにも記録が大切ですが、そのためのツールである、R Markdown について説明します。2022 年頃から、R だけでなく、Python や Julia など複数の言語上で利用する目的で、R Markdown の後継の Quarto という新しいシステムも使われています。少しずつ、Quarto についても紹介していきますが、R Markdown を知っていれば、簡単に、乗り換えることができ、R Markdown も今後ともサポートされるとのことですので、ここでは、R Markdown について書き、後ほど、別立てで、Quarto について書いていくことにします。

15.1 再現可能性とプログラムの文書化

データサイエンスは、サイエンス（科学）ということばもついています。特に、根拠に基づいた（evidence based）とか、データに基づいた（data based）ということばを使うときには、なおさら、再現可能性（reproducibility）や、コードの内容の説明などのコミュニケーションにも注力する必要があります。このことを心がけて、データサイエンスを学んでいきましょう。

表題にある、“Reproducible and Literate Programming” は、Reproducible（再現可能）かつ、Literate な（理解できるように記述した）Program（プログラム・コード）を共有することをたいせつにしましょうということです。

注：Literate Programming の訳は「文芸的プログラミング」とか「文学的プログラミング」が使われているようですが、読んで理解できるということですから、文書化ぐらいがいいかなと思っています。より適切な訳があれば、今後変更していきたいと思います。

15.1.1 目的、問いなど

プロジェクトの目的、問いなどは、途中で変わっていくこともありますが、その都度に、メモをしておくといいでしょう。

15.1.2 データについて

どのようなデータをどのように取得してきたかを、記録し、伝えられるようにすることが、必要です。データを取得するときから、取得方法や、それを伝える方法にも常に気をつけましょう。

15.1.3 コードについて

どのようなコードでそのグラフ (chart) などが得られたかも、単にコードを記述するだけでなく、それぞれの部分に、説明を付与することも有効です。

15.1.4 グラフについて

視覚化 (visualization) は、とても有効です。そこで、見て理解したこと、観察したこと (observations) などは、簡単でも構いませんから、必ず、記録しておきましょう。

15.1.5 まとめ：R Markdown の目的

まさに、このようなことを可能にするのが、R Markdown です。少しずつ学んでいきましょう。

15.2 準備：パッケージのインストール

R パッケージは、R の拡張機能で、コード、データ、ドキュメントを標準化されたコレクション形式で含んでおり、標準的なものは、R Studio の Top Bar の Tool > Install Packages からインストールできます。

- tidyverse
- rmarkdown
- tinytex

インストールを複数回しても問題はありませんが、インストールされているかどうかは、Packages タブから確認することができます。

インストールは一回だけ。ときどき、Tools > Check for Package Update をつかって、Update しておくといいでしょう。

15.3 R Notebook

R Markdown はデータサイエンスのためのオーサリングフレームワーク。

コード（プログラム）とその実行結果、を記録・表示し、高品質のレポートの作成を可能にします。

R Notebook は、独立してインタラクティブに実行できるチャンクを持つ R Markdown ドキュメントの一つの形式で、入力のすぐ下に出力が表示することができます。

1. File > New File > R Notebook
2. Save with a file name, say, test-notebook
3. Preview by [Preview] button
4. Run Code Chunk `plot(cars)` and then Preview again.

15.4 日本語のテンプレート

下のリンクを開き、右上の Code ボタンから、Download Rmd を選択すると、ダウンロードできますから、ダウンロードしたものを、プロジェクト・フォルダーに移動またはコピーしてください。ダウンロードできないときは、Ctrl を押しながら、Download Rmd をクリックすると、Save As で保存できると思います。ブラウザによって仕様が異なりますから、適切な方法を選んでください。

- <https://ds-sl.github.io/intro2r/RNotebook-J.nb.html>
- <https://ds-sl.github.io/intro2r/Rmarkdown-J.nb.html>

Windows でも、Mac でも提供されている、Google Chrome の場合には、Code ボタンから、ダウンロードされるはずですが。

RNotebook の新しいファイルを作成し、下のサイトを表示させて、コピー・ペーストで、書き換えることも可能です。

- <https://ds-sl.github.io/intro2r/RNotebook-J.html>
- <https://ds-sl.github.io/intro2r/Rmarkdown-J.html>

15.5 R Markdown いくつかの Output

```
title: "Testing R Markdown Formats"
```

```
author: "ID Your Name"
```

```
header-includes:
```

```
- \usepackage{ctex}
```

```
output:
```

```
html_notebook: default
```

```
html_document: default
```

```
pdf_document: default
```

```
  latex_engine: xelatex
```

```
word_document: default
```

```
powerpoint_presentation: default
ioslides_presentation: default
---
```

PDF でエラー? コンソールで `tinytex::install_tinytex()`

- TeX システムがインストールされている場合は不要

エラーの例を書いておきます。 `tinytex::install_tinytex()` おらず、他の TeX システムもインストールしていない環境で、上に引用した、RNotebook-J から、PDF を作成したときに生じたエラーです。ここに

If you are not sure, you may install TinyTeX in R: `tinytex::install_tinytex()`

「よくわからない場合は、R で TinyTeX をインストールすることもできます:
`tinytex::install_tinytex()`」

と書いてあります。

エラーメッセージの例

```
processing file: RNotebook-J.Rmd
```

```
"C:/Program Files/RStudio/bin/quarto/bin/tools/pandoc" +RTS -K512m -RTS RNotebook
```

```
Error: LaTeX failed to compile RNotebook-J.tex. See https://yihui.org/tinytex/r/#
```

```
No LaTeX installation detected (LaTeX is required to create PDF output). You should
```

```
If you are not sure, you may install TinyTeX in R: tinytex::install_tinytex()
```

```
Otherwise consider MiKTeX on Windows - http://miktex.org
```

```
MacTeX on macOS - https://tug.org/mactex/ (NOTE: Download with Safari rather than
```

```
Linux: Use system package manager
```

15.6 YouTube Video - rmarkdown

15.6.1 RMarkdown で PDF を作成するときの注意

RMarkdown では、`tinytex` パッケージというとても素晴らしいものが提供されています。これは、TeXLive というシステムを扱うためのものですが、LiveTeX 全体は、3GB 程度ありますが、非常に小さなファイルで、必要なものだけ、導入できるようになっています。

TeX システムを、使われる方は、TeXLive が、インストールされているかと思いたすので、それを使うことが狩野樹になっています。

一応、TeXLive がインストールされていて、そのあとに、TinyTeX をインストールしようとすると、拒否されますが、インストール、アンインストールなどをしていると、Path も変更になるため、問題が起こり、PDF が作成できなくなる場合もあります。

その場合の対処も含めて、下に書きます。

アカウントに漢字やカタカナが入っていると、PDF が作成できません。

15.6.1.1 tinytex によるインストール方法

- `tinytex::install_tinytex(dir = "C:/TTeX")`
- 環境変数の設定：
 - `Path=C:**UserName**\AppData\Local\Microsoft\WindowsApps;C:\TTeX\bin\windows`
 - `UserName` の部分は、ご自分のユーザネームに変更 は日本語システムでは、`¥` と表示されているかもしれません。

これで解決しない場合は、Windows に、もう一つ、アカウントを作成し^{*1}、アカウント名を英語にして、そちらから、インストールしてください。元の日本語アカウントとファイルを共有したいときは、共有ディレクトリーにコピーしてください。

すべての状況は確認できませんので、ここまでとします^{*2}。Chat GPT に確認したやりとりは、ここにあります。自分で確認をして、HOME の変更などをしていても良いですが、問題が生じた時に、サポートできませんから、一般的な方法としては、書かないことにします。

注：矢内勇生さんのサイトには、詳細な説明があります。

*1 [スタート] > [設定] > [アカウント] の順に選択し、[ファミリーとその他のユーザー] を選択し、[アカウントの追加] から作成。

*2 元のアカウントから、利用したり、Home を変更したりなど、いろいろな方法で、解決することも可能ですが、自信がない場合には、上の方法で、別のアカウントから、利用してください。

第 16 章

base R

16.1 はじめに

R を使って、データサイエンスをはじめる最初のステップについて説明します。また、R を起動すると、同時に読み込まれる、base と呼ばれるパッケージについても簡単に解説します。

本書では、Tidyverse と呼ばれるパッケージ群を主として使いますが、基礎を支えるのは、base ですので、Tidyverse でデータサイエンスをするため、最低限知っておいていただきたい、base についての説明という感じでしょうか。

R Project に基本文書がありますが、100 ページを越しますので、参考程度にさせていただくのがよいかと思えます。

リンク：

- An Introduction to R Manual にあります。下に日本語版へのリンクもあります。
- CheatSheet base R 早見表^{*1}。検索すると日本語版も見つかると思えます。
 - Posit CheatSheets: Posit co の提供する公式早見表
 - Contributed CheatSheet at Posit: Posit co のサイトからリンクのある早見表

16.2 プロジェクト - Project

RStudio で R を利用する場合には、プロジェクトを作成することを強く勧めます。

1. まず、R Studio を起動します。
2. 上のメニューの、File から、New Project を選択します。New Directory (新しいディレクトリー) を選択し、プロジェクトを作成する Directory を決めて、名前を

^{*1} CheatSheet は、直訳は、カンニング・ペーパーです。ここでは、もう少し品のよさそうな、早見表ということばを使いました。

つけます。その名前が、プロジェクト名になります。

- Directory (フォルダー) を指定してその名前をつけて、プロジェクトを作成します。
 - Directory が階層に分かれているときは、どこに作成するかを選択してから、名前をつけて、作成します。
3. 一旦、R Studio を終了してみましょう。
 4. プロジェクトの起動には、いくつかの方法があります。
 - まず、R Studio を起動。一つしかプロジェクトがない場合は、そのプロジェクトが起動すると思います。上に、プロジェクト名が掲載されていれば、問題ありません。
 - File から、Open Project を選択し、起動したい、プロジェクトの Directory (フォルダー) を選択して起動します。
 - File から、Recent Project (最近使ったプロジェクト) を選択すると、プロジェクト名が表示されますから、選択すると起動することができます。
 - コンピュータのプロジェクト入っているディレクトリー (フォルダー) をさがし、そこに、プロジェクト名.Rproj とあるものを見つけて、それを開くと、そのプロジェクトが起動します。
 5. 作業後は、保存しますかと聞かれますから、保存して終了してください。

16.3 コンソールで実行 - Run in Console

プログラム (コード) の実行には、いくつかの方法がありますが、一番、基本的な、コンソール (Console) での実行について、説明します。Console は、R Studio の左下にあります。(左の枠が一つになっているかもしれません。その一番左のタブが Console です。選択されていない場合は、Console を選択してください。)

16.3.1 最初の四つ

下の、四つを、一つずつ、一番下の、> マークの次には書き (または、コピー・ペーストして) Return または、Enter キーを押してください。実行結果が、その下に出ます。最後の、`plot(cars)` は、`cars` というデータの、散布図が右下の、Plots タブに表示されます。

- `head(cars)`
- `str(cars)`
- `summary(cars)`
- `plot(cars)`

エラーが表示されたら、もう一度、スペルを確認して、入力してみてください。

次のような、結果が表示されると思います。簡単な説明をつけます。

```
head(cars)
#>   speed dist
#> 1     4    2
#> 2     4   10
#> 3     7    4
#> 4     7   22
#> 5     8   16
#> 6     9   10
```

`head(cars)` は、`cars` という、R に付属している、データの、最初 (頭 `head`) の 6 行を、表示します。

```
str(cars)
#> 'data.frame':   50 obs. of  2 variables:
#>  $ speed: num  4 4 7 7 8 9 10 10 10 11 ...
#>  $ dist : num  2 10 4 22 16 10 18 26 34 17 ...
```

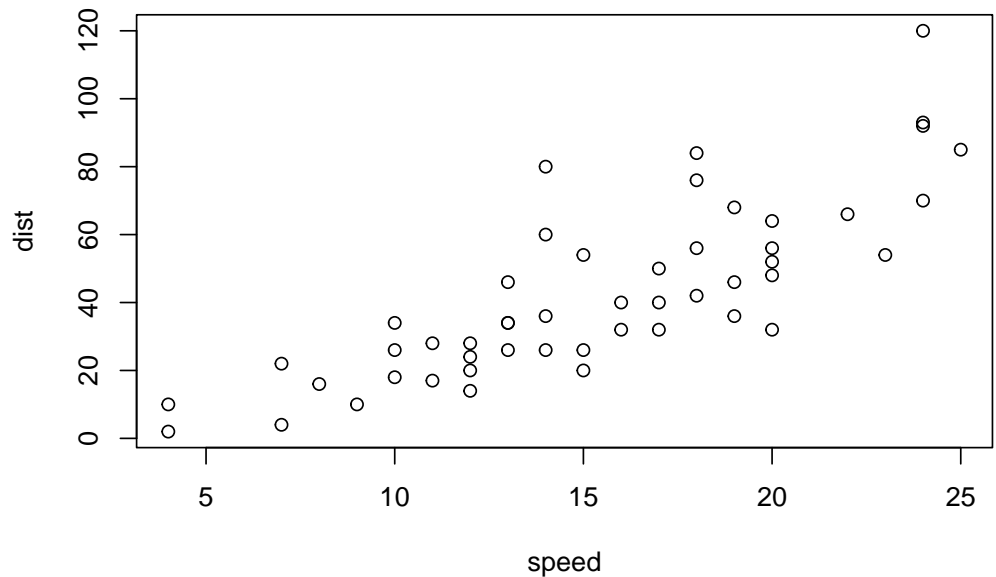
`str(cars)` は、`cars` という、R に付属している、データの構造 (structure) を表示します。 `data.frame` とありますが、これは、矩形になったデータ (各列の長さがおなじ) の一番簡単なクラスの名前で、2 変数、それぞれが、50 個の数値データ (numerical data) からなっていることがわかります。

`head(cars)` では、縦に表示されていたものが、横に表示されています。`$speed`、`$dist` とありますが、`cars$speed`、`cars$dist` は、`cars` データの、それぞれの列を意味します。

```
summary(cars)
#>      speed          dist
#> Min.   : 4.0    Min.    : 2.00
#> 1st Qu.:12.0   1st Qu.: 26.00
#> Median :15.0   Median  : 36.00
#> Mean   :15.4   Mean    : 42.98
#> 3rd Qu.:19.0   3rd Qu.: 56.00
#> Max.   :25.0   Max.    :120.00
```

`cars` データの概要 (`summary`) が表示されます。各列 (変数) について、最小値 (Minimum)、小さい方から、4 分の 1 を切り捨てたときの最小の値 (1st Quartile)、中央値 (Median)、平均 (Mean)、大きい方から、4 分の 1 を切り捨てたときの最大の値 (3rd Quartile)、最大値 (Maximum) が表示されます。

```
plot(cars)
```



右下の、窓枠の、Plots に、上のグラフ（散布図）が表示されると思います。Export と書いてある、プルダウンメニューがあり、そこから、画像として保存することも、可能です。

以前は、このように取り出した画像を、Word などに貼り付けて、使っていました。現在でも、そのような方法を知っていることは有効だと思います。

16.3.2 アサインメント、ヘルプ

コンソールで次のそれぞれを、試してみてください。

- `df <- cars`

`df` に、`cars` をアサインします。すなわち、`df` が、`cars` の内容に置き換わります。`cars` はデータですが、データを含む、オブジェクトの名前を設定するためにも使います。オブジェクト名は、英文字から始まれば、かなりの自由度がありますが、わたしは、英文字と数字と `_` (underscore) 程度しか使わないようにしています。

- `head(df)`

`head(df)` は、`head(cars)` と同じ出力が得られます。

- `View(cars)`

左上の、窓枠が開き、`cars` というデータの内容が表示されます。列名のところには、三角形も表示され、それをうけると、大きい順、小さい順などに、並び替えることも可能です。また、フィルター機能も使えます。

- `?cars`

右下の、窓枠の Help タブに、`cars` の情報が表示されます。Help タブにある、虫眼鏡がついた、検索窓 (search window) に、`cars` と入れても、同じ結果が得られます。内容を確認してください。

一番上には `cars {datasets}` とありますが、これは、`datasets` というパッケージの、`cars` だという意味です。そこで、`datasets` を調べてみましょう。

- `?datasets`

“The R Datasets Package” だと書かれていて、さらに、

This package contains a variety of datasets. For a complete list, use `library(help = "datasets")`.

さまざまなデータが含まれています。全てのリストをみるには、`library(help = "datasets")` を使ってください。

とありますから、`library(help = "datasets")` をコンソールに入力してみてください。

- `library(help = "datasets")`

左上の窓枠に、リストが表示されます。古いデータばかりですが、例として使うには、十分すぎるぐらいの、数のデータがあります。これらは、Toy Data (おもちゃのデータ) と呼ばれることもあります。

`cars` も見つかりましたか。

16.3.3 おすすめ

コンピュータのシステムが、日本語であると、R の言語も日本語になっているはずですが、そこで、エラーが発生すると、一部、日本語で表示されます。しかし、ネット上などで、そのエラーの対応を検索するときは、英語のエラーメッセージで検索した方が、解決方法が得られる可能性が圧倒的に高いので、わたしは、英語に設定しています。英語にするには、Console で次のようにします。

言語を英語に設定：`Sys.setenv(LANG = "en")`

RStudio を終了して、もう一度起動すると、日本語に戻っていると思います。ですから、作業の最初、または、エラーが出たら、変更することをお勧めします。

日本語に戻りたいときは、次のようにします。

言語を日本語に設定：`Sys.setenv(LANG = "ja")`

さまざまな Help など、すべて日本語で表示されれば日本語を使うのは有効かもしれませんが、すくなくとも、現在は、そうではないので、上に説明したことから、英語に設定することをお勧めします。

16.3.4 練習

1. `head(cars, 10L)` は何が出力されますか。 `head(cars, n=10L)` と同じですか。

2. `?head` または、Help の検索窓に `head` と入力して、説明を見てみてください。
`head(cars, n=10L)` などについて、書いてありましたか。他には、どのようなことが分かりましたか。
3. `datasets` のデータのいくつかについて、そのデータの `help` や、`head`, `str`, `summary` などを使ってみてください。これらで表示できない場合はありますか。データについては、最初に、これら、三つを試してみることをお勧めします。わかったことをメモしておくといいでしょう。`datasets` のリストをみるには、`library(help = "datasets")` でしたね。

16.4 RStudio について

RStudio は多くの機能を持っています。

16.4.1 四つの窓枠とタブ Four Panes and Tabs

- 左上 (Top Left) : スクリプトや文書、データなどの編集 (Source Editor)
- 右上 (Top Right) : 環境変数 (Environment) , 履歴 (History) など (etc.)
- 左下 (Bottom Left) : コードの実行・実行結果などを表示するコンソール (Console) , コンピュータシステムの端末 (Terminal) , 文書の機械語翻訳 (Render) , 背後での作業 (Background Jobs)
- 右下 (Bottom Right) : ファイル (Files) , 描画 (Plots) , パッケージ (Packages) , ヘルプ (Help) , 文書などの表示窓 (Viewer) , R Markdown の HTML, PDF 表示 (Presentation*²)

16.5 R Script 実行記録

R Script を使って、コードを実行すると、その記録を残すことができます。

16.5.1 R Script の作成

- RStudio の上のメニュー・バーから `File > New File > R Script` を選択します。
- `File > Save As` で、名前をつけて保存します。 `{file_name}.R` が作成されます。
– 右下の、Files から、ファイルを確認してください。
- `head(cars)`, `str(cars)`, `summary(cars)`, `plot(cars)` などと改行をしながらコードを書きます。
- 実行するには、カーソルの場所で `Ctrl+Shift+Enter` (Win) または `Cmd+Shift+Enter` (Mac) とすると、カーソルのある行か、その下の行で、最初のコードが実行されます。

*² Viewer への表示を使っており、Presentation への表示を使っておらず不明

- R Script エディターの上にある、Run ボタンを押しても、同様に実行されま
す。
- Run ボタンの右の、Source ボタンを押すと、そのスクリプトの、最初からす
べてが実行されます。
- 最後には保存しておきましょう。たとえば、myfirstscript などとすると、File
のところに、myfirstscript.R というファイルができていることを確認できます。

16.5.2 R Script による実行

新しく、R Script を作成し、この下の、コード（ハイライトされている部分）をコピー・
ペーストして、保存し、実行してみてください。

それぞれ、どのようなことをしているのでしょうか。

詳細を説明すると、時間もかかりますから、必要になったときに、少しずつ説明してい
きます。ここでは、スクリプトを使って、実行するという、二つ目の方法に慣れていただ
ければ十分です。

16.5.2.1 スクリプト 1: basics.R

```
#####  
#  
# basics.R  
#  
#####  
# 'Quick R' by DataCamp may be a handy reference:  
#   https://www.statmethods.net/management/index.html  
# Cheat Sheet at RStudio: https://www.rstudio.com/resources/cheatsheets/  
# Base R Cheat Sheet: https://github.com/rstudio/cheatsheets/raw/main/base-r.pdf  
# To execute the line: Control + Enter (Window and Linux), Command + Enter (Mac)  
## try your experiments on the console  
  
## calculator  
  
3 + 7  
  
### +, -, *, /, ^ (or **), %, %/  
  
3 + 10 / 2  
  
3^2  
  
2^3
```

```
2*2*2

### assignment: <-, (=  
>, ->, assign())

x <- 5

x

#### object_name <- value, '<-'  
shortcut: Alt (option) + '-' (hyphen or minus)  
#### Object names must start with a letter and can only contain letter, numbers,  


this_is_a_long_name <- 5^3

this_is_a_long_name

char_name <- "What is your name?"

char_name

#### Use 'tab completion' and 'up arrow'

### ls(): list of all assignments

ls()
ls.str()

#### check Environment in the upper right pane

### (atomic) vectors

5:10

a <- seq(5,10)

a

b <- 5:10

identical(a,b)

seq(5,10,2) # same as seq(from = 5, to = 10, by = 2)
```

```
c1 <- seq(0,100, by = 10)

c2 <- seq(0,100, length.out = 10)

c1

c2

length(c1)

#### ? seq    ? length    ? identical

(die <- 1:6)

zero_one <- c(0,1) # same as 0:1

die + zero_one # c(1,2,3,4,5,6) + c(0,1). re-use

d1 <- rep(1:3,2) # repeat

d1

die == d1

d2 <- as.character(die == d1)

d2

d3 <- as.numeric(die == d1)

d3

### class() for class and typeof() for mode
### class of vectors: numeric, characters, logical
### types of vectors: doubles, integers, characters, logicals (complex and raw)

typeof(d1); class(d1)

typeof(d2); class(d2)
```

```
typeof(d3); class(d3)

sqrt(2)

sqrt(2)^2

sqrt(2)^2 - 2

typeof(sqrt(2))

typeof(2)

typeof(2L)

5 == c(5)

length(5)

### Subsetting

(A_Z <- LETTERS)

A_F <- A_Z[1:6]

A_F

A_F[3]

A_F[c(3,5)]

large <- die > 3

large

even <- die %in% c(2,4,6)

even

A_F[large]

A_F[even]
```

```
A_F[die < 4]

### Compare df with df1 <- data.frame(number = die, alphabet = A_F)
df <- data.frame(number = die, alphabet = A_F, stringsAsFactors = FALSE)

df

df$number

df$alphabet

df[3,2]

df[4,1]

df[1]

class(df[1])

class(df[[1]])

identical(df[[1]], die)

identical(df[1],die)

#####
# The First Example
#####

plot(cars)

# Help

? cars

# cars is in the 'datasets' package

data()

# help(cars) does the same as ? cars
# You can use Help tab in the right bottom pane
```

```
help(plot)
? par

head(cars)

str(cars)

summary(cars)

x <- cars$speed
y <- cars$dist

min(x)
mean(x)
quantile(x)

plot(cars)

abline(lm(cars$dist ~ cars$speed))

summary(lm(cars$dist ~ cars$speed))

boxplot(cars)

hist(cars$speed)
hist(cars$dist)
hist(cars$dist, breaks = seq(0,120, 10))
```

16.5.2.2 スクリプト 2: coronavirus.R

```
# https://coronavirus.jhu.edu/map.html
# JHU Covid-19 global time series data
# See R package coronavirus at: https://github.com/RamiKrispin/coronavirus
# Data taken from: https://github.com/RamiKrispin/coronavirus/tree/master/csv
# Last Updated
Sys.Date()

## Download and read csv (comma separated value) file
coronavirus <- read.csv("https://github.com/RamiKrispin/coronavirus/raw/master/cs
# write.csv(coronavirus, "data/coronavirus.csv")
```

```
## Summaries and structures of the data
head(coronavirus)
str(coronavirus)
coronavirus$date <- as.Date(coronavirus$date)
str(coronavirus)

range(coronavirus$date)
unique(coronavirus$country)
unique(coronavirus$type)

## Set Country
COUNTRY <- "Japan"
df0 <- coronavirus[coronavirus$country == COUNTRY,]
head(df0)
tail(df0)
(pop <- df0$population[1])
df <- df0[c(1,6,7,13)]
str(df)
head(df)
### alternatively,
head(df0[c("date", "type", "cases", "population")])
###

## Set types
df_confirmed <- df[df$type == "confirmed",]
df_death <- df[df$type == "death",]
df_recovery <- df[df$data_type == "recovery",]
head(df_confirmed)
head(df_death)
head(df_recovery)

## Histogram
plot(df_confirmed$date, df_confirmed$cases, type = "h")
plot(df_death$date, df_death$cases, type = "h")
# plot(df_recovered$date, df_recovered$cases, type = "h") # no data for recovery

## Scatter plot and correlation
plot(df_confirmed$cases, df_death$cases, type = "p")
cor(df_confirmed$cases, df_death$cases)
```

```

## In addition set a period
start_date <- as.Date("2022-07-01")
end_date <- Sys.Date()
df_date <- df[df$date >=start_date & df$date <= end_date,]
##

## Set types
df_date_confirmed <- df_date[df_date$type == "confirmed",]
df_date_death <- df_date[df_date$type == "death",]
df_date_recovery <- df_date[df_date$data_type == "recovery",]
head(df_date_confirmed)
head(df_date_death)
head(df_date_recovery)

## Histogram
plot(df_date_confirmed$date, df_date_confirmed$cases, type = "h")
plot(df_date_death$date, df_date_death$cases, type = "h")
# plot(df_date_recovered$date, df_date_recovered$cases, type = "h") # no data for

plot(df_date_confirmed$cases, df_date_death$cases, type = "p")
cor(df_date_confirmed$cases, df_date_death$cases)

#### Extra
plot(df_confirmed$date, df_confirmed$cases, type = "h",
     main = paste("Comfirmed Cases in",COUNTRY),
     xlab = "Date", ylab = "Number of Cases")

```

16.5.3 練習

上の、coronavirus.R について

1. COUNTRY <- "Japan" の Japan を他の国に変えてみましょう。
2. start_date <- as.Date("2022-07-01") の日付を、他の日付に変えてみましょう。
3. df_confirmed\$cases と df_death\$cases についてどんなことがわかりますか。
4. 発見や、問いがあれば、書き出してみましょう。

16.5.4 Tips

キーボード・ショートカットと言われる、さまざまな機能があります。

- 上のメニュー・バー：Help > Keyboard Short Cut Help 確認してみてください。

- 右下の窓枠: Files タブから、ファイルの確認ができます。

16.6 パッケージ - Packages

R packages are extensions to the R statistical programming language containing code, data, and documentation in a standardised collection format that can be installed by users of R using Tool > Install Packages in the top menu bar of R Studio. https://en.wikipedia.org/wiki/R_package

R パッケージは、R の拡張機能で、コード、データ、ドキュメントを標準化されたコレクション形式で含んでおり、標準的なものは、R Studio の Top Bar の Tool > Install Packages からインストールできます。

16.6.1 パッケージのインストール

いずれ使いますので、まずは、三つのパッケージをインストールしてみましょう。

- tidyverse
- rmarkdown
- tinytex

インストール方法はいくつかあります。

一つ目は、上のメニューバーの Tool から、Install Packages ... を選択します。そして、パッケージズにインストールしたい、パッケージ名を入力します。そのパッケージ名が下にも出れば、Install ボタンを押してください。入力した名前の下にパッケージ名が出ない場合は、スペルが間違っている可能性がありますから、確認して、入れ直してください。

Console に、`install.packages("tidyverse")` などと表示され、たくさんメッセージが出ます。終了すると、> のマークがでます。

二つ目は、`install.packages("tidyverse")` のような書式で書いて、Console に入れる方法です。

三つ目は、右下の窓枠の Packages のタブにある、Install というボタンを押す方法です。すると、一番目の方法に、戻り、パッケージ名を入力できるようになります。

この Packages タブにある、ものが、すでに、インストールされているパッケージです。そのなかで、base や、datasets などいくつかは、チェックがついていると思いますが、それらは、ロードされていて、いつでも、使える状態になっていることを意味しています。ロードは、たとえば、`library(tidyverse)` のようにしますが、それは、いずれもう一度説明します。

インストールは一回だけ。ときどき、Tools > Check for Package Update をつかって、Update しておくといいでしょう。

パッケージのインストールで問題が生じることがあります。特に、Windows の日本語システムの場合です。(4.3.2 R Studio のインストールの下に書いてある部分を参照してください。)

回避方法もいくつかあるようですが、混乱をさけるため、その場合は、Posit Cloud (旧: RStudio Cloud) を使うと良いでしょう。それを見越して、最初は、Posit Cloud ではじめることを、わたしはお薦めしています。自分のコンピュータで、R が RStudio 上で問題なく動いていても、Cloud 上にアカウントを持っていて、実行できることは有効ですし、全員が、同じ環境で作業できることもたいせつなことです。他にも、すぐ、Cheat Sheets (早見表) や、Posit Primers という練習問題 (Tutorial) を利用できたり、プロジェクトを共有したりなど、さまざまなメリットがあります。

16.6.2 備考

Package によっては、Source から Compile するかと聞いてくる場合があります。どちらでも、良いのですが、特に、問題が起こっていなければ、No でよいと思います。コンピュータにあった形でインストールすることが必要な場合は、Yes とします。

同じパッケージをもう一度、インストールしたり、または、関連するパッケージがあるような場合、R をリスタートするかと聞いてくる場合があります。特に問題が起こらなければ、No で構いません。ただ、エラーが起こって、それに関連して、特別なパッケージをインストールする必要がある場合がありますが、そのときは、Yes としてください。

16.7 練習問題 Posit Primers

Posit Primers <https://posit.cloud/learn/primers>

教科書 “R for Data Science” は、`tidyverse` パッケージを中心に、データサイエンスについて解説したのですが、Posit Primers は、演習問題をしながら、教科書の内容を理解できるように構成されています。

Posit Cloud からは、左のメニュー (隠れている場合は左上の 3 本線をクリックして表示させて) から選ぶことができます。そうでない場合は、直接、上のリンクから、利用してください。

16.7.1 最初の演習 The Basics – r4ds: Explore, I

- Visualization Basics
- Programming Basics

ぜひこれら二つの演習問題を、トライしてください。解説を読んでもただでは、データサイエンスは身につけません。

16.8 参考文献 References

一番目は、すでに紹介した、教科書です。二番目は、この文書を作成している、Bookdown というパッケージのサイトですが、そこに、たくさんの本が、無償で公開されています。素晴らしい本がたくさん含まれています。

- R For Data Science, by H. Wickham: <https://r4ds.had.co.nz>
 - Introduction: <https://r4ds.had.co.nz/explore-intro.html#explore-intro>
- Bookdown: <https://bookdown.org>, Archive

下の一番目は、R 入門を、2 時限の講義でしたときのものです。二番目と三番目は、講義で使ったものを、まとめたものです。教科書のように、できていませんが、参考になる部分もあるかと思しますので、紹介しておきます。

- Introduction to R
- Data Analysis for Researchers 2022
- Data Analysis for Researchers 2021

16.9 YouTube Video - getstarted

- ファイル: <https://ds-sl.github.io/intro2r/getstarted.html>

16.10 まとめと復習

ここでは、詳細までは理解していただかなくて結構ですが、基本事項をまとめておきます。

1. R Studio で R を使うときには、まず、Project を作成し、その中で行う。
2. コンソール (Console) に、入力して実行させる方法と、スクリプト (R Script) から実行する方法がある。もう一つ、RMarkdown の章で紹介した、RMarkdown の中で、実行する方法があります。
3. データを簡単にみる三つの命令、`head()`、`str()`、`summary()` を紹介しました。他にも、`View()` でデータを見たり、`Help` を使って、調べる方法を紹介しました。
4. パッケージによって、便利なさまざまな拡張機能を利用するために、そのインストール方法を紹介しました。
5. 対話型の練習問題集、Posit Primers の紹介をしました。

第 17 章

Tidyverse

17.1 はじめに

R のはじめかたについて少し説明しました。R を起動させると、最初に `base` などいくつかのパッケージが自動的に読み込まれますが、その基本の基本を紹介したということです。ここでは、第二部で学ぶことの、核となる `tidyverse` というパッケージ群についてその基本を学びます。

サイト：<https://www.tidyverse.org>

R は、さまざまな分野で、統計分析に利用されてきたこともあり、それぞれの分析でよく使われる関数 (functions, 小さなプログラム) を集めたパッケージが作られ、それぞれの分野で使われてきました。まさに、痒いところに手が届く、さまざまなパッケージが存在します。他方、さまざまな人たちが開発を続けてきたために、統一性がない、保守の継続性が十分ではない、異なるパッケージに同じ名前の関数があるなどの問題も生じてきたように思います。

さらに、それぞれの分野での、統計分析だけではなく、学際分野としてのデータサイエンスでの利用、AI への応用などにも使われるようになったこと、さらに、データサイエンスでは、特に可視化が重要で、その部分の基本が、幾何表現の文法 (Grammar of Graphics) を踏まえた `ggplot2` パッケージのによって、確立したことから、統一した思想のもとで、構築されたのが、`tidyverse` パッケージ群です。さまざまな改善によって、プログラミング言語としても、十分なレベルの言語となっていると思います。

一連の `tidyverse` パッケージ群の開発が、R 自体にも影響を与え、R の起動時に読み込まれるようになったものもあります。

パッケージ群と呼びましたが、たくさんのパッケージが全体として、`tidyverse` と呼ばれ、`tidyverse` をインストールすると、`tidyverse` パッケージ群のパッケージがすべてインストールされます。

それを、`library(tidyverse)` など、ロード (使えるようにするために) すると、`tidyverse` パッケージ群の主要なパッケージがみな、読み込まれます。ただ、こちらは、

「主要な」ものだけですので、tidyverse パッケージ群のパッケージでも、後ほど、個別に読み込む必要があるものもありますので、注意してください。

17.2 あやめ (iris) のデータ

利用するデータ、特に、その変数 (列) 名に日本語 (中国語・韓国語など) を使う場合には、`install.packages('showtext')` で、`showtext` パッケージをインストールして、下のように設定ます。標準的には、最初の行 `library(tidyverse)` だけで十分です。

```
library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats    1.0.0      v stringr    1.5.0
#> v ggplot2    3.4.3      v tibble     3.2.1
#> v lubridate  1.9.2      v tidyr      1.3.0
#> v purrr      1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()   masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all con
library(showtext)
#> Loading required package: sysfonts
#> Loading required package: showtextdb
showtext_auto()
```

いろいろなメッセージが表示されました。簡単に説明しておきます。

- `Attaching core tidyverse packages` : パッケージを読み込む、使えるようにするのが、`library()` という、命令ですが、ここでは、たくさんのパッケージが読み込まれたことが書かれています。Tidyverse パッケージ群の中の核となるパッケージです (core tidyverse packages)。実は、これ以外にも、Tidyverse パッケージ群は、パッケージを含んでいますが、それは、別に、読み込まないといけません。その都度説明します。
- `Conflicts` : 実は、違うパッケージの中に、同じ名前の関数 (命令) が含まれている場合があります。その場合は、あとから読み込んだパッケージの命令が優先されます。いま、tidyverse を読み込んだので、その前に、読み込まれていたパッケージのいくつかの、関数が使えなくなりましたというメッセージです。
- 次に、`showtext` を読み込んでいます。そこでは、`sysfonts` と `showtextdb` も必要なので、読み込んだと書かれています。

以下では、R の起動時に読み込まれる、`datasets` パッケージの中の、`iris` データを使います。以前には、同じパッケージに含まれる、`cars` を使いました。まず、このデータ

に、`ds_iris` という名前をつけて使うことにします。これは、アサインと言い、`<-` を使います。半角で入力します。

備考：

1. Help の Keyboard Shortcuts Help をみると、さまざまなキーボードショートカットが書かれています。`<-` は、Windows では、`Alt + -`、Mac では、`Option + -` で、入力することもできます。`Alt + -` などは、`Alt` キーを押しながら、`-` キーを押すという意味です。
2. `df_iris <- iris` でも問題ないと思います。下のようにするのは、すでに、`iris` という変数を使っているかもしれないので、`datasets` パッケージの `iris` を使うという意味です。

```
df_iris <- datasets::iris
class(df_iris)
#> [1] "data.frame"
```

あやめのデータ `iris` は `data.frame` というクラスであることがわかります。`tidyverse` には、`tibble` というデータのクラス (`data.frame` のサブクラス) もあります。

```
tbl_iris <- as_tibble(datasets::iris)
class(tbl_iris)
#> [1] "tbl_df"      "tbl"        "data.frame"
```

`df_iris` と `tbl_iris` で出力が変わる場合もありますが、今のところは、あまり気にせず、`df_iris` を使っていきたいと思います。

17.2.1 データを見てみよう

17.2.1.1 復習

すでに、いくつかの関数を学んでいますから、それを使ってみましょう。`head`, `str`, `summary` でした。

```
head(df_iris)
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1         5.1         3.5         1.4         0.2  setosa
#> 2         4.9         3.0         1.4         0.2  setosa
#> 3         4.7         3.2         1.3         0.2  setosa
#> 4         4.6         3.1         1.5         0.2  setosa
#> 5         5.0         3.6         1.4         0.2  setosa
#> 6         5.4         3.9         1.7         0.4  setosa
```

■17.2.1.1.1 head : データの頭の部分 データの最初の 6 行 (head) が表示されました。

```
str(df_iris)
#> 'data.frame': 150 obs. of 5 variables:
#> $ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
#> $ Sepal.Width : num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
#> $ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
#> $ Petal.Width : num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
#> $ Species : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1
```

■17.2.1.1.2 str : データの構造 (structure) 全体の構造 (structure) の概略が表示されました。5つの変数、Sepal.Length (萼(がく)長), Sepal.Width (萼幅), Petal.Length (花弁長), Petal.Width (花弁幅), Species (種別) について、150 のデータ (observations) が含まれており、Species は、三つの Factor になっているということがわかります。Factor については、後ほど学びますが、三つに分類されているという意味で、ここでは、あやめの種類、setosa, versicolor, virginica となっています。三つ目は見えていないかもしれませんが。

次の、summary() を使うと、Species の三種類がなにかわかりませんが、unique() という関数を使って、みてみましょう。df_iris の Species に対応する列は、df_iris\$Species で取り出せますので、次のようにします。Level と書いてあるのは、Factor という形式になっているので、その順序が書かれています。この場合は、アルファベットの順序になっています。

```
unique(df_iris$Species)
#> [1] setosa versicolor virginica
#> Levels: setosa versicolor virginica
```

```
summary(df_iris)
#> Sepal.Length Sepal.Width Petal.Length
#> Min. :4.300 Min. :2.000 Min. :1.000
#> 1st Qu.:5.100 1st Qu.:2.800 1st Qu.:1.600
#> Median :5.800 Median :3.000 Median :4.350
#> Mean :5.843 Mean :3.057 Mean :3.758
#> 3rd Qu.:6.400 3rd Qu.:3.300 3rd Qu.:5.100
#> Max. :7.900 Max. :4.400 Max. :6.900
#> Petal.Width Species
#> Min. :0.100 setosa :50
```



```
#> 1st Qu.:0.300 versicolor:50
#> Median :1.300 virginica :50
#> Mean   :1.199
#> 3rd Qu.:1.800
#> Max.   :2.500
```

■17.2.1.1.3 summary: **データの概要** それぞれの最小値 (Min. Minimum)、第一四分位 (1st Qu. First Quartile 小さい方から、4分の1を切り捨てたときの最小の値)、中央値 (Median)、平均 (Mean)、第三四分位 (3rd Qu. Third Quartile、大きい方から、4分の1を切り捨てたときの最大の値) 最大値 (Max. Maximum) が、書かれており、各種それぞれ 50 個のデータからなっていることがわかります。

base R では、さらに、`plot`、`View`、`help` についても学びました。

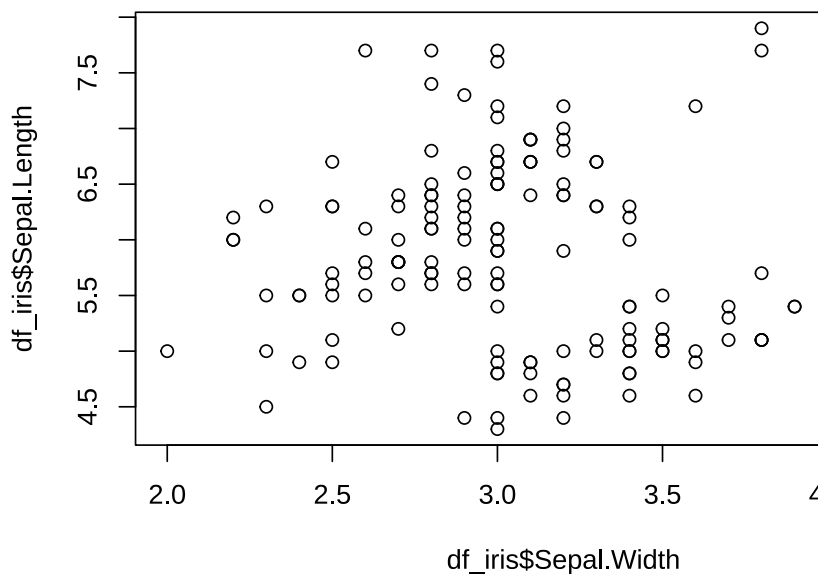
これは、あやめ (iris) の、長さや幅のデータでしたが、単位は何でしょうか。それも、`help` で調べることができます。Help の検索ボックスに、`iris` と入れてみてください。この場合は、Description (概要) に次のように書かれています。

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

この有名な (フィッシャーまたはアンダーソンの) あやめのデータは、3種のあやめからそれぞれ 50 個の花について、萼片の長さや幅、花弁の長さや幅の変数をそれぞれセンチメートル単位で測定したものです。種は、*Iris setosa*, *versicolor*, *virginica* です。

各、変数の説明が書かれている場合もありますので、Help は、かならず確認するようにしてください。

```
plot(df_iris$Sepal.Width, df_iris$Sepal.Length)
```



■17.2.1.1.4 plot: 散布図

何を x 軸、何を y 軸の値とするかを指定します。それは、`df_iris$Sepal.Width` で、`df_iris` の、`Sepal.Width` の列、`df_iris$Sepal.Length` で、`df_iris` の `Sepal.Length` の列の値を指定しています。

```
View(df_iris)
```

■17.2.1.1.5 View: データテーブルの表示 `View(df_iris)` を、Console に入れると、データテーブルが開きます。順番の並び替えもできるようになっています。

R Studio では、右上の窓に、Environment タブがありますが、そこに、`df_iris` があると思いますから、それを、クリックすると、同じ、データテーブルが表示されます。

```
help(iris)
```

■17.2.1.1.6 help: ヘルプ `help(iris)` を、Console に入れるか、または、右下の窓枠の Help に、`iris` と入れると、説明などが出ます。

ここまでは、`tidyverse` を使わずにもできることですが、これからは、`tidyverse` の関数を紹介していきます。

17.2.2 dplyr 変形

ここでは、`head` と、`str` に対応する、二種類の関数を紹介するにとどめますが、次の、章の中心的トピックです。

17.2.2.1 slice: 行を切り取る

`head` を一般化したものです。`slice`, `slice_head`, `slice_tail`, `slice_max`, `slice_min`, `slice_sample` とあります。

```
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width
#> 1           5.1           3.5           1.4           0.2
#> 2           4.9           3.0           1.4           0.2
#> 3           4.7           3.2           1.3           0.2
#> 4           4.6           3.1           1.5           0.2
#> 5           5.0           3.6           1.4           0.2
#> 6           5.4           3.9           1.7           0.4
#> 7           4.6           3.4           1.4           0.3
#> 8           5.0           3.4           1.5           0.2
#> 9           4.4           2.9           1.4           0.2
#> 10          4.9           3.1           1.5           0.1
#>   Species
#> 1   setosa
#> 2   setosa
#> 3   setosa
#> 4   setosa
#> 5   setosa
#> 6   setosa
#> 7   setosa
#> 8   setosa
#> 9   setosa
#> 10  setosa
```

`|>` は、パイプ (pipe) コマンドと言われるもので、`tidyverse` では、`%>%` もほぼ同じ機能ですが、R 4.0 以降には、含まれていますので、`tidyverse` なしでも使えますから、こちらを使うようにします。最初の 10 行を切り出すという意味で、パイプを使わないときは、一つ前のものが、最初の変数 (argument) となりますから、下のものでも同じです。

```
slice(df_iris, 1:10)
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width
#> 1           5.1           3.5           1.4           0.2
#> 2           4.9           3.0           1.4           0.2
#> 3           4.7           3.2           1.3           0.2
#> 4           4.6           3.1           1.5           0.2
#> 5           5.0           3.6           1.4           0.2
#> 6           5.4           3.9           1.7           0.4
#> 7           4.6           3.4           1.4           0.3
#> 8           5.0           3.4           1.5           0.2
```

```
#> 9      4.4      2.9      1.4      0.2
#> 10     4.9      3.1      1.5      0.1
#> Species
#> 1 setosa
#> 2 setosa
#> 3 setosa
#> 4 setosa
#> 5 setosa
#> 6 setosa
#> 7 setosa
#> 8 setosa
#> 9 setosa
#> 10 setosa
```

```
1:10
```

```
#> [1] 1 2 3 4 5 6 7 8 9 10
```

1 から 10 のベクトルです。ということは、1:10 の部分をいろいろと変えれば、さまざまな部分を取り出すことができます。詳細は、Help 検索窓で、`slice` と調べてください。下も同じ結果を出力します。

```
df_iris |> slice_head(n=10)
```

```
#> Sepal.Length Sepal.Width Petal.Length Petal.Width
#> 1      5.1      3.5      1.4      0.2
#> 2      4.9      3.0      1.4      0.2
#> 3      4.7      3.2      1.3      0.2
#> 4      4.6      3.1      1.5      0.2
#> 5      5.0      3.6      1.4      0.2
#> 6      5.4      3.9      1.7      0.4
#> 7      4.6      3.4      1.4      0.3
#> 8      5.0      3.4      1.5      0.2
#> 9      4.4      2.9      1.4      0.2
#> 10     4.9      3.1      1.5      0.1
#> Species
#> 1 setosa
#> 2 setosa
#> 3 setosa
#> 4 setosa
#> 5 setosa
#> 6 setosa
#> 7 setosa
#> 8 setosa
#> 9 setosa
```

```
#> 10 setosa

df_iris |> slice_max(order_by = Sepal.Length, n=7)
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width
#> 1           7.9         3.8         6.4         2.0
#> 2           7.7         3.8         6.7         2.2
#> 3           7.7         2.6         6.9         2.3
#> 4           7.7         2.8         6.7         2.0
#> 5           7.7         3.0         6.1         2.3
#> 6           7.6         3.0         6.6         2.1
#> 7           7.4         2.8         6.1         1.9
#>   Species
#> 1 virginica
#> 2 virginica
#> 3 virginica
#> 4 virginica
#> 5 virginica
#> 6 virginica
#> 7 virginica
```

Sepal.Length の値を大きい方から順に並べて、最初の7つを選択するというものです。

17.2.2.2 glimpse: データの構造

str の改良版です。他の、tidyverse の関数と一緒に使うこともできます。

```
glimpse(df_iris)
#> Rows: 150
#> Columns: 5
#> $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.~
#> $ Sepal.Width <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.~
#> $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.~
#> $ Petal.Width <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.~
#> $ Species <fct> setosa, setosa, setosa, setosa, setos~
```

```
df_iris |> glimpse()
#> Rows: 150
#> Columns: 5
#> $ Sepal.Length <dbl> 5.1, 4.9, 4.7, 4.6, 5.0, 5.4, 4.6, 5.~
#> $ Sepal.Width <dbl> 3.5, 3.0, 3.2, 3.1, 3.6, 3.9, 3.4, 3.~
#> $ Petal.Length <dbl> 1.4, 1.4, 1.3, 1.5, 1.4, 1.7, 1.4, 1.~
#> $ Petal.Width <dbl> 0.2, 0.2, 0.2, 0.2, 0.2, 0.4, 0.3, 0.~
#> $ Species <fct> setosa, setosa, setosa, setosa, setos~
```

`str(df_iris)` では `num` (数値データ) と表示されましたが、ここでは、もう少し詳 R の 6 個のデータタイプ `Double` (連続データ), `Integer` (整数値データ), `Character` (文字データ), `Logical` (論理値データ), `Raw` (素データ), `Complex` (複素数データ) が表示されます。最初の四種類が主要なものと考えてください。

データの列を指定するときは、データ名のあとに、ドルマークをつけ、列名を加えます。`df_iris$Sepal.Width` などです。これは、一列目ですから、`typeof(df_iris[[1]])` とすることもできます。2 行目の 1 列目というときは、次のようにします。

`df_iris[2,1] = 4.9` is the fourth entry of `Sepal.Width`.

```
typeof(df_iris$Sepal.Width)
```

```
#> [1] "double"
```

```
typeof(df_iris$Species)
```

```
#> [1] "integer"
```

```
class(df_iris$Species)
```

```
#> [1] "factor"
```

ファクター `factors = fct` については the R Document または、Factor in R: Categorical Variable & Continuous Variables をみてください。必要になったときに、説明します。

```
typeof(df_iris$Sepal.Length)
```

```
#> [1] "double"
```

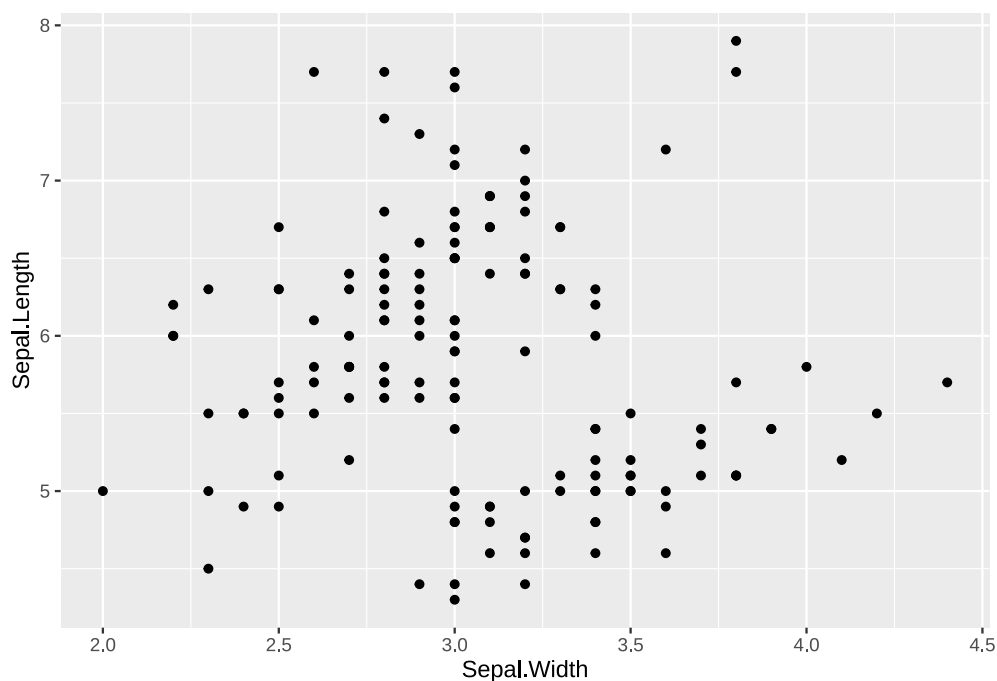
```
class(df_iris$Sepal.Length)
```

```
#> [1] "numeric"
```

17.2.2.3 ggplot2: グラフの描画

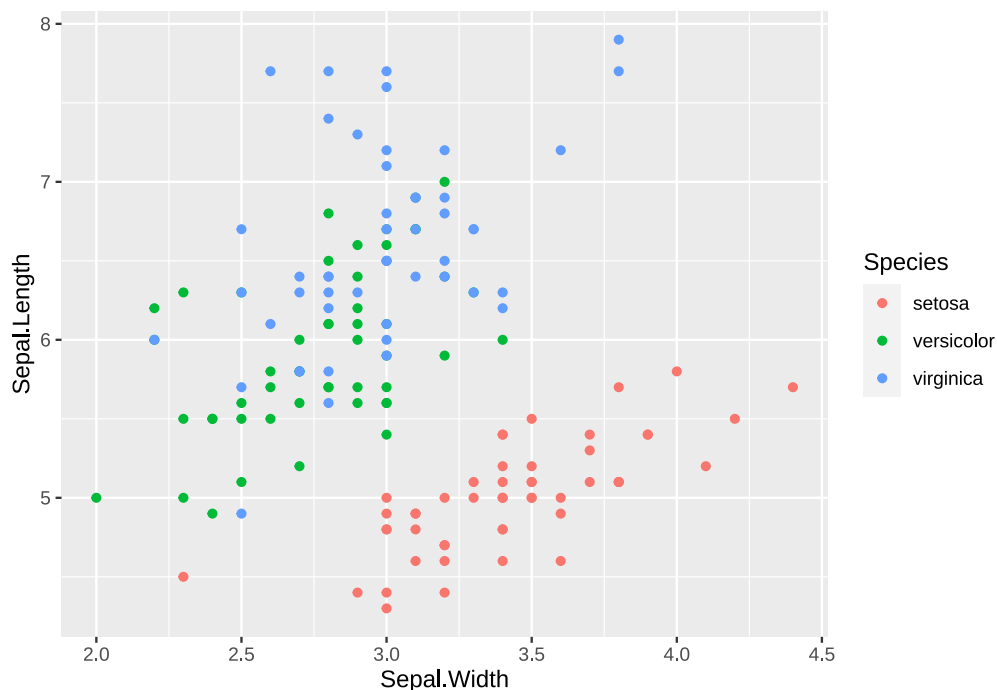
`plot` の拡張ですが、`tidyverse` パッケージ群の核をなすものでもあります。詳細は、視覚化で扱います。

```
df_iris |> ggplot(aes(Sepal.Width, Sepal.Length)) + geom_point()
```



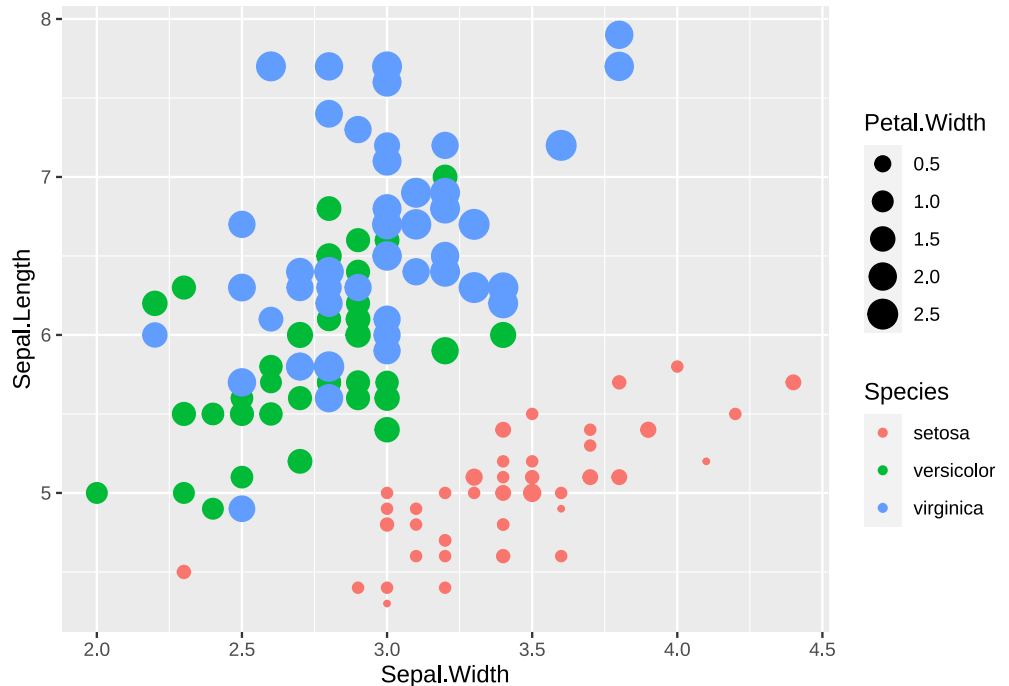
さまざまな描画が可能です。一番、一般的な、散布図、`plot` に対応するものを書きました。`ggplot` の中の、`aes` (aesthetic) の部分に、x 軸、y 軸に対応する変数 (列名) を書きます。種類 (Species) ごとに色を変える場合には、`color = Species` とします。

```
df_iris |> ggplot(aes(Sepal.Width, Sepal.Length, color = Species)) +  
  geom_point()
```



さらに、点の大きさを、`Petal.Width` によって変える場合には次のように、`size = Petal.Width` を加えます。

```
df_iris |>
  ggplot(aes(Sepal.Width, Sepal.Length, color = Species,
             size = Petal.Width)) +
  geom_point()
```



少しずつ学んでいきましょう。

17.3 WDI のデータ

以下の説明では、世界開発指標 (World Development Indicator) の実際のデータも使って説明していきます。例として使うデータを取得して、上で学んだことの復習もかねて、簡単にみておきたいと思います。

すでに、tidyverse は読み込んでありますから、その場合は、WDI パッケージを読み込むだけで十分です。

```
library(tidyverse)
library(WDI)
```

WDI の使い方は、世界銀行の部分で紹介しますが、はじめてのデータサイエンスの例でも紹介したように、データコードを利用して、データを読み込みます。ここでは、出生時の平均寿命と、一人当たりの GDP と、総人口のデータを使います。

- SP.DYN.LE00.IN: Life expectancy at birth, total (years) 出生時の平均寿命
- SP.POP.TOTL: Population, total 総人口
- NY.GDP.PCAP.KD: GDP per capita (constant 2015 US\$) 一人当たりの GDP

次のコードで読み込みます。

```
df_wdi <- WDI(
  country = "all",
  indicator = c(lifeExp = "SP.DYN.LE00.IN", pop = "SP.POP.TOTL", gdpPercap = "NY.GDP.PCAP.KD")
)
```

```
#> Rows: 16758 Columns: 7
#> -- Column specification -----
#> Delimiter: ","
#> chr (3): country, iso2c, iso3c
#> dbl (4): year, lifeExp, pop, gdpPercap
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

注：WDI のサイトは、頻繁に、保守をしているため、時々、データをダウンロードできないことがあります。そのときは、すでに、ダウンロードしてものがわたしの GitHub サイトにありますから、そこから次のコードで読み込んでください。

最初の 10 行をみてみましょう。

```
df_wdi %>% slice(1:10)
#> # A tibble: 10 x 7
#>   country      iso2c iso3c  year lifeExp      pop gdpPercap
#>   <chr>        <chr> <chr> <dbl>  <dbl>    <dbl>    <dbl>
#> 1 Afghanistan AF     AFG   1960   32.5  8622466     NA
#> 2 Afghanistan AF     AFG   1961   33.1  8790140     NA
#> 3 Afghanistan AF     AFG   1962   33.5  8969047     NA
#> 4 Afghanistan AF     AFG   1963   34.0  9157465     NA
#> 5 Afghanistan AF     AFG   1964   34.5  9355514     NA
#> 6 Afghanistan AF     AFG   1965   35.0  9565147     NA
#> 7 Afghanistan AF     AFG   1966   35.5  9783147     NA
#> 8 Afghanistan AF     AFG   1967   35.9 10010030     NA
#> 9 Afghanistan AF     AFG   1968   36.4 10247780     NA
#> 10 Afghanistan AF     AFG   1969   36.9 10494489     NA
```

```
df_wdi_extra <- WDI(
  country = "all",
  indicator = c(lifeExp = "SP.DYN.LE00.IN", pop = "SP.POP.TOTL", gdpPercap = "NY.GDP.PCAP.KD"),
  extra = TRUE
)
```

すこし、追加情報を付加したのも取得しておきます。

```
#> Rows: 16758 Columns: 15
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (6): year, lifeExp, pop, gdpPercap, longitude, lati...
#> lgl (1): status
#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa
```

```
df_wdi_extra
#> # A tibble: 16,758 x 15
#>   country      iso2c iso3c  year status lastupdated lifeExp
#>   <chr>         <chr> <chr> <dbl> <lgl> <date>      <dbl>
#> 1 Afghanistan AF    AFG   2014 NA    2023-09-19  62.5
#> 2 Afghanistan AF    AFG   2012 NA    2023-09-19  61.9
#> 3 Afghanistan AF    AFG   2009 NA    2023-09-19  60.4
#> 4 Afghanistan AF    AFG   2013 NA    2023-09-19  62.4
#> 5 Afghanistan AF    AFG   1971 NA    2023-09-19  37.9
#> 6 Afghanistan AF    AFG   2015 NA    2023-09-19  62.7
#> 7 Afghanistan AF    AFG   1969 NA    2023-09-19  36.9
#> 8 Afghanistan AF    AFG   2010 NA    2023-09-19  60.9
#> 9 Afghanistan AF    AFG   2011 NA    2023-09-19  61.4
#> 10 Afghanistan AF    AFG   2008 NA    2023-09-19  59.9
#> # i 16,748 more rows
#> # i 8 more variables: pop <dbl>, gdpPercap <dbl>,
#> #   region <chr>, capital <chr>, longitude <dbl>,
#> #   latitude <dbl>, income <chr>, lending <chr>
```

違いはわかりましたか。同じような変数についてのデータですが、WDI からダウンロードした実際のデータの場合には、練習用のデータとは、違った困難がいくつもあります。それを、少しずつ見ていきながら、現実世界のデータを扱えるようにしていきましょう。

`glimpse` を使ってみるとどのような違いがわかりますか。

```
df_wdi |> glimpse()
#> Rows: 16,758
#> Columns: 7
#> $ country <chr> "Afghanistan", "Afghanistan", "Afghanist~
#> $ iso2c <chr> "AF", "AF", "AF", "AF", "AF", "AF", "AF"~
#> $ iso3c <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AFG"~
#> $ year <dbl> 1960, 1961, 1962, 1963, 1964, 1965, 1966~
#> $ lifeExp <dbl> 32.535, 33.068, 33.547, 34.016, 34.494, ~
```

```
#> $ pop      <dbl> 8622466, 8790140, 8969047, 9157465, 9355~
#> $ gdpPercap <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
```

```
df_wdi_extra |> glimpse()
#> Rows: 16,758
#> Columns: 15
#> $ country    <chr> "Afghanistan", "Afghanistan", "Afghani~
#> $ iso2c      <chr> "AF", "AF", "AF", "AF", "AF", "AF", "A~
#> $ iso3c      <chr> "AFG", "AFG", "AFG", "AFG", "AFG", "AF~
#> $ year       <dbl> 2014, 2012, 2009, 2013, 1971, 2015, 19~
#> $ status     <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA~
#> $ lastupdated <date> 2023-09-19, 2023-09-19, 2023-09-19, 2~
#> $ lifeExp    <dbl> 62.545, 61.923, 60.364, 62.417, 37.923~
#> $ pop        <dbl> 32716210, 30466479, 27385307, 31541209~
#> $ gdpPercap  <dbl> 602.5166, 596.4424, 512.4090, 608.3863~
#> $ region     <chr> "South Asia", "South Asia", "South Asi~
#> $ capital    <chr> "Kabul", "Kabul", "Kabul", "Kabul", "K~
#> $ longitude  <dbl> 69.1761, 69.1761, 69.1761, 69.1761, 69~
#> $ latitude   <dbl> 34.5228, 34.5228, 34.5228, 34.5228, 34~
#> $ income     <chr> "Low income", "Low income", "Low incom~
#> $ lending    <chr> "IDA", "IDA", "IDA", "IDA", "IDA", "ID~
```

列の数が違いますね。どちらも行は、16,758 行です。こんなに多いものは、表で見ている、よくわかりません。これらを扱う方法を少しずつ学んでいきます。

RStudio の右上の窓の Environment から、df_wdi や、df_wdi_extra を選択して、データを表示してみてください。それぞれの列 (変数) の右の二つの三角を押すと、昇順 (小さい順)、降順 (大きい順) に並び替えることもできますし、Filter を使うと、検索も可能です。たとえば、Japan や、JP や、JPN などと検索窓に入れて、日本のデータを表示させてみてください。h 表示させてから、Year 列の右の二つの三角を押して、順序をかえてみてください。それぞれの変数何年からのデータがありますか。

• 引数 (argument) について

関数の中に指定するものを引数と言います。たとえば、head(df_iris) の df_iris、

WDI() の () の中の、country や、indicator、ggplot() の () の中の、aes や、color や size などです。Help を一つ一つ見てみるのが良いのですが、引数に名前がついている場合と、ついていない場合があります。三通りがあります。

1. 引数の要・不要
2. 引数の名前がある場合の名前の省略

3. 引数の順序

すべて Help を見ないとわかりませんが、原則は、以下のとおりです。

1. 引数が必要であっても、その初期値 (default value) が決まっています、変更の必要がなければ、引数を指定する必要なし。
2. 引数の名前がある場合であっても、引数の順序どおりであれば、名前は省略可能。
3. 引数には順序があるが、名前のついている引数は、名前をつければ、順序を変更しても構わない。

例で見ていきましょう。

- `head(df_iris)` は、Help には、`head(x, ...)` と書いてあり、`x` は `object` と書いてありますから、この場合は、`df_iris` が `x` に対応します。... は、任意で、詳細は Help の下の方に書いてあります。さらに見ると、`fable` (表 S3: flat contingency table) の場合には、`head(x, n = 6L, ...)` とあります。最初の 6 行を取り出すという意味です。`n=6L` (6L は整数値で 6 の意味) は、6L が初期値ですから、書く必要はありませんが、もし、6 を他の数字、たとえば 10 に変えれば、10 行ということになります。L の部分は、なくても自然数 (1,2,3, ...) なら普通に理解してくれます。`head(df_iris, n=-145)` とすると何が表示されるでしょうか。
- `WDI(country = "all", indicator = c(lifeExp = "SP.DYN.LE00.IN", pop = "SP.POP.TOTL", gdpPercap = "NY.GDP.PCAP.KD"))` はどうでしょうか。Help を見てみると、次のようにあります。

```
WDI(
  country = "all",
  indicator = "NY.GDP.PCAP.KD",
  start = 1960,
  end = NULL,
  extra = FALSE,
  cache = NULL,
  latest = NULL,
  language = "en"
)
```

* 最初の引数は、`country` で初期値は “all”、`indicator` は、初期値は “NY.GDP.PCAP.KD” ですが、それを、`c()` を使ったベクトル表示で与えています。

* `country = "all"` は初期値と同じですから、省略可能です。二番目の引数は、`indicator` ですから、`country = "all"` を書いてあれば、“`indicator =`” の部分は省略可能です。

* `extra = FALSE` とありますから、この場合は、何も指定しませんでした
が、`df_wdi_extra` の場合には、`extra = TRUE` としました。

– `df_iris |> ggplot(aes(Sepal.Width, Sepal.Length, color = Species, size =
Petal.Width))` はどうでしょうか。Help を見ると下のようになっていますが、
Details の部分も書いておきます。これは、他のパターンもあります。

```
ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())  
ggplot(data = df, mapping = aes(x, y, other aesthetics))
```

* `data` の部分は、pipe (`|>`) で、最初に引数に `data = df_iris` とある
のと同じです。

* 二番目の引数が、`aes` の部分ですから、`mapping =` は省略しても問題あり
ません。

* `aes()` の中身は、最初が、`x`、次が `y` となっていますから、変数の順番があ
っていれば、`x = Sepal.Width`, `y = Sepal.Length` と書いても、書かなく
ても同じです。

* `color = Species`, `size = Petal.Width` の部分は、引数名が書かれているの
で、順番は、関係ありません。

慣れが必要ですから、原則を確認して、あとは、出てきたときに、一つ一つ確認し
てください。

17.4 まとめ

あやめ (`iris`) のデータを、`df_iris` として、読み込み、`WDI` パッケージを用いて、`WDI`
のデータを `df_wdi` および `df_wdi_extra` として読み込み、それをみてみました。

また、あやめのデータについては、散布図も描いて、それからわかることも少し考えてみ
ました。

`WDI` データはかなりの大きさですから、表を眺めていただけでは、わかることは限られ
ていますが、それでも、まずは、表がどのようなものかを確認することは、大切です。

17.4.1 関数および命令

この章で新たに使った関数をリストしておきます。

17.4.1.1 base R

- `unique()`

17.4.1.2 tidyverse

- `slice()`, `slice_head()`, `slice_max()`
- `glimpse()`
- `ggplot()`, `geom_point()`

17.4.1.3 WDI

- WDI

第 18 章

変形 (Transform)

ここでは、`tidyverse` パッケージを構成する `dplyr` パッケージを用いて、データを変形することを学びます。変形とは、例えば、データのなかのある条件を満たす行や列を抽出（取り出）したり、特定の列を修正したり、データの中のいくつかの列に含まれる情報をもとに計算して、新たな列を作ったり、グループに分けたり、順番を入れ替えたりといったことです。データを集約し（まとめ）たり、グラフを作成するときにも、必要不可欠な作業です。

`dplyr` パッケージを用いて、二つのデータを結合することなども可能ですが、それは、あとから扱います。これも、`tidyverse` パッケージを構成する `tidyr` パッケージによる変形も、後ほど扱います。

`dplyr` パッケージは、`tidyverse` パッケージをインストールしたり、使えるように `library(tidyverse)` で読み込んだりするときに、一緒に読み込まれますから、あらためて、`dplyr` を指定する必要はありません。

```
library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats   1.0.0      v stringr   1.5.0
#> v ggplot2   3.4.3      v tibble    3.2.1
#> v lubridate 1.9.2      v tidyr     1.3.0
#> v purrr     1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()   masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

18.1 dplyr 概要

dplyr はデータ操作の文法のようなもので、最も一般的なデータ操作に役立つ一貫したいくつかの「動詞」の役割を果たすものを提供しています。

- `select()` 変数とその名前によって選択 - 列の選択に対応します。
- `filter()` ケースをその値によって選択 - 行の選択に対応します。
- `mutate()` 新しい変数を既存の値を使って定義します - 新しい列を作成することに対応しています。
- `summarise()`^{*1} たくさんの値を一つの値に集約します - 代表値 (平均、メディアンなど) を求めることに対応します。
- `arrange()` 行の順序を変更します。
- `group_by()` グループを指定した表に変換します。

さらに詳しく知りたい場合は Console (コンソール) に `vignette("dplyr")` と入れるか、こちらを参照してください。上では、一つの表について述べていますが、二つの表の扱いは、Console (コンソール) に `vignette("two-table")` と入れるか、こちらを参照してください。二つの表の扱いについては、後から説明します。

dplyr を初めて使われる場合には、まず R for data science (2e) Transform を学ばれることをお勧めします。以下も、このサイトに沿った説明をします。

b

18.1.1 select: 名前とタイプによって、列 (変数) を選択

補助関数	条件	例
-	列の排除	<code>select(iris, -Species)</code>
:	列の範囲	<code>select(iris, 1:4)</code>
<code>contains()</code>	指定文字列を含む	<code>select(iris, contains("Width"))</code>
<code>ends_with()</code>	指定文字列で終わる	<code>select(iris, ends_with("th"))</code>
<code>matches()</code>	正規表現に適合	<code>select(iris, matches("S"))</code>
<code>num_range()</code>	末尾の指定数値範囲	Not applicable with iris
<code>one_of()</code>	指定した名前に含まれる	<code>select(iris, one_of(c("Sepal.Width", "Petal.Width")))</code>
<code>starts_with()</code>	指定文字列で始まる	<code>select(iris, starts_with("Petal"))</code>

「正規表現」という言葉が登場します。検索などでは基本的なものですので、ネット上で調べてください。適切に動作するかを、チェックする、正規表現チェッカーもあります。参

^{*1} `summarize()` とつづりをアメリカ式にしても問題ありません。他にも、さまざまな用語は、アメリカ式の綴りでも、イギリス式でも、問題ないようになっています。

照：Wikipedia

18.1.2 filter: 列の値の条件に適合した行の選択

論理作用素	条件	例
>	y より大きい x	x > y
>=	y 以上の x	x >= y
<	y より小さい x	x < y
<=	y 以下の x	x <= y
==	y と等しい x	x == y
!=	y と等しくない x	x != y
is.na()	値が NA である x	is.na(x)
!is.na()	値が NA でない x	!is.na(x)

NA は、not available 値が存在しないという意味です。欠損値のことで、df_wdi などには、一番上に、NA が出てきます。欠損値の扱いも、データサイエンスにおいてはとても大切です。

18.1.3 arrange

arrange() では、選択した列の値によって、行を並び替えます。

注意点すべきは、他の、dplyr の動詞とは異なり、基本的に、グループ化は、無視し、その表全体に適用します。グループ内で並び替えをしたい場合には、グループ化した変数を指定するか、.by_group = TRUE とします。

18.1.4 mutate

新しい列を作成または、既存の列を修正、削除します。

以下は便利な補助関数の例です。Help を参照してください。利用するときに、少しずつ説明していきます。ある程度慣れてきたら、どのようなことができるかを把握しておくことは、助けになります。

- +, -, log(), など：通常の数学記号を表します。
- lead(), lag() : lead は次の値、lag は一つ前の値です。
- dense_rank(), min_rank(), percent_rank(), row_number(), cume_dist(), ntile(): rank は、階級、順序を決めます。base::rank もあります。基本的に、同じ値があったときに、どのように順序を決めるかがそれぞれによって異なります。
- cumsum(), cummean(), cummin(), cummax(), cumany(), cumall(): cum は、

cumulative (累計) ですから、`cumsum` は累計、`cummean()` は、その値までの平均 (小計)、`cummin()` は、その値までの最小、`cummax()` その値までの最大、`cumany()`、`cumall()` は、Help を参照してください。

- `na_if()`、`coalesce()`: `na_if(x,y)` は、`x` の中で、`y` と等しいものは、NA に置き換え、`coalesce()` は、最初の欠損していない値を返します。

18.1.5 group_by

指定した列の値によって表全体をグループ化した表を作成します。表自体が変形されるわけではありませんから、注意してください。次の、`summarize` と合わせて利用すると便利です。

18.1.6 summarise or summarize

値の集約 (`summarize`) または、集計に利用します。グループ化された表の場合には、そのグループごとに、値を集約します。

18.1.6.1 集約のための関数

`summarize` には、`sum()`、`max()`、や `mean()` が使われますが、ベクトルに対して定義され、一つの値だけを出力する関数であれば、なんでも使うことができます。以下は、その例です。それぞれの関数については、Help で調べてください。

1. 特定の値 - `mean(x)`、`median(x)`、`quantile(x, 0.25)`、`min(x)`、`max(x)`
2. 分布の値 - `sd(x)`、`var(x)`、`IQR(x)`、`mad(x)`
3. 値の位置 - `first(x)`、`nth(x, 2)`、`last(x)`
4. 個数 - `n_distinct(x)`、`n()` (引数なし: 表またはグループのサイズ)
5. 論理値の数または割合 - `sum(!is.na(x))`、`mean(y == 0)`

条件文で値を指定することも可能です。

- `if_else()`、`recode()`、`case_when()`

注: `quantile` (分位数) と `quartile` (四分位数): 分位数はデータセットを任意の個数の等しい部分に分割するための値を示し、四分位数はデータセットを 4 つの等しい部分に分割するための特定の値。

18.1.7 パイプ (Pipe) %>% |>

`pipe` in R は、すでに説明してましたが、ここにまとめておきます。

%>% は、tidyverse パッケージで、関数のチェーン化を行うために使用されるパイプ演算子ですが、R 4.1 以降は、|> が、R に組み込まれた、ネイティブなパイプライン演算子になっています。tidyverse を使っているときは、どちらを使うことも可能ですが、|> を使うことをお勧めします。R の version を確認するには、コンソール (Console) で、R.Version() または、R.version\$version.string とします。

18.1.7.1 備考

1. パイプを使うコードで、複数行にまたがるときは、|> の後で、改行してください。次の行に続くことがわかります。
2. 実際には、tidyverse の %>% と、R に組み込まれた |> とは多少異なるようです。こちらの記事をご覧ください。本書では、|> を使いますが、注意が必要な場合は、コメントします。

18.2 例から学ぶ dplyr, I

18.2.1 Data iris

R 起動時に読み込まれる、datasets の中の、iris (あやめのデータ) を使い、いくつかの例を示します。iris は、何もしなくてもそのまま使えますが、もし、下の結果と違う場合には、すでに、iris を使い、変形などを行っている可能性がありますから、そのときは、下のコードを実行してください。

```
df_iris <- datasets::iris
```

確認します。

```
head(df_iris)
```

```
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1         5.1         3.5         1.4         0.2  setosa
#> 2         4.9         3.0         1.4         0.2  setosa
#> 3         4.7         3.2         1.3         0.2  setosa
#> 4         4.6         3.1         1.5         0.2  setosa
#> 5         5.0         3.6         1.4         0.2  setosa
#> 6         5.4         3.9         1.7         0.4  setosa
```

```
str(df_iris)
```

```
#> 'data.frame':   150 obs. of  5 variables:
#> $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
#> $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
#> $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
#> $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
#> $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1 1 1 1 1 1 ...
```

`df_iris` データは、名前のついた 5 個の変数 (列) `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, `Species` があり、それぞれ、150 の値 (observations (obs.) 観察値) からなっています。5 番目の `Species` は、三つのレベルのファクター (factor) `setosa`, `versicolor` ともう一つ `virginica` からなっていることがわかります。`unique` は、ベクトル型の変数 (今の場合は、`iris` の `Species` という列) の中の異なる値を抽出します。ファクター (factor) は、ある分け方がされているという意味です。いずれ説明します。

```
unique(df_iris$Species)
#> [1] setosa    versicolor virginica
#> Levels: setosa versicolor virginica
```

18.2.2 select 列の抽出

18.2.3 select 1 - 第 1, 2, 5 列を抽出

```
head(select(df_iris, c(1,2,5)))
#>   Sepal.Length Sepal.Width Species
#> 1           5.1           3.5  setosa
#> 2           4.9           3.0  setosa
#> 3           4.7           3.2  setosa
#> 4           4.6           3.1  setosa
#> 5           5.0           3.6  setosa
#> 6           5.4           3.9  setosa
```

第 1 列、第 2 列、第 5 列を、`c(1,2,5)` で表しています。列名で指定することもできます。

`head(select(df_iris, c(1,2,5)))` としてありますから、その最初の 6 行を表示していますが、新しい変数を割り当ててはいませんから、`df_iris` 自体は変更されていません。

```
head(df_iris)
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1           5.1           3.5           1.4           0.2  setosa
#> 2           4.9           3.0           1.4           0.2  setosa
#> 3           4.7           3.2           1.3           0.2  setosa
#> 4           4.6           3.1           1.5           0.2  setosa
#> 5           5.0           3.6           1.4           0.2  setosa
#> 6           5.4           3.9           1.7           0.4  setosa
```

第 1 列、第 2 列、第 5 列を選んだものを、あとから使いたいときは、新しい名前をつける必要があります。以下同様ですが、この場合だけ、`df_iris125` という名前 (object name) をつけておきましょう。

```
df_iris125 <- select(df_iris, c(1,2,5))
head(df_iris125)
```

```
#>   Sepal.Length Sepal.Width Species
#> 1         5.1         3.5  setosa
#> 2         4.9         3.0  setosa
#> 3         4.7         3.2  setosa
#> 4         4.6         3.1  setosa
#> 5         5.0         3.6  setosa
#> 6         5.4         3.9  setosa
```

18.2.3.1 パイプ (Pipe) |> を使ったコード

最初の例では `head(select(df_iris, c(1,2,5)))` としました。 `head` の引数として、 `select(df_iris, c(1,2,5))` を使ったからです。しかし、慣れてくると、順番に関数を適用することを表現するには、パイプを使うのも便利です。パイプでは、直前の出力が次の関数の第一引数として引き継がれるというルールになっています。以下のようになります。

```
df_iris |> select(1,2,5) |> head()
#>   Sepal.Length Sepal.Width Species
#> 1         5.1         3.5  setosa
#> 2         4.9         3.0  setosa
#> 3         4.7         3.2  setosa
#> 4         4.6         3.1  setosa
#> 5         5.0         3.6  setosa
#> 6         5.4         3.9  setosa
```

もちろん、 `select(c(1,2,5))` でも同じ結果が得られます。より、詳しく知りたい場合には、 `Help` に `|>` と入れて説明を読んでください。

In the following, we use pipes.

18.2.4 select 2 - Species 列以外

列名の前に、`-` ハイフン (半角) をつけると、その列以外を抽出します。

```
select(df_iris, -Species) |> head()
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width
#> 1         5.1         3.5         1.4         0.2
#> 2         4.9         3.0         1.4         0.2
#> 3         4.7         3.2         1.3         0.2
#> 4         4.6         3.1         1.5         0.2
#> 5         5.0         3.6         1.4         0.2
#> 6         5.4         3.9         1.7         0.4
```

18.2.5 select 3 - 列を抽出するとともに列名も変更

新しい列名 = 現在の列名という形式で、列名を変更することも可能です。

```
select(df_iris, sl = Sepal.Length, sw = Sepal.Width, sp = Species) |>
  head()
#>   sl  sw   sp
#> 1 5.1 3.5 setosa
#> 2 4.9 3.0 setosa
#> 3 4.7 3.2 setosa
#> 4 4.6 3.1 setosa
#> 5 5.0 3.6 setosa
#> 6 5.4 3.9 setosa
```

18.2.6 select 4 - 列順の変更

```
select(df_iris, c(5,3,4,1,2)) |> head()
#>   Species Petal.Length Petal.Width Sepal.Length Sepal.Width
#> 1  setosa           1.4           0.2           5.1           3.5
#> 2  setosa           1.4           0.2           4.9           3.0
#> 3  setosa           1.3           0.2           4.7           3.2
#> 4  setosa           1.5           0.2           4.6           3.1
#> 5  setosa           1.4           0.2           5.0           3.6
#> 6  setosa           1.7           0.4           5.4           3.9
```

18.2.6.1 備考

1. 上の、`select` の補助関数 (helper functions) にも書きましたように、他にも様々な方法で、列を抽出することが可能です。たくさんの列がある表に出会ったら、ぜひ活用してください。
2. 最初は、それほど大きなデータ (表) を扱うことは少ないかもしれませんが、それでも、上で紹介した、列の名前を変更したり、列の順序を変更するなどは、有益です。使いやすく、見やすい形に変えてから、作業をすることはおすすめです。

18.2.7 filter - 値による抽出

`Species` の列の値が、`virginica` であるものだけを抽出します。== とイコールが二つであることと、文字列の場合には、半角の引用符 (double quote) でくくります。'virginica' でも同じです。(列名に空白や、特殊記号が含まれているときには、back tick と呼ばれる、反対向きの引用符で括ります。あとからその必要が生じたときに説明します。)

```
filter(df_iris, Species == "virginica") %>% head()
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width
#> 1         6.3         3.3         6.0         2.5
#> 2         5.8         2.7         5.1         1.9
#> 3         7.1         3.0         5.9         2.1
#> 4         6.3         2.9         5.6         1.8
#> 5         6.5         3.0         5.8         2.2
#> 6         7.6         3.0         6.6         2.1
#>   Species
#> 1 virginica
#> 2 virginica
#> 3 virginica
#> 4 virginica
#> 5 virginica
#> 6 virginica
```

18.2.7.1 備考

- `filter` は特に重要ですが、もう少し複雑なデータ（表）を例にとって、後ほど説明します。
- 上にも少し書いた、正規表現を使えるようになると、かなり複雑な条件での、抽出も可能になります。

18.2.8 arrange - 昇順、降順

次の例では、Sepal.Length の値の、昇順（小さい順）にし、同じ値の中では、Sepal.Width の降順（大きい順（desc は descending order からとったもの））にします。Sepal.Length の値が、4.4 の部分の、Sepal.Width の値をみてください。

```
arrange(df_iris, Sepal.Length, desc(Sepal.Width)) %>% head()
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1         4.3         3.0         1.1         0.1 setosa
#> 2         4.4         3.2         1.3         0.2 setosa
#> 3         4.4         3.0         1.3         0.2 setosa
#> 4         4.4         2.9         1.4         0.2 setosa
#> 5         4.5         2.3         1.3         0.3 setosa
#> 6         4.6         3.6         1.0         0.2 setosa
```

18.2.9 mutate - rank (階級)

小さい順に順序付けて新しい列を作成し、その順序で表示します。順序付も様々な種類があります。ここでは、同じ値のときには、同じ階級にしています。下の出力を右の方を見てください。1, 2, 2, 2, 5, ... となっています。

```
df_iris |> mutate(sl_rank = min_rank(Sepal.Length)) |>
  arrange(sl_rank) |> head()
#>   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
#> 1         4.3         3.0         1.1         0.1 setosa
#> 2         4.4         2.9         1.4         0.2 setosa
#> 3         4.4         3.0         1.3         0.2 setosa
#> 4         4.4         3.2         1.3         0.2 setosa
#> 5         4.5         2.3         1.3         0.3 setosa
#> 6         4.6         3.1         1.5         0.2 setosa
#>   sl_rank
#> 1       1
#> 2       2
#> 3       2
#> 4       2
#> 5       5
#> 6       6
```

18.2.9.1 備考

1. パイプを紹介したときにも書きましたが、改行は、|> のあとでしてください。
2. mutate は、新しい列を加えるだけでなく、既存の列を修正するためにも使うことができます。修正した、データを使うときは、同じ名前のオブジェクトに -> で、アサインすることも、可能ですが、新しい名前をつけて、新しいオブジェクトとして活用することをお勧めします。

18.2.10 group_by and summarize

グループにわけて、グループごとに、平均を求めています。

```
df_iris |>
  group_by(Species) |>
  summarize(sl = mean(Sepal.Length), sw = mean(Sepal.Width),
            pl = mean(Petal.Length), pw = mean(Petal.Width))
#> # A tibble: 3 x 5
#>   Species      sl      sw      pl      pw
#>   <fct>    <dbl> <dbl> <dbl> <dbl>
```



```
#> 1 setosa      5.01  3.43  1.46  0.246
#> 2 versicolor  5.94  2.77  4.26  1.33
#> 3 virginica   6.59  2.97  5.55  2.03
```

- mean: `mean()` or `mean(x, na.rm = TRUE)` - 相加平均 (average)
- median: `median()` , `median(x, na.rm = TRUE)` - 中央値 (mid value)

以下のリンクには、もう少し例が書かれています。

dplr_iris

18.3 dplyr の参考文献

より詳しい説明は、教科書の該当箇所を見てください。

- R for Data Science, Part II Explore
- R for Data Science (2e), Transform

18.3.1 RStudio Primers: 対話型の演習問題

2. Work with Data – r4ds: Wrangle, I

- Working with Tibbles
- Isolating Data with dplyr
- Deriving Information with dplyr

18.4 例から学ぶ dplyr, II

18.4.1 df_wdi, df_wdi_extra

前の章の Tidyverse で読み込み、概観した、世界開発指標 (World Development Indicators) のデータを使います。参照: WDI のデータ

```
library(tidyverse)
library(WDI)
```

WDI の使い方は、世界銀行の部分で紹介しますが、はじめてのデータサイエンスの例でも紹介したように、データコードを利用して、データを読み込みます。ここでは、出生時の平均寿命と、一人当たりの GDP と、総人口のデータを使います。

- SP.DYN.LE00.IN: Life expectancy at birth, total (years) 出生時の平均寿命
- SP.POP.TOTL: Population, total 総人口
- NY.GDP.PCAP.KD: GDP per capita (constant 2015 US\$) 一人当たりの GDP

次のコードで読み込みます。

```
df_wdi <- WDI(
  country = "all",
  indicator = c(lifeExp = "SP.DYN.LE00.IN", pop = "SP.POP.TOTL", gdpPercap = "NY.")
)

#> Rows: 16758 Columns: 7
#> -- Column specification -----
#> Delimiter: ","
#> chr (3): country, iso2c, iso3c
#> dbl (4): year, lifeExp, pop, gdpPercap
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa

df_wdi_extra <- WDI(
  country = "all",
  indicator = c(lifeExp = "SP.DYN.LE00.IN", pop = "SP.POP.TOTL", gdpPercap = "NY.")
  extra = TRUE
)

```

すこし、追加情報を付加したのもも取得しておきます。

```
#> Rows: 16758 Columns: 15
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (6): year, lifeExp, pop, gdpPercap, longitude, lati...
#> lgl (1): status
#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa

df_wdi_extra
#> # A tibble: 16,758 x 15
#>   country      iso2c iso3c  year status lastupdated lifeExp
#>   <chr>         <chr> <chr> <dbl> <lgl> <date>      <dbl>
#> 1 Afghanistan AF    AFG   2014 NA    2023-09-19  62.5
#> 2 Afghanistan AF    AFG   2012 NA    2023-09-19  61.9
#> 3 Afghanistan AF    AFG   2009 NA    2023-09-19  60.4
#> 4 Afghanistan AF    AFG   2013 NA    2023-09-19  62.4
#> 5 Afghanistan AF    AFG   1971 NA    2023-09-19  37.9
#> 6 Afghanistan AF    AFG   2015 NA    2023-09-19  62.7
#> 7 Afghanistan AF    AFG   1969 NA    2023-09-19  36.9

```

```
#> 8 Afghanistan AF AFG 2010 NA 2023-09-19 60.9
#> 9 Afghanistan AF AFG 2011 NA 2023-09-19 61.4
#> 10 Afghanistan AF AFG 2008 NA 2023-09-19 59.9
#> # i 16,748 more rows
#> # i 8 more variables: pop <dbl>, gdpPercap <dbl>,
#> # region <chr>, capital <chr>, longitude <dbl>,
#> # latitude <dbl>, income <chr>, lending <chr>
```

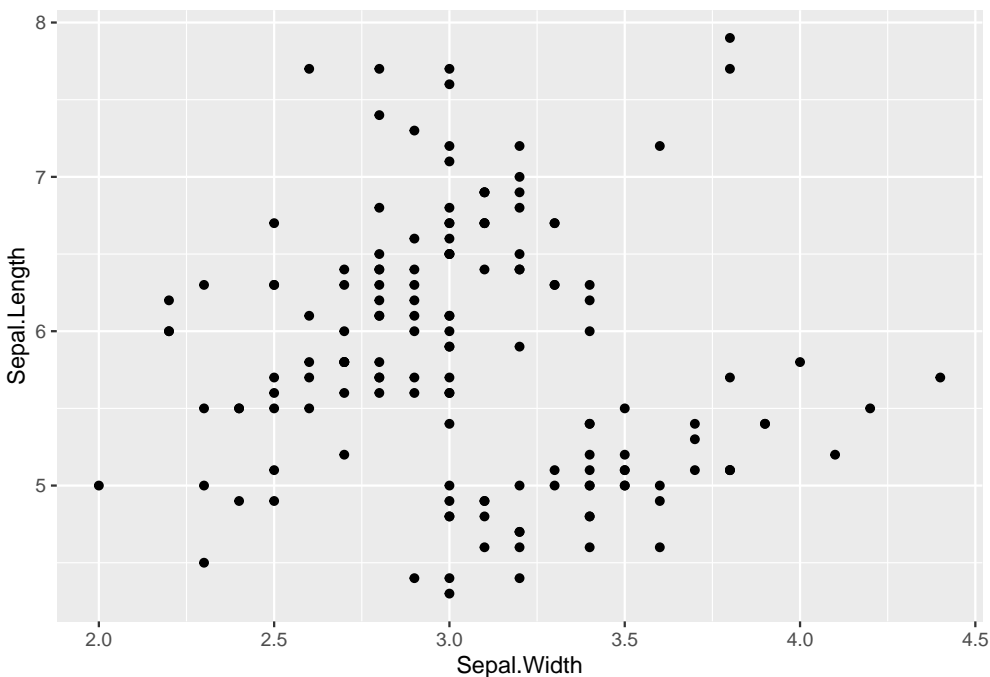
18.4.2 ggplot2

また、前の章の Tidyverse の説明で、あやめ (iris) のデータを使って、視覚化の説明を、散布図を使って、少ししました。参照：ggplot2 グラフの描画

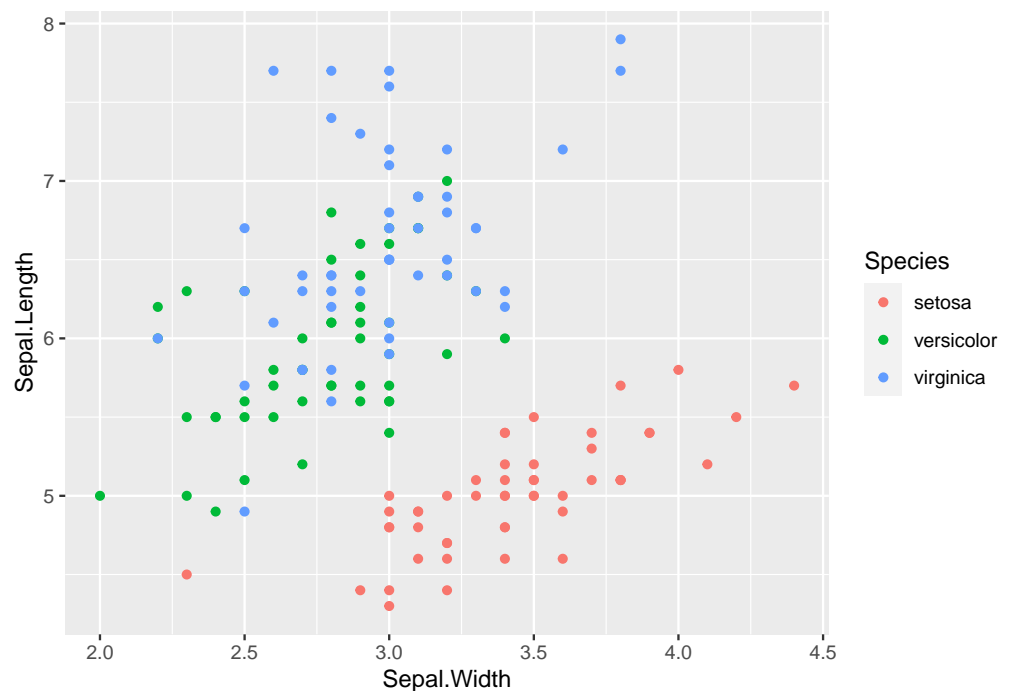
詳細は、次の、視覚化のところで説明しますが、視覚化のためには、データの変形、整理が必要であることを、理解していただくために、散布図と、箱ひげ図の基本的な場合のみ、以下のような形式で利用します。

例 1 散布図 (scatter plot)

```
df_iris |> ggplot(aes(Sepal.Width, Sepal.Length)) + geom_point()
```



```
df_iris |> ggplot(aes(Sepal.Width, Sepal.Length, color = Species)) +
  geom_point()
```



18.4.3 dplyr の応用

すべて、dplyr というわけではありませんが、データを理解していきたいと思います。

二つのデータ `df_wdi` と `df_wdi_extra` がありました。この違いをまずは見ていきましょう。colnames は、列名が並んだもの (ベクトル) を出力します。

```
df_wdi |> colnames()
#> [1] "country" "iso2c" "iso3c" "year"
#> [5] "lifeExp" "pop" "gdpPercap"
```

```
df_wdi_extra |> colnames()
#> [1] "country" "iso2c" "iso3c" "year"
#> [5] "status" "lastupdated" "lifeExp" "pop"
#> [9] "gdpPercap" "region" "capital" "longitude"
#> [13] "latitude" "income" "lending"
```

`df_wdi_extra` の一部が、`df_wdi` のようです。

ここからは、しばらく、`df_wdi_extra` をみていきます。range は、最小、最大を見るのに使います。`df_wdi_extra$year` は、`df_wdi_extra` の `year` の列を取り出したもの (ベクトルといいます) です。

```
range(df_wdi_extra$year)
#> [1] 1960 2022
```

unique を使って、その列で相異なるものをすべてリストすることもできます。

```
unique(df_wdi_extra$income)
#> [1] "Low income"          "Aggregates"
#> [3] "Upper middle income" "Lower middle income"
#> [5] "High income"         NA
#> [7] "Not classified"
```

Aggregates とか、Not classified とか、NA は何を表しているのでしょうか。あとで見てください。Google Public Data Explorer を使ったときに、Income Level というものがありました。覚えていますか。

参考：The World by Income and Region

```
unique(df_wdi_extra$region)
#> [1] "South Asia"
#> [2] "Aggregates"
#> [3] "Europe & Central Asia"
#> [4] "Middle East & North Africa"
#> [5] "East Asia & Pacific"
#> [6] "Sub-Saharan Africa"
#> [7] "Latin America & Caribbean"
#> [8] "North America"
#> [9] NA
```

これらは、あとで使いますから、残しておきましょう。そこで、country, iso2c, year, lifeExp, pop, gdpPercap, income, region だけを取り出します。列を選択するので、select を使います。

```
df_wdi3 <- df_wdi_extra |> select(country, iso2c, year, lifeExp, pop, gdpPercap, income, region)
df_wdi3
#> # A tibble: 16,758 x 8
#>   country      iso2c  year lifeExp      pop gdpPercap income
#>   <chr>        <chr> <dbl> <dbl>    <dbl> <dbl> <chr>
#> 1 Afghanistan AF      2014    62.5 32716210    603. Low i~
#> 2 Afghanistan AF      2012    61.9 30466479    596. Low i~
#> 3 Afghanistan AF      2009    60.4 27385307    512. Low i~
#> 4 Afghanistan AF      2013    62.4 31541209    608. Low i~
#> 5 Afghanistan AF      1971    37.9 11015857     NA Low i~
#> 6 Afghanistan AF      2015    62.7 33753499    592. Low i~
#> 7 Afghanistan AF      1969    36.9 10494489     NA Low i~
#> 8 Afghanistan AF      2010    60.9 28189672    569. Low i~
#> 9 Afghanistan AF      2011    61.4 29249157    551. Low i~
#> 10 Afghanistan AF     2008    59.9 26427199    437. Low i~
#> # i 16,748 more rows
#> # i 1 more variable: region <chr>
```

何列目かを指定することもできます。今回は少し多いので大変ですが。

```
df_wdi_extra |> select(1,2,4,7,8,9,10,14)
#> # A tibble: 16,758 x 8
#>   country      iso2c year lifeExp      pop gdpPercap region
#>   <chr>        <chr> <dbl> <dbl> <dbl> <dbl> <chr>
#> 1 Afghanistan AF      2014  62.5 32716210  603. South~
#> 2 Afghanistan AF      2012  61.9 30466479  596. South~
#> 3 Afghanistan AF      2009  60.4 27385307  512. South~
#> 4 Afghanistan AF      2013  62.4 31541209  608. South~
#> 5 Afghanistan AF      1971  37.9 11015857   NA South~
#> 6 Afghanistan AF      2015  62.7 33753499  592. South~
#> 7 Afghanistan AF      1969  36.9 10494489   NA South~
#> 8 Afghanistan AF      2010  60.9 28189672  569. South~
#> 9 Afghanistan AF      2011  61.4 29249157  551. South~
#> 10 Afghanistan AF      2008  59.9 26427199  437. South~
#> # i 16,748 more rows
#> # i 1 more variable: income <chr>
```

Aggregates は何を意味しているのかみてみましょう。filter で、その行だけを取り出します。

```
df_wdi3 |> filter(income == "Aggregates")
#> # A tibble: 2,646 x 8
#>   country      iso2c year lifeExp      pop gdpPercap income
#>   <chr>        <chr> <dbl> <dbl> <dbl> <dbl> <chr>
#> 1 Africa Easte~ ZH      1989  50.7 3.01e8  1367. Aggre~
#> 2 Africa Easte~ ZH      1990  50.6 3.10e8  1329. Aggre~
#> 3 Africa Easte~ ZH      1973  47.7 1.88e8  1450. Aggre~
#> 4 Africa Easte~ ZH      1974  47.6 1.94e8  1491. Aggre~
#> 5 Africa Easte~ ZH      1977  48.6 2.11e8  1448. Aggre~
#> 6 Africa Easte~ ZH      1978  48.8 2.17e8  1421. Aggre~
#> 7 Africa Easte~ ZH      1972  46.9 1.83e8  1429. Aggre~
#> 8 Africa Easte~ ZH      1987  50.1 2.84e8  1352. Aggre~
#> 9 Africa Easte~ ZH      1975  47.8 1.99e8  1473. Aggre~
#> 10 Africa Easte~ ZH      1976  48.3 2.05e8  1472. Aggre~
#> # i 2,636 more rows
#> # i 1 more variable: region <chr>
```

どうも、国ではなく、地域のようなようです。確認しておきましょう。unique を使って、

```
unique(filter(df_wdi3, income == "Aggregates")$country)
#> [1] "Africa Eastern and Southern"
#> [2] "Africa Western and Central"
#> [3] "Arab World"
#> [4] "Caribbean small states"
#> [5] "Central Europe and the Baltics"
#> [6] "Early-demographic dividend"
#> [7] "East Asia & Pacific"
#> [8] "East Asia & Pacific (excluding high income)"
#> [9] "East Asia & Pacific (IDA & IBRD countries)"
#> [10] "Euro area"
#> [11] "Europe & Central Asia"
#> [12] "Europe & Central Asia (excluding high income)"
#> [13] "Europe & Central Asia (IDA & IBRD countries)"
#> [14] "European Union"
#> [15] "Fragile and conflict affected situations"
#> [16] "Heavily indebted poor countries (HIPC)"
#> [17] "IBRD only"
#> [18] "IDA & IBRD total"
#> [19] "IDA blend"
#> [20] "IDA only"
#> [21] "IDA total"
#> [22] "Late-demographic dividend"
#> [23] "Latin America & Caribbean (excluding high income)"
#> [24] "Latin America & the Caribbean (IDA & IBRD countries)"
#> [25] "Least developed countries: UN classification"
#> [26] "Low & middle income"
#> [27] "Middle East & North Africa"
#> [28] "Middle East & North Africa (excluding high income)"
#> [29] "Middle East & North Africa (IDA & IBRD countries)"
#> [30] "Middle income"
#> [31] "North America"
#> [32] "OECD members"
#> [33] "Other small states"
#> [34] "Pacific island small states"
#> [35] "Post-demographic dividend"
#> [36] "Pre-demographic dividend"
#> [37] "Small states"
#> [38] "South Asia"
#> [39] "South Asia (IDA & IBRD)"
#> [40] "Sub-Saharan Africa (excluding high income)"
```

```
#> [41] "Sub-Saharan Africa (IDA & IBRD countries)"
#> [42] "World"
```

とする手もありますが、パイプを使うときは、そのまま、country で異なるものを取り出した表にした方が簡単なので、distinct を使って、次のようにすることもできます。

```
df_wdi3 |> filter(income == "Aggregates") |> distinct(country)
#> # A tibble: 42 x 1
#>   country
#>   <chr>
#> 1 Africa Eastern and Southern
#> 2 Africa Western and Central
#> 3 Arab World
#> 4 Caribbean small states
#> 5 Central Europe and the Baltics
#> 6 Early-demographic dividend
#> 7 East Asia & Pacific
#> 8 East Asia & Pacific (excluding high income)
#> 9 East Asia & Pacific (IDA & IBRD countries)
#> 10 Euro area
#> # i 32 more rows
```

unique を使った場合と同じ表示にしたければ、最後に pull() を付け加えて、その列だけを、並べて (ベクトル) 表示することも可能です。

```
df_wdi3 |> filter(income == "Aggregates") |> distinct(country) |> pull()
#> [1] "Africa Eastern and Southern"
#> [2] "Africa Western and Central"
#> [3] "Arab World"
#> [4] "Caribbean small states"
#> [5] "Central Europe and the Baltics"
#> [6] "Early-demographic dividend"
#> [7] "East Asia & Pacific"
#> [8] "East Asia & Pacific (excluding high income)"
#> [9] "East Asia & Pacific (IDA & IBRD countries)"
#> [10] "Euro area"
#> [11] "Europe & Central Asia"
#> [12] "Europe & Central Asia (excluding high income)"
#> [13] "Europe & Central Asia (IDA & IBRD countries)"
#> [14] "European Union"
#> [15] "Fragile and conflict affected situations"
#> [16] "Heavily indebted poor countries (HIPC)"
#> [17] "IBRD only"
```



```

#> [18] "IDA & IBRD total"
#> [19] "IDA blend"
#> [20] "IDA only"
#> [21] "IDA total"
#> [22] "Late-demographic dividend"
#> [23] "Latin America & Caribbean (excluding high income)"
#> [24] "Latin America & the Caribbean (IDA & IBRD countries)"
#> [25] "Least developed countries: UN classification"
#> [26] "Low & middle income"
#> [27] "Middle East & North Africa"
#> [28] "Middle East & North Africa (excluding high income)"
#> [29] "Middle East & North Africa (IDA & IBRD countries)"
#> [30] "Middle income"
#> [31] "North America"
#> [32] "OECD members"
#> [33] "Other small states"
#> [34] "Pacific island small states"
#> [35] "Post-demographic dividend"
#> [36] "Pre-demographic dividend"
#> [37] "Small states"
#> [38] "South Asia"
#> [39] "South Asia (IDA & IBRD)"
#> [40] "Sub-Saharan Africa (excluding high income)"
#> [41] "Sub-Saharan Africa (IDA & IBRD countries)"
#> [42] "World"

```

unclassified と、NA もみておきましょう。まず、Aggregates 以外を見たいので、等しくないは、!= を使います。

```

df_wdi3 |> filter(income != "Aggregates") |> distinct(income)
#> # A tibble: 5 x 1
#>   income
#>   <chr>
#> 1 Low income
#> 2 Upper middle income
#> 3 Lower middle income
#> 4 High income
#> 5 Not classified

```

NA もなくなりましたから、あとは、Not classified ですね。

```

df_wdi3 |> filter(income != "Aggregates") |>
  filter(income == "Not classified") |> distinct(country)

```

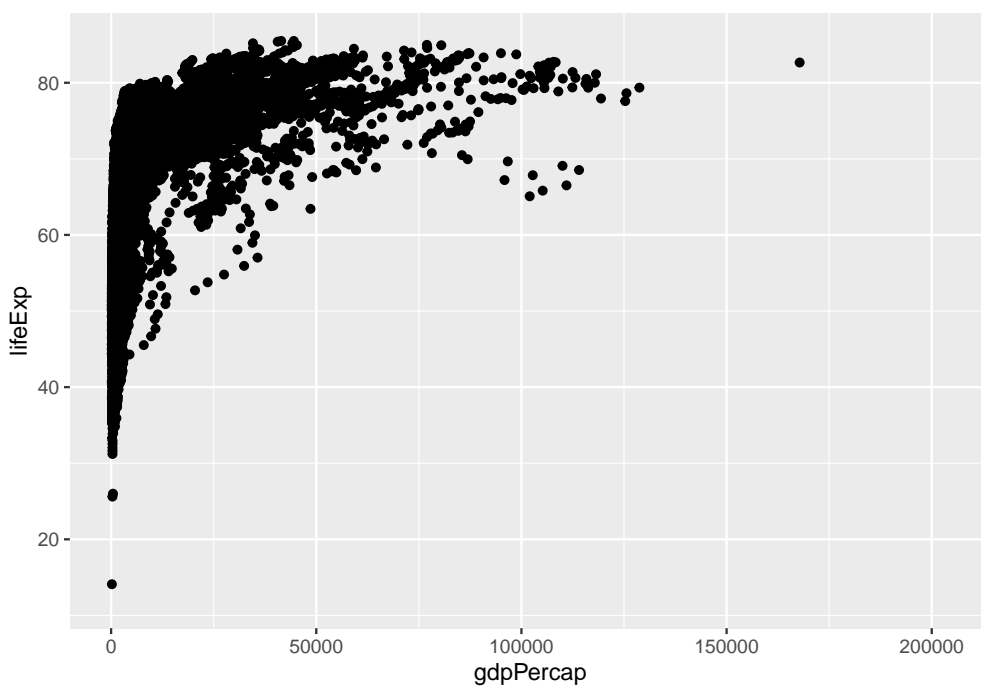
```
#> # A tibble: 1 x 1
#>   country
#>   <chr>
#> 1 Venezuela, RB
```

ベネズエラ (Venezuela, RB) は、経済が安定していないので、このようになるようです。これは、残しておいて、Aggregates でないものだけ選択しましょう。通常の、country のみですから、df_wdi3c としておきます。

```
df_wdi3c <- df_wdi3 |> filter(income != "Aggregates")
df_wdi3c
#> # A tibble: 13,608 x 8
#>   country      iso2c year lifeExp      pop gdpPercap income
#>   <chr>        <chr> <dbl> <dbl>    <dbl>    <dbl> <chr>
#> 1 Afghanistan AF      2014    62.5 32716210    603. Low i~
#> 2 Afghanistan AF      2012    61.9 30466479    596. Low i~
#> 3 Afghanistan AF      2009    60.4 27385307    512. Low i~
#> 4 Afghanistan AF      2013    62.4 31541209    608. Low i~
#> 5 Afghanistan AF      1971    37.9 11015857     NA Low i~
#> 6 Afghanistan AF      2015    62.7 33753499    592. Low i~
#> 7 Afghanistan AF      1969    36.9 10494489     NA Low i~
#> 8 Afghanistan AF      2010    60.9 28189672    569. Low i~
#> 9 Afghanistan AF      2011    61.4 29249157    551. Low i~
#> 10 Afghanistan AF      2008    59.9 26427199    437. Low i~
#> # i 13,598 more rows
#> # i 1 more variable: region <chr>
```

散布図を書いてみましょう。

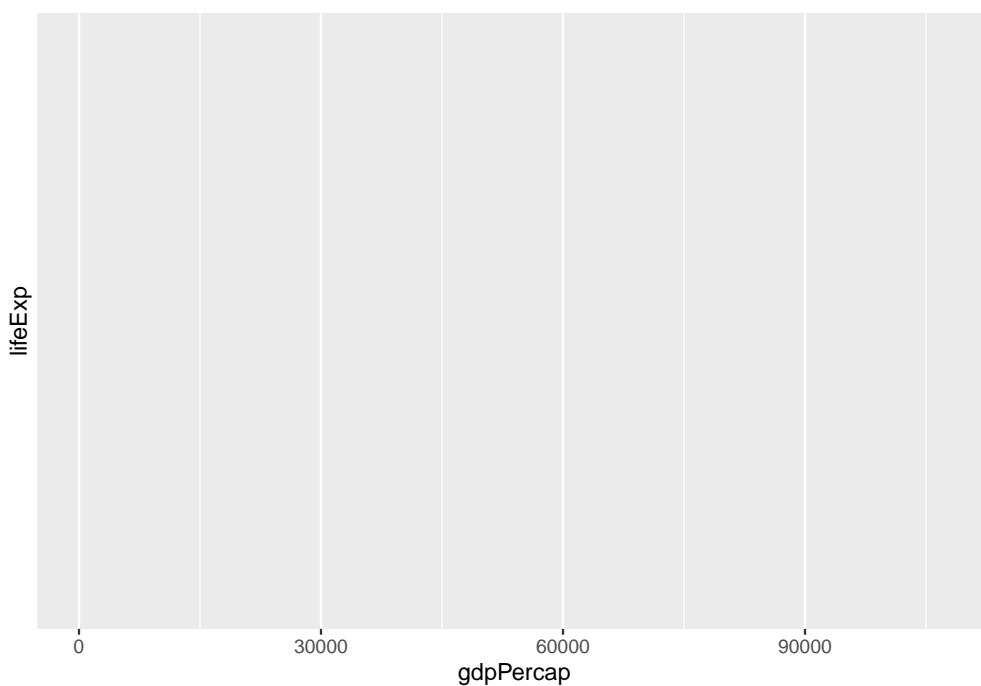
```
df_wdi3c |> ggplot(aes(gdpPercap, lifeExp)) + geom_point()
#> Warning: Removed 3981 rows containing missing values
#> (`geom_point()`).
```



一応、なにか描けましたが、どうも、すごく点の数が多いですね。それと、欠損値もたくさんあるようです。また、gdpPercap でみると、値が小さいところに固まってあるようです。

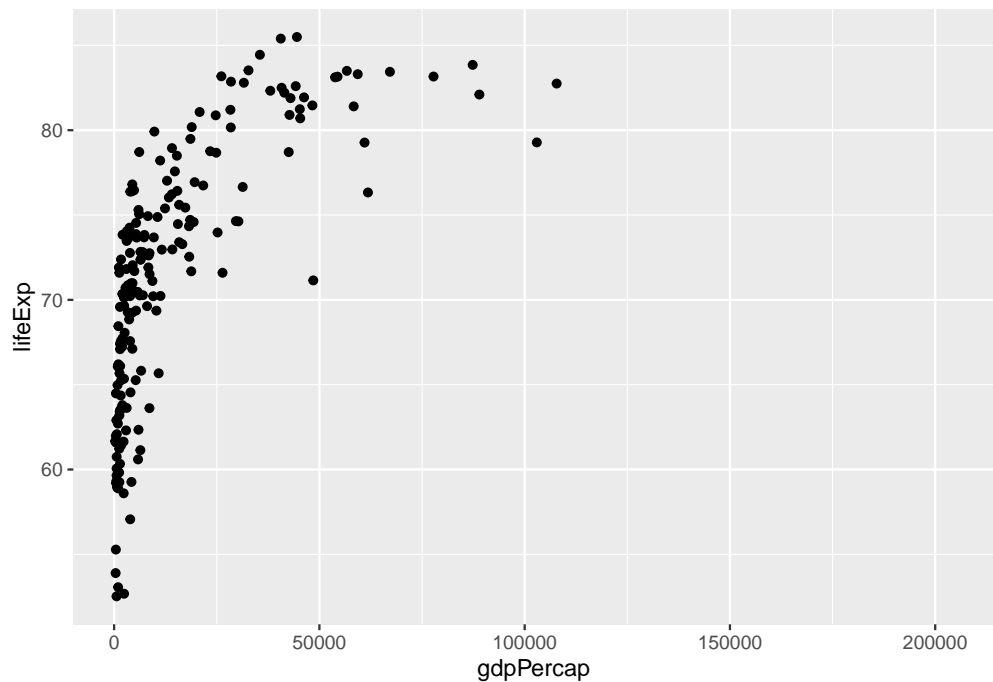
まずは、すべての年について、値を使っていますから、2022 年に限ってみましょう。

```
df_wdi3c |> filter(year == "2022") |>  
  ggplot(aes(gdpPercap, lifeExp)) + geom_point()  
#> Warning: Removed 216 rows containing missing values  
#> (`geom_point()`).
```



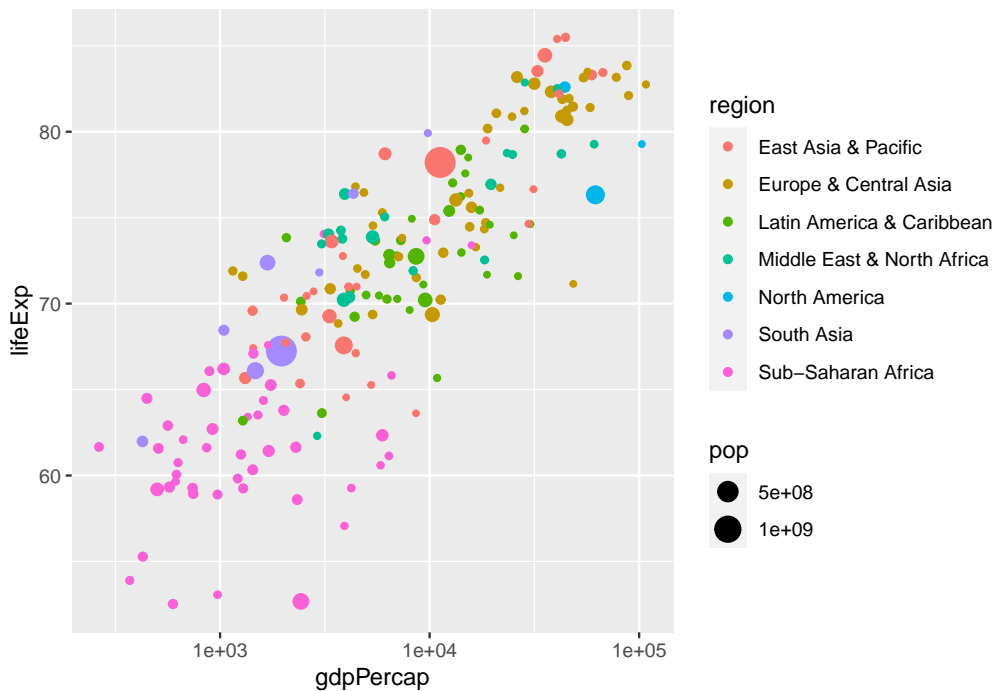
どうも、何も現れません。2021年ならどうでしょうか。

```
df_wdi3c |> filter(year == 2021) |>  
  ggplot(aes(gdpPercap, lifeExp)) + geom_point()  
#> Warning: Removed 23 rows containing missing values  
#> (`geom_point()`).
```



データがない国もあるようですが、どうやら表示されました。データがない国は、削除しておいたり region を色で表したり、人口を、点の大きさと表したり、gdpPercap を log10 をとって、値を修正して表すなども可能ですが、それは、可視化の項目で学びます。

```
df_wdi3c |> filter(year == 2021) |> drop_na(lifeExp, gdpPercap) |>  
  ggplot(aes(gdpPercap, lifeExp, color = region, size = pop)) +  
  geom_point() + scale_x_log10()
```



こうしてみると、gdpPerCap が大きい国ほど、lifeExp も大きい傾向があることも見えますね。

ここまでは、`select` と、`filter` を使ってきましたが、`arrange`、`mutate`、`group_by` や、`summarize` も使ってみましょう。

まず、gdpPerCap は一人当たりの GDP でしたから、この値に、pop 人口をかければ、原理的には、GDP が得られることになります。（さまざまな基準がありますから、詳細は、調べてください。）順番を、入れ替えて、year の次に、gdp がくるようにしておきます。他にも、year の後に置くようにとの指示もできます。Help で確認してください。例も載っています。また、掛け算は、* を使います。これを、df_wdi4c としておきましょう。また、year が順番になっていないのが気になりますから、大きい順に並べ替えておきましょう。

```
df_wdi4c <- df_wdi3c |> mutate(gdp = pop*gdpPerCap) |>
  select(1,2,3,9,4,5,6,7,8) |> arrange(country, desc(year))
df_wdi4c
#> # A tibble: 13,608 x 9
#>   country iso2c year      gdp lifeExp  pop gdpPerCap
#>   <chr>    <chr> <dbl>   <dbl> <dbl> <dbl> <dbl>
#> 1 Afghanistan AF      2022 NA         NA  4.11e7      NA
#> 2 Afghanistan AF      2021 1.71e10 62.0 4.01e7    426.
#> 3 Afghanistan AF      2020 2.16e10 62.6 3.90e7    553.
#> 4 Afghanistan AF      2019 2.21e10 63.6 3.78e7    584.
#> 5 Afghanistan AF      2018 2.12e10 63.1 3.67e7    579.
#> 6 Afghanistan AF      2017 2.10e10 63.0 3.56e7    589.
#> 7 Afghanistan AF      2016 2.05e10 63.1 3.46e7    590.
```

```
#> 8 Afghanistan AF      2015  2.00e10   62.7 3.38e7   592.
#> 9 Afghanistan AF      2014  1.97e10   62.5 3.27e7   603.
#> 10 Afghanistan AF     2013  1.92e10   62.4 3.15e7   608.
#> # i 13,598 more rows
#> # i 2 more variables: income <chr>, region <chr>
```

```
df_wdi3c |> mutate(gdp = pop*gdpPercap, .after = year) |>
  arrange(country, desc(year))
#> # A tibble: 13,608 x 9
#>   country      iso2c year      gdp lifeExp  pop gdpPercap
#>   <chr>        <chr> <dbl>   <dbl> <dbl> <dbl> <dbl>
#> 1 Afghanistan AF      2022 NA           NA  4.11e7      NA
#> 2 Afghanistan AF      2021  1.71e10   62.0 4.01e7   426.
#> 3 Afghanistan AF      2020  2.16e10   62.6 3.90e7   553.
#> 4 Afghanistan AF      2019  2.21e10   63.6 3.78e7   584.
#> 5 Afghanistan AF      2018  2.12e10   63.1 3.67e7   579.
#> 6 Afghanistan AF      2017  2.10e10   63.0 3.56e7   589.
#> 7 Afghanistan AF      2016  2.05e10   63.1 3.46e7   590.
#> 8 Afghanistan AF      2015  2.00e10   62.7 3.38e7   592.
#> 9 Afghanistan AF      2014  1.97e10   62.5 3.27e7   603.
#> 10 Afghanistan AF     2013  1.92e10   62.4 3.15e7   608.
#> # i 13,598 more rows
#> # i 2 more variables: income <chr>, region <chr>
```

2022年のデータはあまりないようですから、2021年のGDPの大きい順に並べてみましょう。gdpが 2.052946e13 などと書かれているのは、指数表示 (Scientific Notation) で、e3は、1000千、e6は、1,000,000百万、e9は、1,000,000,000千億、e12は、1,000,000,000,000超です。つまり、アメリカのGDPは、だいたい、20兆ドル、日本は、4.5兆ドルと言ったところです。

```
df_wdi4c |> filter(year == 2021) |> arrange(desc(gdp))
#> # A tibble: 216 x 9
#>   country      iso2c year      gdp lifeExp  pop gdpPercap
#>   <chr>        <chr> <dbl>   <dbl> <dbl> <dbl> <dbl>
#> 1 United Stat~ US      2021  2.05e13   76.3 3.32e8  61830.
#> 2 China        CN      2021  1.59e13   78.2 1.41e9  11223.
#> 3 Japan        JP      2021  4.46e12   84.4 1.26e8  35508.
#> 4 Germany      DE      2021  3.55e12   80.9 8.32e7  42726.
#> 5 United King~ GB      2021  3.04e12   80.7 6.70e7  45334.
#> 6 India        IN      2021  2.76e12   67.2 1.41e9  1962.
#> 7 France       FR      2021  2.58e12   82.3 6.77e7  38046.
#> 8 Italy         IT      2021  1.87e12   82.8 5.91e7  31601.
```

```
#> 9 Brazil BR 2021 1.85e12 72.8 2.14e8 8622.
#> 10 Korea, Rep. KR 2021 1.69e12 83.5 5.17e7 32731.
#> # i 206 more rows
#> # i 2 more variables: income <chr>, region <chr>
```

地域ごとの平均寿命を求めてみましょう。

```
df_wdi4c |> drop_na(lifeExp) |>
  group_by(region) |> summarize(lifeexpregion = mean(lifeExp)) |>
  arrange(desc(lifeexpregion))
#> # A tibble: 7 x 2
#>   region                lifeexpregion
#>   <chr>                  <dbl>
#> 1 North America          75.6
#> 2 Europe & Central Asia  72.2
#> 3 Latin America & Caribbean 68.3
#> 4 Middle East & North Africa 66.2
#> 5 East Asia & Pacific     66.0
#> 6 South Asia             58.1
#> 7 Sub-Saharan Africa     52.1
```

income でも計算してみましょう。

```
df_wdi4c |> drop_na(lifeExp) |>
  group_by(income) |> summarize(lifeexpincome = mean(lifeExp)) |>
  arrange(desc(lifeexpincome))
#> # A tibble: 5 x 2
#>   income                lifeexpincome
#>   <chr>                  <dbl>
#> 1 High income           73.4
#> 2 Not classified        69.2
#> 3 Upper middle income   66.4
#> 4 Lower middle income   58.8
#> 5 Low income            50.1
```

ある年の値がないためにグラフがかけない場合もあります。各指標の年毎のデータがどれくらいあるかみてみましょう。is.na(x) は、x が NA なら TRUE (値は 1)、x が NA でなければ、FALSE (値は 0) を返します。!is.na(x) はその否定ですから、逆になります。そこで、たとえば、sum(!is.na(gdp)) は、gdp という列 (年毎にグループ分けした) の中の、NA ではないものがいくつあるか数えることになります。sum は summation で、合計です。

```
df_wdi4c |> group_by(year) |>
  summarize(gdp_n = sum(!is.na(gdp)), lifeExp_n = sum(!is.na(lifeExp)), pop_n = sum(!is.na(pop)), gdpPer
```

```

#> # A tibble: 63 x 5
#>   year gdp_n lifeExp_n pop_n gdpPercap_n
#>   <dbl> <int>   <int> <int>   <int>
#> 1  2022   184     0   216     184
#> 2  2021   200    208   216     200
#> 3  2020   205    208   216     205
#> 4  2019   205    208   216     205
#> 5  2018   206    208   216     206
#> 6  2017   206    209   216     206
#> 7  2016   206    209   216     206
#> 8  2015   209    209   216     209
#> 9  2014   207    209   216     207
#> 10 2013   207    209   216     207
#> # i 53 more rows

```

18.4.4 filter

一番上に、国名 (country) に、アフガニスタン (Afghanistan) がありましたから、アフガニスタンのデータを選び、平均寿命 (lifeExp: Life Expectancy) の折線グラフ (line graph) を描いてみましょう。まずは、filter で、アフガニスタンのデータを抽出します。スペルに注意してください。コピーをするのが安全かもしれません。フ (line graph) を描いてみましょう。まずは、filter で、アフガニスタンのデータを抽出します。スペルに注意してください。コピーをするのが安全かもしれません。

```
filter(country == "Afghanistan")
```

特定の値のデータを抽出するには、== を使うのでした。二つの = ですから、間違わないでください。

```

df_wdi |> filter(country == "Afghanistan")
#> # A tibble: 63 x 7
#>   country iso2c iso3c year lifeExp pop gdpPercap
#>   <chr>   <chr> <chr> <dbl> <dbl> <dbl> <dbl>
#> 1 Afghanistan AF AFG 1960 32.5 8622466 NA
#> 2 Afghanistan AF AFG 1961 33.1 8790140 NA
#> 3 Afghanistan AF AFG 1962 33.5 8969047 NA
#> 4 Afghanistan AF AFG 1963 34.0 9157465 NA
#> 5 Afghanistan AF AFG 1964 34.5 9355514 NA
#> 6 Afghanistan AF AFG 1965 35.0 9565147 NA
#> 7 Afghanistan AF AFG 1966 35.5 9783147 NA
#> 8 Afghanistan AF AFG 1967 35.9 10010030 NA
#> 9 Afghanistan AF AFG 1968 36.4 10247780 NA
#> 10 Afghanistan AF AFG 1969 36.9 10494489 NA

```

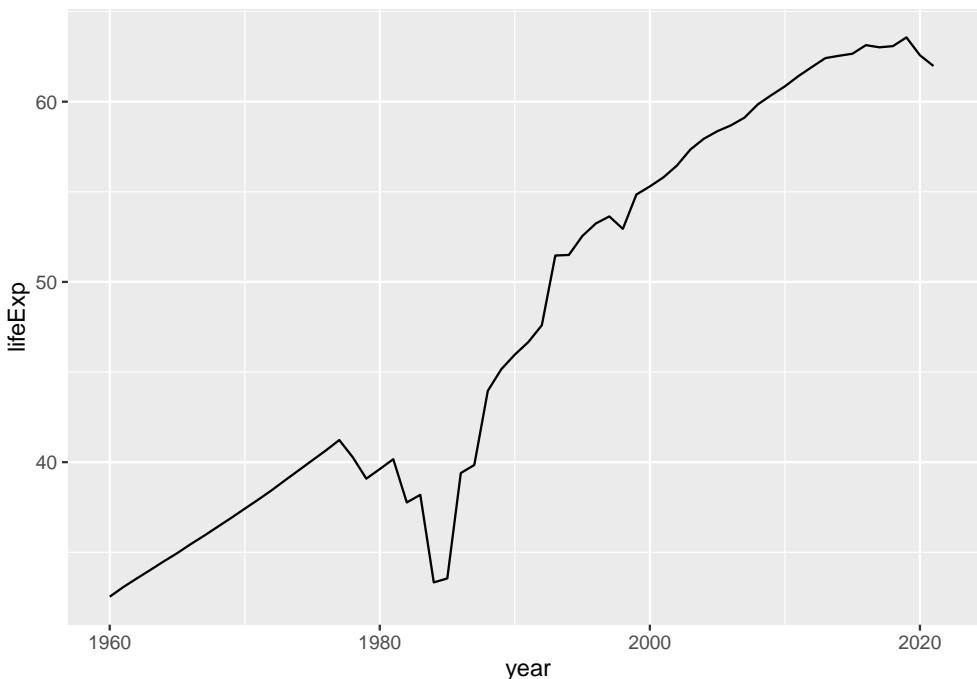


```
#> # i 53 more rows
```

確認できましたか。

折線グラフを書きます。この場合は、GEOM は、`geom_line` です。

```
df_wdi |> filter(country == "Afghanistan") %>%
  ggplot(aes(x = year, y = lifeExp)) + geom_line()
#> Warning: Removed 1 row containing missing values
#> (`geom_line()`).
```



アフガニスタンでは 1952 年の誕生時の平均寿命 (life expectancy at birth) は 30 歳以下 (28.8 歳) でした。2007 年でも 50 歳以下 (48.8 歳) のようですね。改善されていることも確かです。

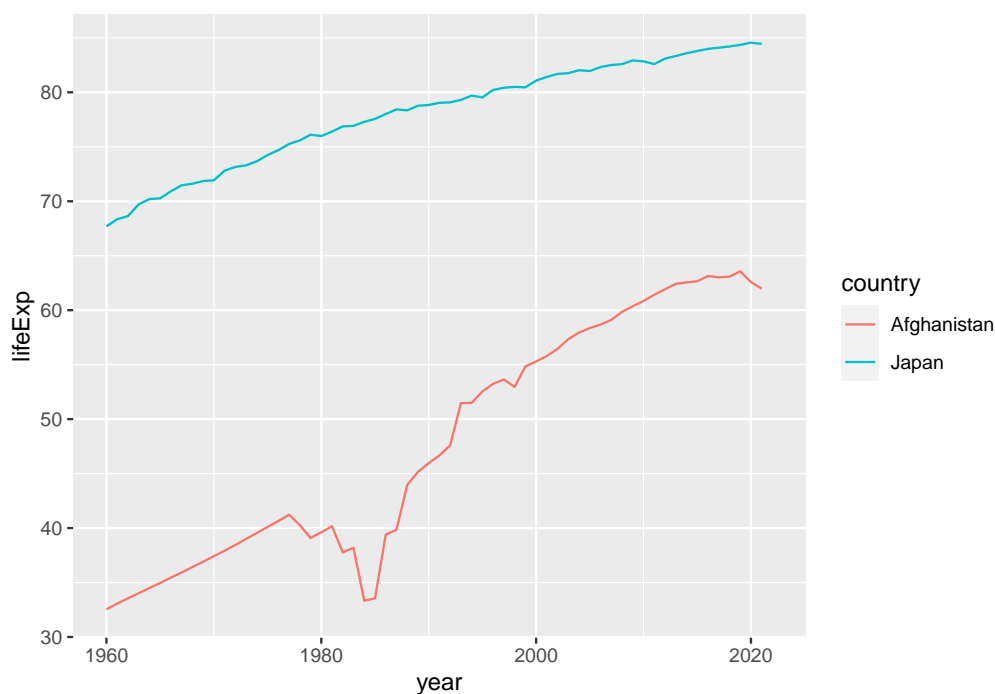
アフガニスタンと日本両方を抽出してみましょう。そのときは、`country %in% c("Afghanistan", "Japan")` とします。

```
df_wdi %>% filter(country %in% c("Afghanistan", "Japan"))
#> # A tibble: 126 x 7
#>   country iso2c iso3c year lifeExp pop gdpPercap
#>   <chr>     <chr> <chr> <dbl> <dbl> <dbl> <dbl>
#> 1 Afghanistan AF AFG 1960 32.5 8622466 NA
#> 2 Afghanistan AF AFG 1961 33.1 8790140 NA
#> 3 Afghanistan AF AFG 1962 33.5 8969047 NA
#> 4 Afghanistan AF AFG 1963 34.0 9157465 NA
#> 5 Afghanistan AF AFG 1964 34.5 9355514 NA
#> 6 Afghanistan AF AFG 1965 35.0 9565147 NA
```

```
#> 7 Afghanistan AF AFG 1966 35.5 9783147 NA
#> 8 Afghanistan AF AFG 1967 35.9 10010030 NA
#> 9 Afghanistan AF AFG 1968 36.4 10247780 NA
#> 10 Afghanistan AF AFG 1969 36.9 10494489 NA
#> # i 116 more rows
```

グラフにしてみましょう。今度は、区別のため、`color = country` を追加します。すると、線が違う色で表示されます。詳しくは、視覚化を参照してください。

```
df_wdi %>% filter(country %in% c("Afghanistan", "Japan")) %>%
  ggplot(aes(x = year, y = lifeExp, color = country)) + geom_line()
#> Warning: Removed 2 rows containing missing values
#> (`geom_line()`).
```



どのような発見がありますか。かならず書き留めておいてください。

他の国についても調べるときは、国のリストがあるとよいので、`unique(df_gm$country)` とすると、リストが得られます。`distinct(country)` で、異なる国を選択してから、その部分をベクトルとして出力することもできます。

```
df_wdi %>% distinct(country) %>% pull()
#> [1] "Afghanistan"
#> [2] "Africa Eastern and Southern"
#> [3] "Africa Western and Central"
#> [4] "Albania"
#> [5] "Algeria"
#> [6] "American Samoa"
```

```
#> [7] "Andorra"  
#> [8] "Angola"  
#> [9] "Antigua and Barbuda"  
#> [10] "Arab World"  
#> [11] "Argentina"  
#> [12] "Armenia"  
#> [13] "Aruba"  
#> [14] "Australia"  
#> [15] "Austria"  
#> [16] "Azerbaijan"  
#> [17] "Bahamas, The"  
#> [18] "Bahrain"  
#> [19] "Bangladesh"  
#> [20] "Barbados"  
#> [21] "Belarus"  
#> [22] "Belgium"  
#> [23] "Belize"  
#> [24] "Benin"  
#> [25] "Bermuda"  
#> [26] "Bhutan"  
#> [27] "Bolivia"  
#> [28] "Bosnia and Herzegovina"  
#> [29] "Botswana"  
#> [30] "Brazil"  
#> [31] "British Virgin Islands"  
#> [32] "Brunei Darussalam"  
#> [33] "Bulgaria"  
#> [34] "Burkina Faso"  
#> [35] "Burundi"  
#> [36] "Cabo Verde"  
#> [37] "Cambodia"  
#> [38] "Cameroon"  
#> [39] "Canada"  
#> [40] "Caribbean small states"  
#> [41] "Cayman Islands"  
#> [42] "Central African Republic"  
#> [43] "Central Europe and the Baltics"  
#> [44] "Chad"  
#> [45] "Channel Islands"  
#> [46] "Chile"  
#> [47] "China"
```

```
#> [48] "Colombia"
#> [49] "Comoros"
#> [50] "Congo, Dem. Rep."
#> [51] "Congo, Rep."
#> [52] "Costa Rica"
#> [53] "Cote d'Ivoire"
#> [54] "Croatia"
#> [55] "Cuba"
#> [56] "Curacao"
#> [57] "Cyprus"
#> [58] "Czechia"
#> [59] "Denmark"
#> [60] "Djibouti"
#> [61] "Dominica"
#> [62] "Dominican Republic"
#> [63] "Early-demographic dividend"
#> [64] "East Asia & Pacific"
#> [65] "East Asia & Pacific (excluding high income)"
#> [66] "East Asia & Pacific (IDA & IBRD countries)"
#> [67] "Ecuador"
#> [68] "Egypt, Arab Rep."
#> [69] "El Salvador"
#> [70] "Equatorial Guinea"
#> [71] "Eritrea"
#> [72] "Estonia"
#> [73] "Eswatini"
#> [74] "Ethiopia"
#> [75] "Euro area"
#> [76] "Europe & Central Asia"
#> [77] "Europe & Central Asia (excluding high income)"
#> [78] "Europe & Central Asia (IDA & IBRD countries)"
#> [79] "European Union"
#> [80] "Faroe Islands"
#> [81] "Fiji"
#> [82] "Finland"
#> [83] "Fragile and conflict affected situations"
#> [84] "France"
#> [85] "French Polynesia"
#> [86] "Gabon"
#> [87] "Gambia, The"
#> [88] "Georgia"
```

```
#> [89] "Germany"  
#> [90] "Ghana"  
#> [91] "Gibraltar"  
#> [92] "Greece"  
#> [93] "Greenland"  
#> [94] "Grenada"  
#> [95] "Guam"  
#> [96] "Guatemala"  
#> [97] "Guinea"  
#> [98] "Guinea-Bissau"  
#> [99] "Guyana"  
#> [100] "Haiti"  
#> [101] "Heavily indebted poor countries (HIPC)"  
#> [102] "High income"  
#> [103] "Honduras"  
#> [104] "Hong Kong SAR, China"  
#> [105] "Hungary"  
#> [106] "IBRD only"  
#> [107] "Iceland"  
#> [108] "IDA & IBRD total"  
#> [109] "IDA blend"  
#> [110] "IDA only"  
#> [111] "IDA total"  
#> [112] "India"  
#> [113] "Indonesia"  
#> [114] "Iran, Islamic Rep."  
#> [115] "Iraq"  
#> [116] "Ireland"  
#> [117] "Isle of Man"  
#> [118] "Israel"  
#> [119] "Italy"  
#> [120] "Jamaica"  
#> [121] "Japan"  
#> [122] "Jordan"  
#> [123] "Kazakhstan"  
#> [124] "Kenya"  
#> [125] "Kiribati"  
#> [126] "Korea, Dem. People's Rep."  
#> [127] "Korea, Rep."  
#> [128] "Kosovo"  
#> [129] "Kuwait"
```

```
#> [130] "Kyrgyz Republic"
#> [131] "Lao PDR"
#> [132] "Late-demographic dividend"
#> [133] "Latin America & Caribbean"
#> [134] "Latin America & Caribbean (excluding high income)"
#> [135] "Latin America & the Caribbean (IDA & IBRD countries)"
#> [136] "Latvia"
#> [137] "Least developed countries: UN classification"
#> [138] "Lebanon"
#> [139] "Lesotho"
#> [140] "Liberia"
#> [141] "Libya"
#> [142] "Liechtenstein"
#> [143] "Lithuania"
#> [144] "Low & middle income"
#> [145] "Low income"
#> [146] "Lower middle income"
#> [147] "Luxembourg"
#> [148] "Macao SAR, China"
#> [149] "Madagascar"
#> [150] "Malawi"
#> [151] "Malaysia"
#> [152] "Maldives"
#> [153] "Mali"
#> [154] "Malta"
#> [155] "Marshall Islands"
#> [156] "Mauritania"
#> [157] "Mauritius"
#> [158] "Mexico"
#> [159] "Micronesia, Fed. Sts."
#> [160] "Middle East & North Africa"
#> [161] "Middle East & North Africa (excluding high income)"
#> [162] "Middle East & North Africa (IDA & IBRD countries)"
#> [163] "Middle income"
#> [164] "Moldova"
#> [165] "Monaco"
#> [166] "Mongolia"
#> [167] "Montenegro"
#> [168] "Morocco"
#> [169] "Mozambique"
#> [170] "Myanmar"
```

```
#> [171] "Namibia"  
#> [172] "Nauru"  
#> [173] "Nepal"  
#> [174] "Netherlands"  
#> [175] "New Caledonia"  
#> [176] "New Zealand"  
#> [177] "Nicaragua"  
#> [178] "Niger"  
#> [179] "Nigeria"  
#> [180] "North America"  
#> [181] "North Macedonia"  
#> [182] "Northern Mariana Islands"  
#> [183] "Norway"  
#> [184] "Not classified"  
#> [185] "OECD members"  
#> [186] "Oman"  
#> [187] "Other small states"  
#> [188] "Pacific island small states"  
#> [189] "Pakistan"  
#> [190] "Palau"  
#> [191] "Panama"  
#> [192] "Papua New Guinea"  
#> [193] "Paraguay"  
#> [194] "Peru"  
#> [195] "Philippines"  
#> [196] "Poland"  
#> [197] "Portugal"  
#> [198] "Post-demographic dividend"  
#> [199] "Pre-demographic dividend"  
#> [200] "Puerto Rico"  
#> [201] "Qatar"  
#> [202] "Romania"  
#> [203] "Russian Federation"  
#> [204] "Rwanda"  
#> [205] "Samoa"  
#> [206] "San Marino"  
#> [207] "Sao Tome and Principe"  
#> [208] "Saudi Arabia"  
#> [209] "Senegal"  
#> [210] "Serbia"  
#> [211] "Seychelles"
```

```
#> [212] "Sierra Leone"  
#> [213] "Singapore"  
#> [214] "Sint Maarten (Dutch part)"  
#> [215] "Slovak Republic"  
#> [216] "Slovenia"  
#> [217] "Small states"  
#> [218] "Solomon Islands"  
#> [219] "Somalia"  
#> [220] "South Africa"  
#> [221] "South Asia"  
#> [222] "South Asia (IDA & IBRD)"  
#> [223] "South Sudan"  
#> [224] "Spain"  
#> [225] "Sri Lanka"  
#> [226] "St. Kitts and Nevis"  
#> [227] "St. Lucia"  
#> [228] "St. Martin (French part)"  
#> [229] "St. Vincent and the Grenadines"  
#> [230] "Sub-Saharan Africa"  
#> [231] "Sub-Saharan Africa (excluding high income)"  
#> [232] "Sub-Saharan Africa (IDA & IBRD countries)"  
#> [233] "Sudan"  
#> [234] "Suriname"  
#> [235] "Sweden"  
#> [236] "Switzerland"  
#> [237] "Syrian Arab Republic"  
#> [238] "Tajikistan"  
#> [239] "Tanzania"  
#> [240] "Thailand"  
#> [241] "Timor-Leste"  
#> [242] "Togo"  
#> [243] "Tonga"  
#> [244] "Trinidad and Tobago"  
#> [245] "Tunisia"  
#> [246] "Turkiye"  
#> [247] "Turkmenistan"  
#> [248] "Turks and Caicos Islands"  
#> [249] "Tuvalu"  
#> [250] "Uganda"  
#> [251] "Ukraine"  
#> [252] "United Arab Emirates"
```

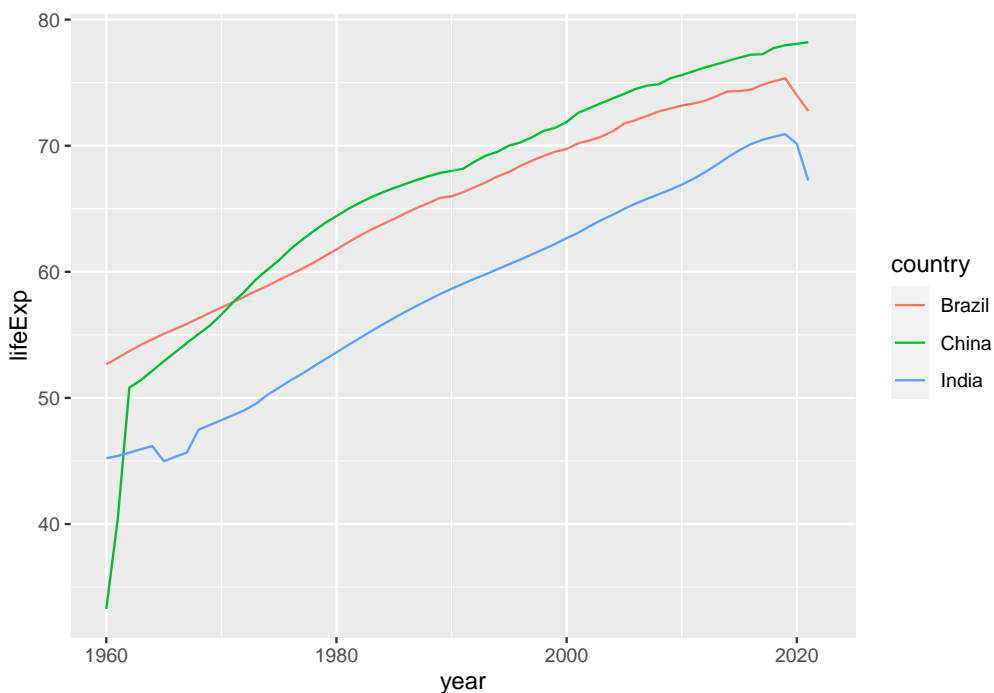


```
#> [253] "United Kingdom"
#> [254] "United States"
#> [255] "Upper middle income"
#> [256] "Uruguay"
#> [257] "Uzbekistan"
#> [258] "Vanuatu"
#> [259] "Venezuela, RB"
#> [260] "Vietnam"
#> [261] "Virgin Islands (U.S.)"
#> [262] "West Bank and Gaza"
#> [263] "World"
#> [264] "Yemen, Rep."
#> [265] "Zambia"
#> [266] "Zimbabwe"
```

このデータには 142 の国のデータがあることがわかりました。

BRICs を選んでみるとどうなるでしょうか。最近では、BRICS として、South Africa を加えることが増えてきているようです。

```
df_wdi %>% filter(country %in% c("Brazil", "Russia", "India", "China")) %>%
  ggplot(aes(x = year, y = lifeExp, color = country)) + geom_line()
#> Warning: Removed 3 rows containing missing values
#> (`geom_line()`).
```



ロシアが含まれていないことがわかります。ロシアは、以前は、ソビエト社会主義連邦で

したから、国が変化したものは含まれていないのかもしれませんが。上の国のリストで見てもありませんね。2007年より新しいデータ、ロシアなども含むデータなど、実際のデータでも見てみたいですね。それは、また後ほど。

18.4.5 練習

1. 平均寿命 `lifeExp` を人口 `pop` や、一人当たりの GDP `gdpPercap` に変えて、試してみてください。
2. ASEAN（東南アジア諸国連合）ではどうでしょうか。
 - Brunei, Cambodia, Indonesia, Laos, Malaysia, Myanmar, Philippines, Singapore.
 - このうち幾つの国がこのデータに含まれていますか。
3. 興味のある国をいくつか選んで、三つの指標について調べてみてください。

18.4.6 `group_by` と `summarize`

データには大陸 (`continent`) という変数があります。幾つの大陸があり、それぞれの大陸のいくつ国のデータがこのデータには入っているのでしょうか。

それぞれの大陸ごとの 2007 年の平均寿命の平均と中央値と最大、最小を求めてみましょう。

```
df_wdi_extra %>% filter(year == 2007) %>%
  group_by(income) %>%
  summarize(mean_lifeExp = mean(lifeExp), median_lifeExp = median(lifeExp), max_l
```

```
#> # A tibble: 7 x 5
#> # Groups:   income [7]
#>   income mean_lifeExp median_lifeExp max_lifeExp min_lifeExp
#>   <chr>      <dbl>         <dbl>         <dbl>         <dbl>
#> 1 Aggre~      67.6           69.2           80.1           53.3
#> 2 High ~      NA              NA              NA              NA
#> 3 Low i~      56.9           55.4           73.7           47.4
#> 4 Lower~      64.4           65.9           77.1           43.1
#> 5 Not c~      73.1           73.1           73.1           73.1
#> 6 Upper~      NA              NA              NA              NA
#> 7 <NA>       NA              NA              NA              NA
```

18.5 練習問題

R Markdown and dplyr

- Create an R Notebook of a Data Analysis containing the following and submit the rendered HTML file (eg. `a2_123456.nb.html`)
 1. create an R Notebook using the R Notebook Template in Moodle, save as `a2_123456.Rmd`,
 2. write your name and ID and the contents,
 3. run each code block,
 4. preview to create `a2_123456.nb.html`,
 5. submit `a2_123456.nb.html` to Moodle.
- 1. Pick data from the built-in datasets besides `cars`. (`library(help = "datasets")` or go to the site The R Datasets Package)
 - Information of the data: Name, Description, Usage, Format, Source, References (Hint: `?cars`)
 - Use `head()`, `str()`, `...`, and create at least one chart using `ggplot2` - Code Chunk.
 - Don't forget to add `library(tidyverse)` in the first code chunk.
 - An observation of the chart - in your own words.
- 2. Load `gapminder` by `library(gapminder)`.
 - Choose `pop` or `gdpPercap`, or both, one country in the data, a group of countries in the data.
 - Create charts using `ggplot2` with `geom_line` and the variables and countries chosen in 1. (See examples of the charts for `lifeExp`.)
 - Study the data as you like.
 - Observations and difficulties encountered.

演習

18.5.0.1 Gapminder と R Package `gapminder`

Gapminder はオラロスリング (Ola Rosling) とアンナロスリング (Anna Rosling Rönnlund) とハンスロスリング (Hans Rosling) が設立した組織です。

ハンス・ロスリングは、**FACTFULNESS (ファクトフルネス) 10 の思い込みを乗り越え、データを基に世界を正しく見る習慣**の著者です。

また、R の `gapminder` パッケージには、ファクトフルネスにも登場するデータで、Gapminder サイトでも使っているデータの一部を、使いやすい、練習用のデータとして提供しているものです。

- Gapminder: <https://www.gapminder.org>
 - Test on Top: You are probably wrong about - upgrade your worldview
 - Bubble Chart: [https://www.gapminder.org/tools/#\\$chart-type=bubbles&url=v1](https://www.gapminder.org/tools/#$chart-type=bubbles&url=v1)

- Dallar Street: [https://www.gapminder.org/tools/#\\$chart-type=bubbles&url=v1](https://www.gapminder.org/tools/#$chart-type=bubbles&url=v1)
- Data: <https://www.gapminder.org/data/>
- R Package gapminder by Jennifer Bryan
 - Package site: <https://CRAN.R-project.org/package=gapminder>
 - Site: <https://github.com/jennybc/gapminder>
 - Documents: <https://www.rdocumentation.org/packages/gapminder/versions/0.3.0>
- Package Help `?gapminder` or `gapminder` in the search window of Help
 - The main data frame `gapminder` has 1704 rows and 6 variables:
 - * `country`: factor with 142 levels
 - * `continent`: factor with 5 levels
 - * `year`: ranges from 1952 to 2007 in increments of 5 years
 - * `lifeExp`: life expectancy at birth, in years
 - * `pop`: population
 - * `gdpPercap`: GDP per capita (US\$, inflation-adjusted)

```
library(tidyverse)
library(gapminder)
library(WDI)
```

18.6 まとめ

18.6.1 関数

18.6.1.1 base

- `colnames()`
- `range()`

18.6.1.2 tidyverse

- `select()`
- `filter()`
- `mutate()`
- `summarize()` (`summarise()`)
- `arrange()`
- `distinct()`

-
- `pull()`
 - `scale_x_log10()`
 - `group_by()`

第 19 章

視覚化 (Visualize)

19.1 基本的なこと

R では、追加パッケージを使わなくても、簡単に、グラフを描画できますが、質の高いグラフを作成するには、`ggplot2` パッケージを用いたものが標準となっています。`ggplot2` は、`tidyverse` パッケージの一部ですので、`tidyverse` パッケージをインストール、使えるように、`library(tidyverse)` として読み込んであれば、そのまま使うことができます。

サイト：<https://ggplot2.tidyverse.org> パッケージサイト：<https://CRAN.R-project.org/package=ggplot2>

19.1.1 ggplot2 概要

`ggplot2` is a system for declaratively creating graphics, based on *The Grammar of Graphics*. You provide the data, tell `ggplot2` how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

`ggplot2` は、グラフィックスの生成に関する「Grammar of Graphics (グラフィックスの文法)」に基づいて、一つ一つの要素を定めていくことによって、グラフを作成するシステムです。データを提供し、変数を視覚的要素にマッピングする方法や、どのようなグラフィカルな基本要素を使用するかを `ggplot2` に伝えると、詳細な部分は `ggplot2` が処理してくれます。

19.1.2 基本的な例

19.1.2.1 tidyverse の読み込み

タイトルや、列名などに日本語を使う場合があるときは、`install.packages('showtext')` で、`showtext` パッケージをインストールして、下のように設定してください。そうでない場合は、最初の行 `library(tidyverse)` だけで他は不要です。

```

library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats    1.0.0      v stringr   1.5.0
#> v ggplot2    3.4.3      v tibble    3.2.1
#> v lubridate  1.9.2      v tidyr     1.3.0
#> v purrr      1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all con
library(showtext)
#> Loading required package: sysfonts
#> Loading required package: showtextdb
showtext_auto()

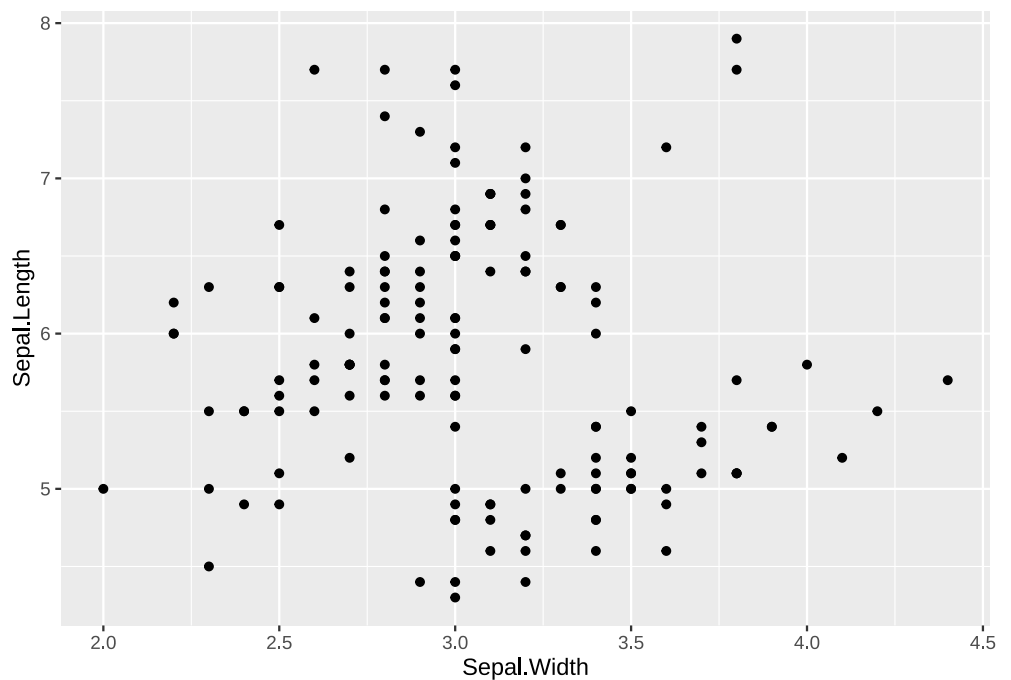
```

19.1.2.2 復習

Tidyverse の章で、紹介した例の復習から始めましょう。

```
df_iris <- datasets::iris
```

```
df_iris |> ggplot(aes(Sepal.Width, Sepal.Length)) + geom_point()
```



さまざまな描画が可能ですが、一番、一般的な、散布図、plot に対応するものです。ggplot の中の、aes (aesthetic) の部分に、x 軸、y 軸に対応する変数 (列名) を書きます。


```
<DATA> |> ggplot(aes(<変数 x の列名>, <変数 y の列名>)) + geom_point()
```

もっと明示的に

```
<DATA> |> ggplot(aes(x = <変数 x の列名>, y = <変数 y の列名>)) +
geom_point()
```

さらには

```
<DATA> |> ggplot(mapping = aes(x = <変数 x の列名>, y = <変数 y の列名>)) +
geom_point()
```

パイプを使わず

```
ggplot(<DATA>, aes(x = <変数 x の列名>, y = <変数 y の列名>)) +
geom_point()
```

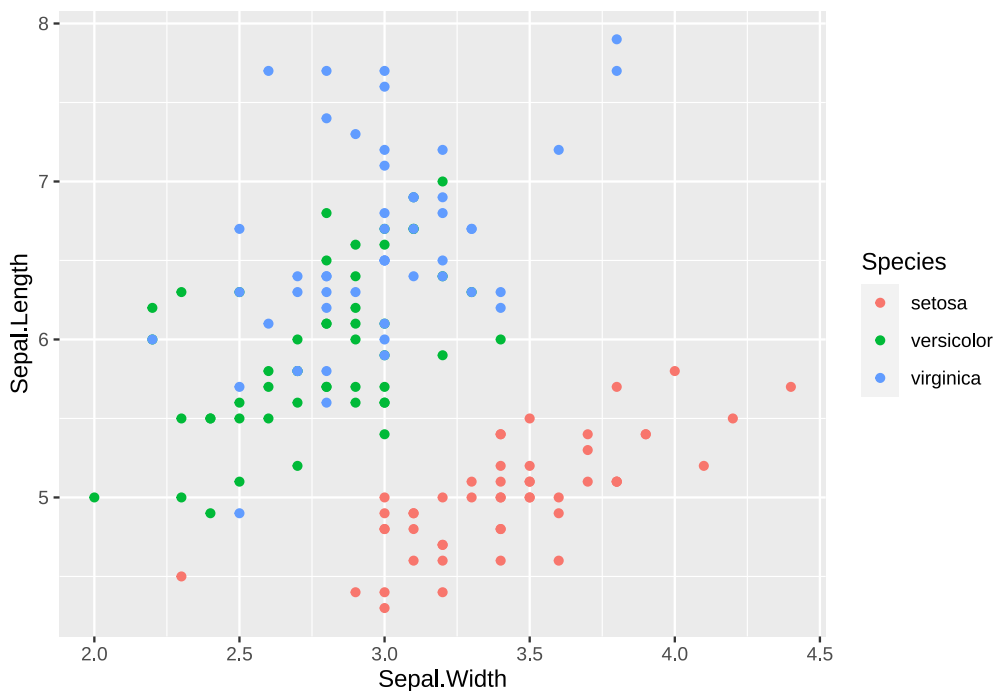
や、さらに、詳しく

```
ggplot(data = <DATA>, mapping = aes(x = <変数 x の列名>, y = <変数 y の列名>)) +
geom_point()
```

も可能です。

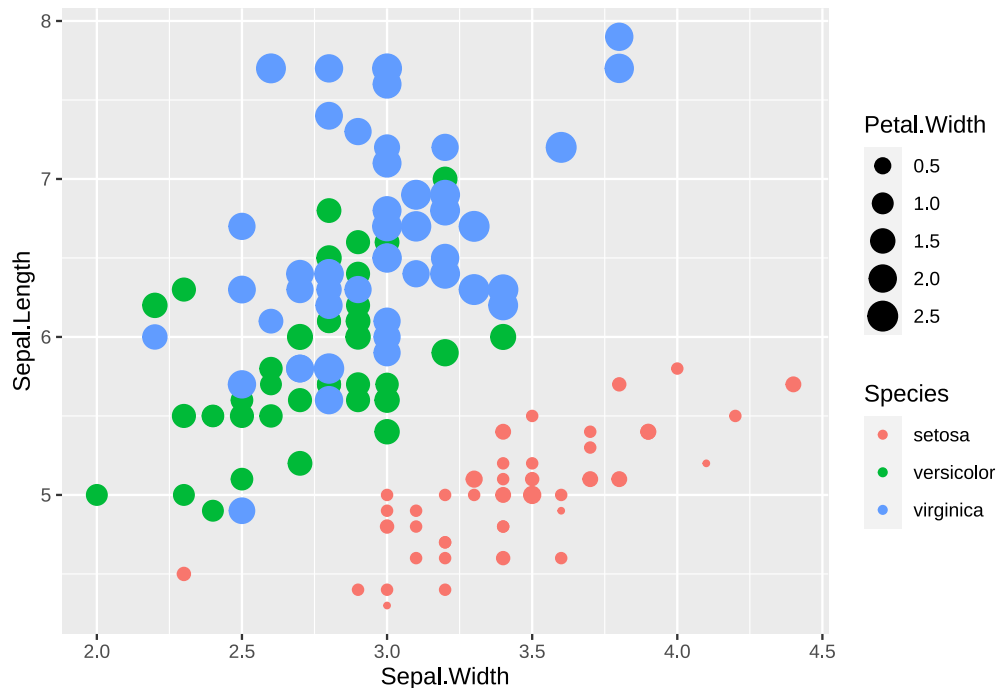
種類 (Species) ごとに色を変える場合には、`color = Species` とします。

```
df_iris |> ggplot(aes(Sepal.Width, Sepal.Length, color = Species)) +
geom_point()
```



さらに、点の大きさを、`Petal.Width` によって変える場合には次のように、`size = Petal.Width` を加えます。

```
df_iris |>
  ggplot(aes(Sepal.Width, Sepal.Length, color = Species,
             size = Petal.Width)) +
  geom_point()
```



ここでは、散布図でしたから、`geom_point()` を使いましたが、これを他のものに変えていくと、さまざまなグラフが描けるようになっています。

19.2 散布図 (Scatter Plot)

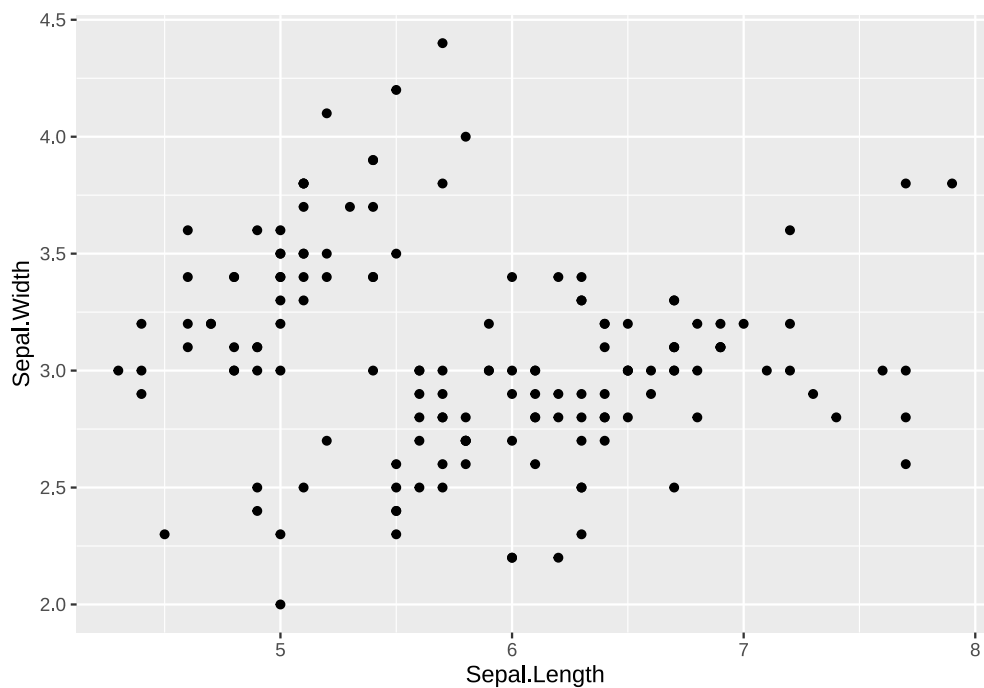
散布図は、データの二つの変数 (列) を x と y に対応させる、最も基本的なグラフです。最初に試すべきグラフだともいえます。mapping = は省略することができます。

```
ggplot(data = <data>, aes(x = <column name for x>, y = <column name for y>)) +
  geom_point()
```

```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point()
```

変形 (Transform) のときにつかった、iris データを使います。

```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point()
```

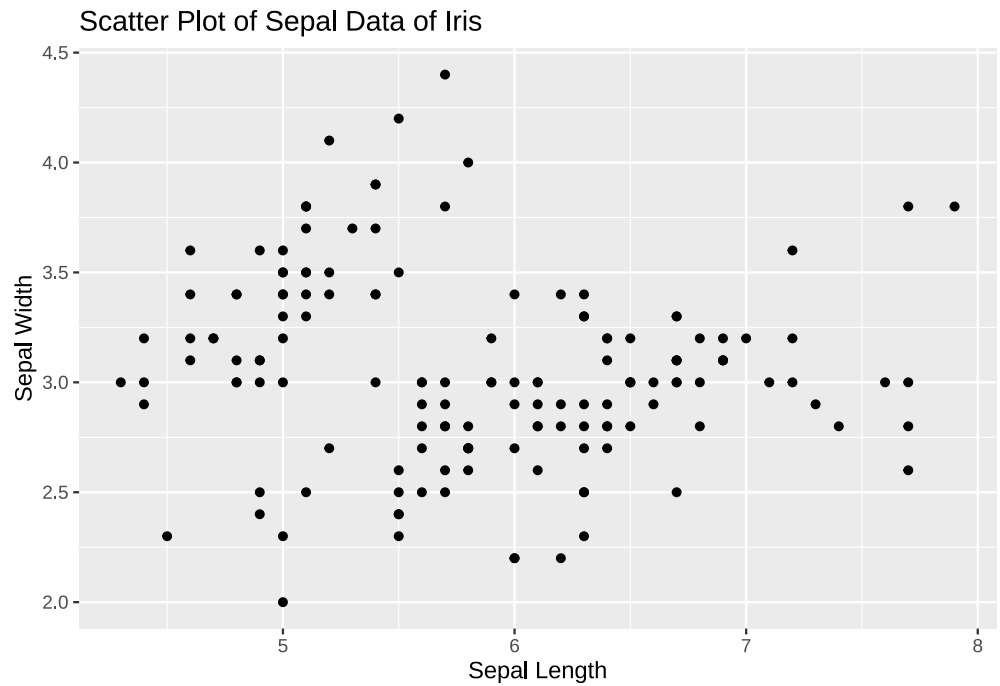


19.2.1 ラベル Labels

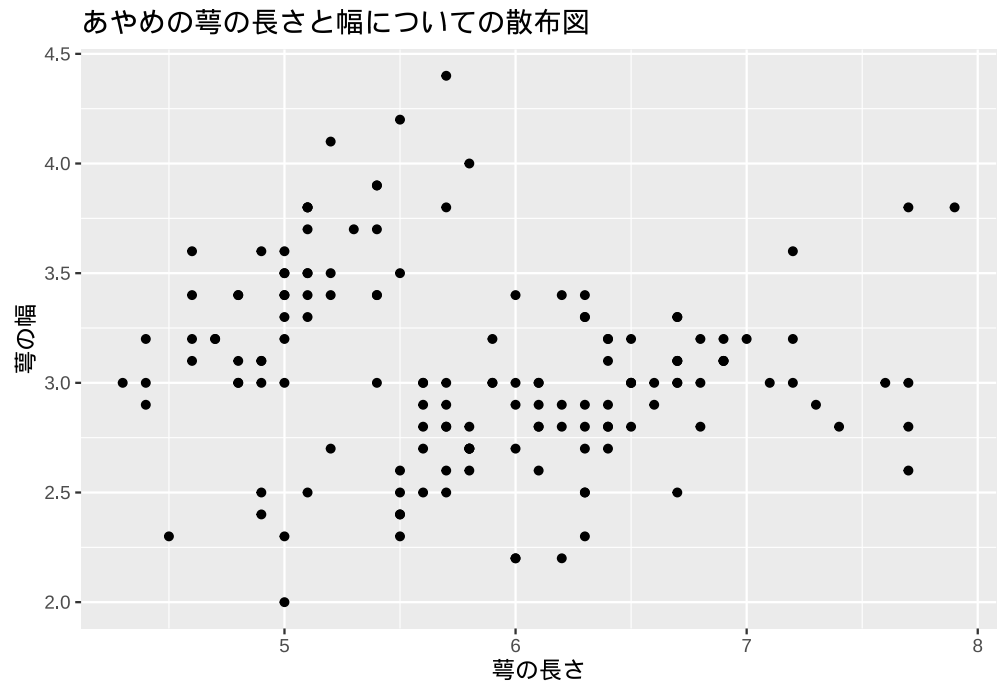
グラフの表題や、x 軸、y 軸のラベルをつけるには `labs()` を使います。

```
ggplot(data = <data>, aes(x = <column name for x>, y = <column name for y>)) +  
  geom_point() +  
  labs(title = "Title", x = "Label for x", y = "Label for y")
```

```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point() +  
  labs(title = "Scatter Plot of Sepal Data of Iris", x = "Sepal Length", y = "Sepal Width")
```



```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point() +  
  labs(title = "あやめの萼の長さ と幅 についての 散布図", x = "萼の長さ", y = "萼の幅")
```

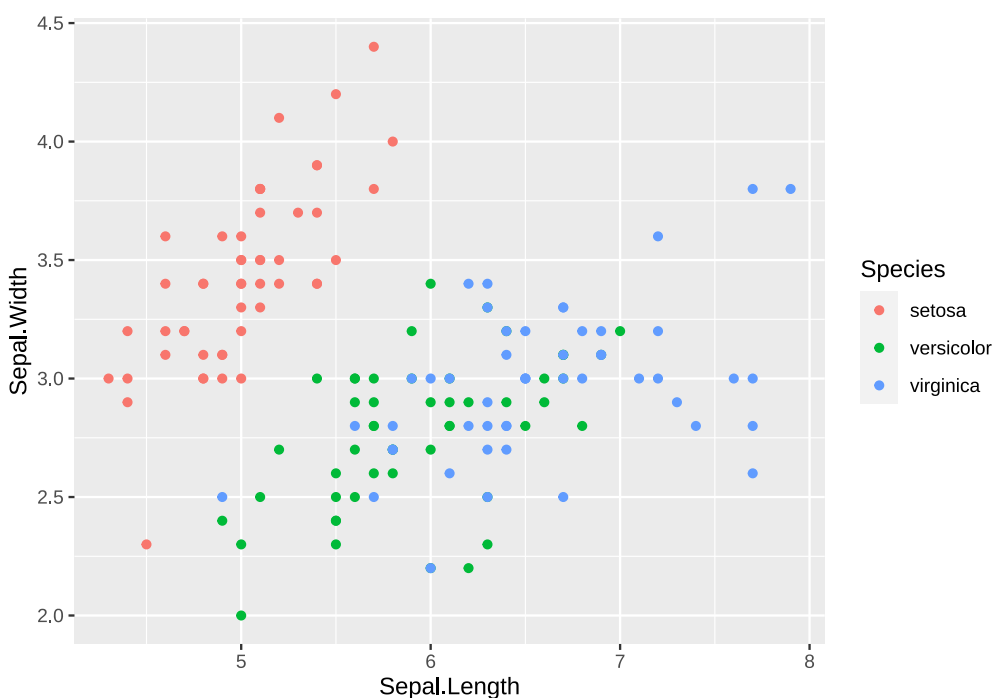


このように、日本語をタイトルや、ラベルに使うことも可能ですが、以後は、データに日本語が含まれていない場合には、そのまま表示します。

19.2.2 色付き Colors

菖蒲 (iris) のデータは、Species 列に、三種類の菖蒲の名前が含まれていました。それぞれに、違う色で表示してみましょう。それには、x 軸、y 軸に対応する変数を指定したように、`color = Species` と指定します。

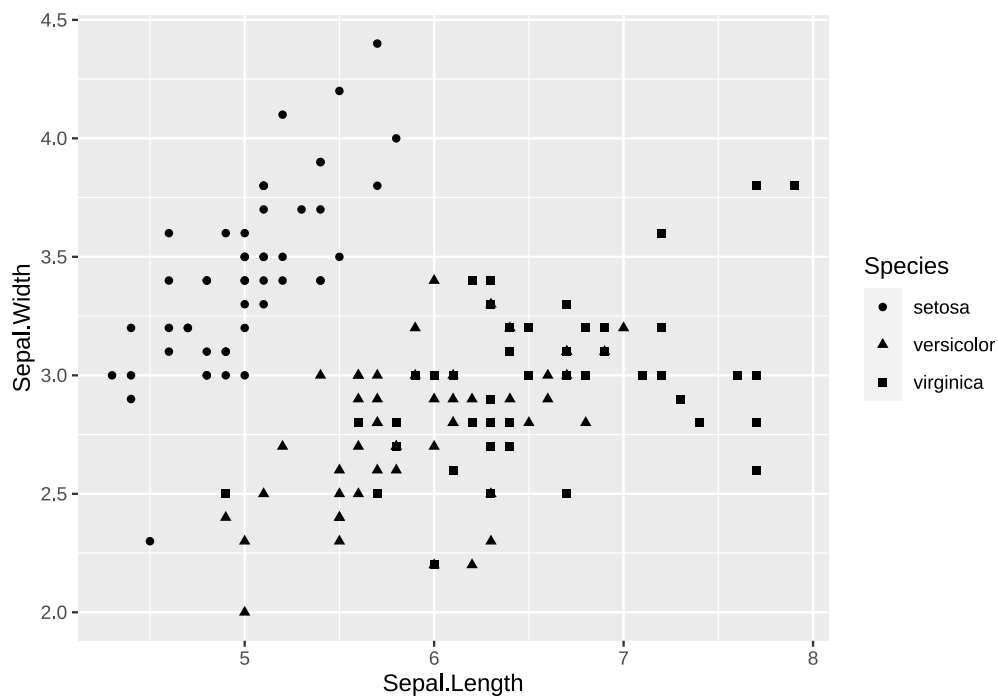
```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  
  geom_point()
```



19.2.3 形状 Shapes

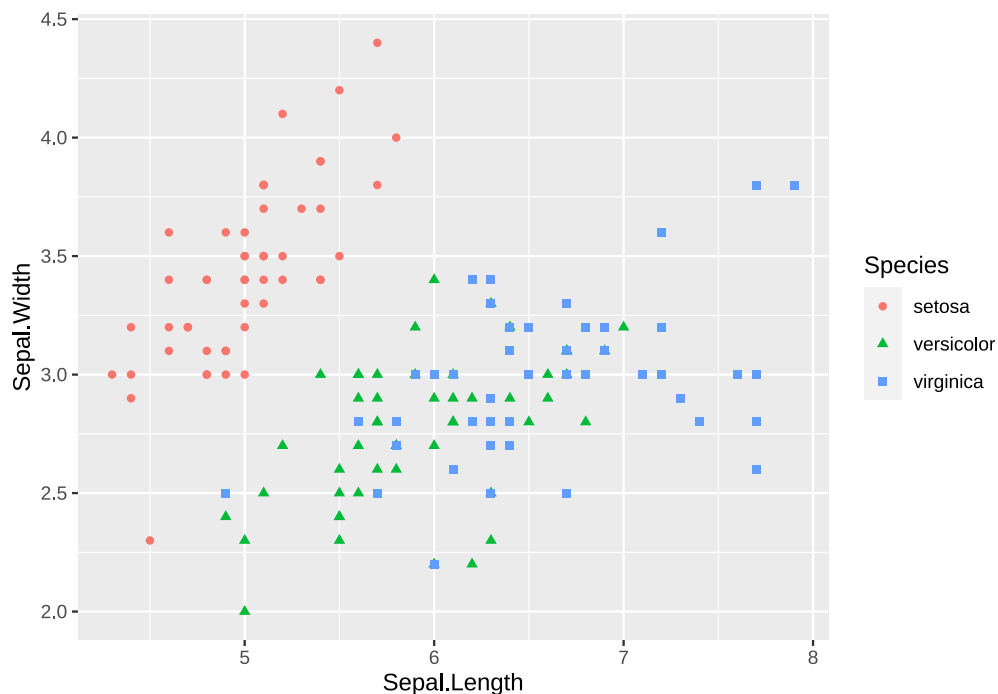
色ではなく、形で Species を区別することも可能です。

```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width, shape = Species)) +  
  geom_point()
```



色と、形、両方を同時に使うことも可能です。

```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species, shape = Species)) +
  geom_point()
```



19.3 箱ひげ図 Boxplot

The boxplot compactly displays the distribution of a continuous variable.

箱ひげ図は、連続な値をとる変数の分布を簡潔な表示でみることができるグラフです。箱や、線の長さ、外れ値の表示なども、正確に決まっていますので、次のビデオをみてください。英語ですが、わかりやすく、まとまっていると思います。

<https://vimeo.com/222358034>

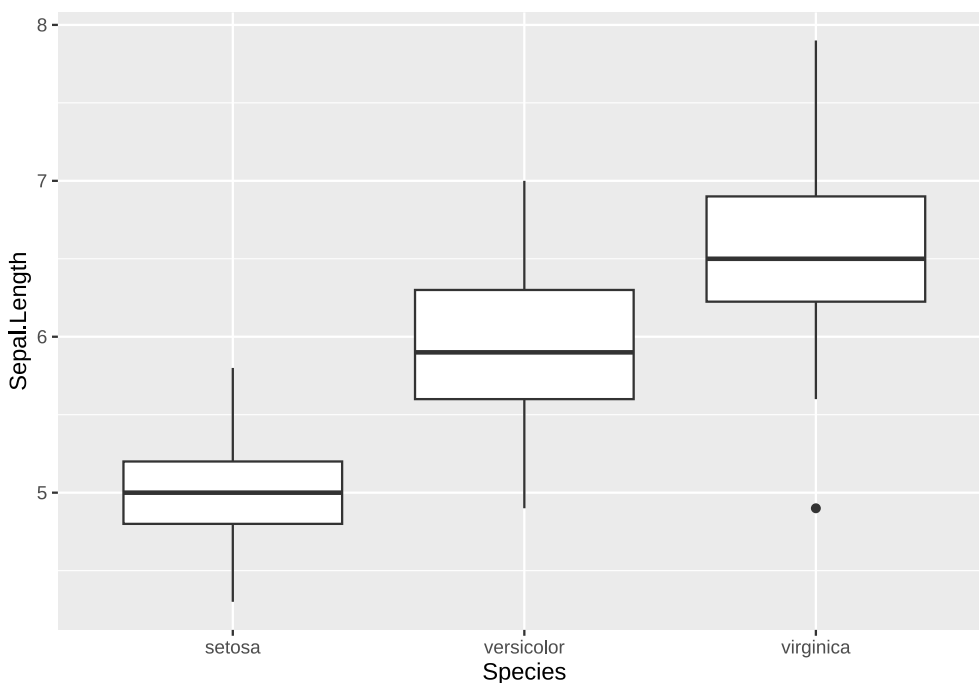
日本語では、下のサイトの箱ひげ図の項目を見ていただければと思います。外れ値についての説明もあります。

箱ひげ図と幹葉表示：<https://bellcurve.jp/statistics/course/5219.html>

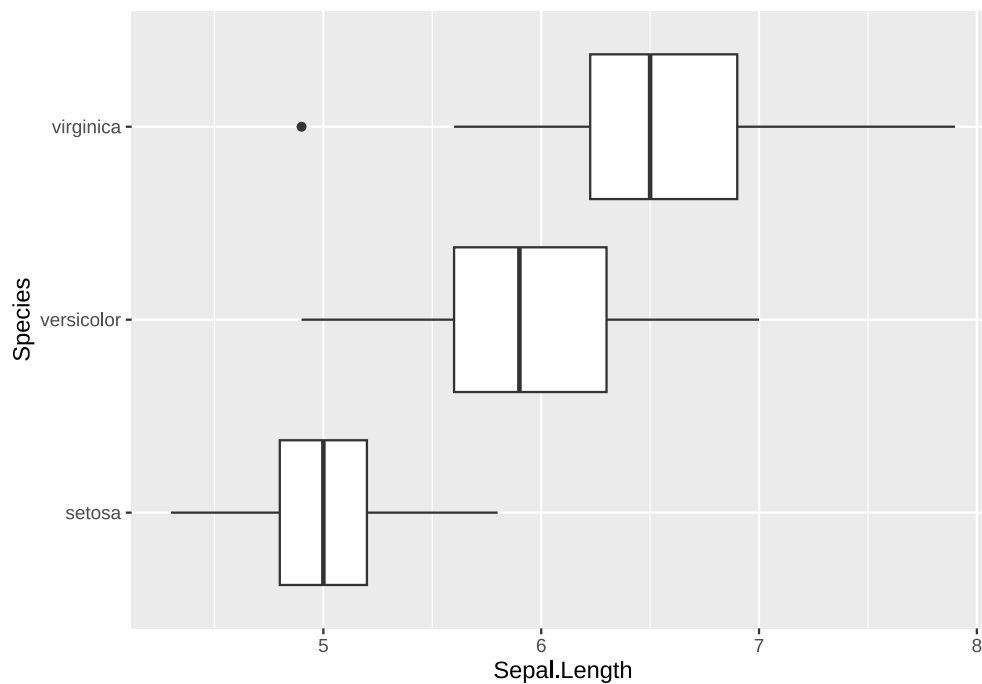
Transcript ボタンから、スクリプトを表示することもできます。

例のように、それぞれのグループごとに箱ひげ図を表示することもできますが、その場合は、文字データや、離散的な数値データ（いくつかの飛び飛びの値をとる変数）を使います。x と y を入れ替えれば、横向きになります。

```
ggplot(data = df_iris, aes(x = Species, y = Sepal.Length)) +  
  geom_boxplot()
```



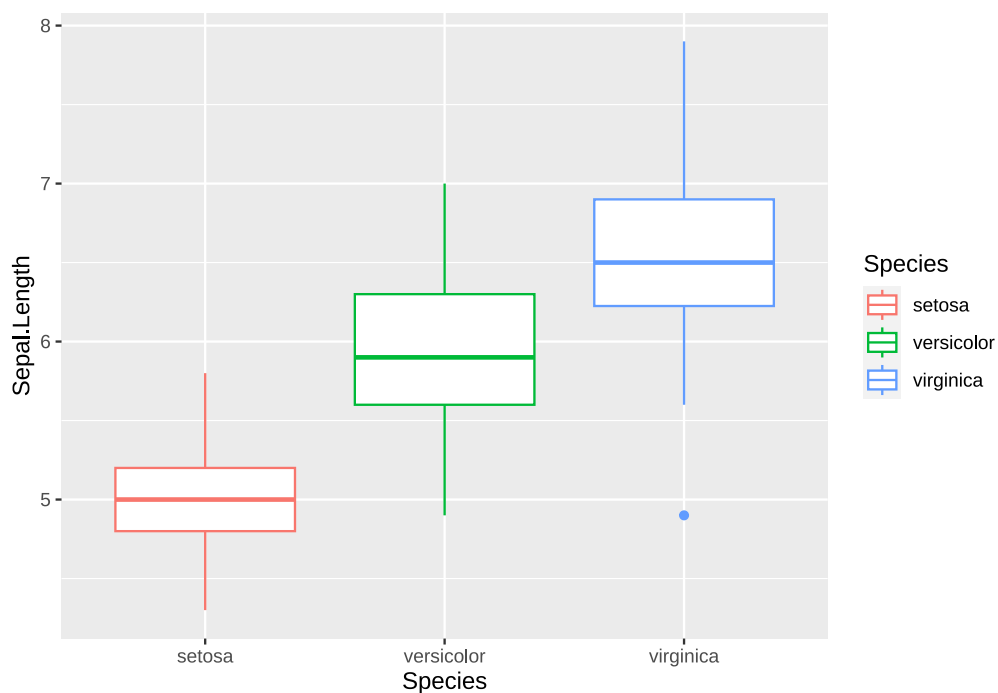
```
ggplot(data = df_iris, aes(y = Species, x = Sepal.Length)) +  
  geom_boxplot()
```



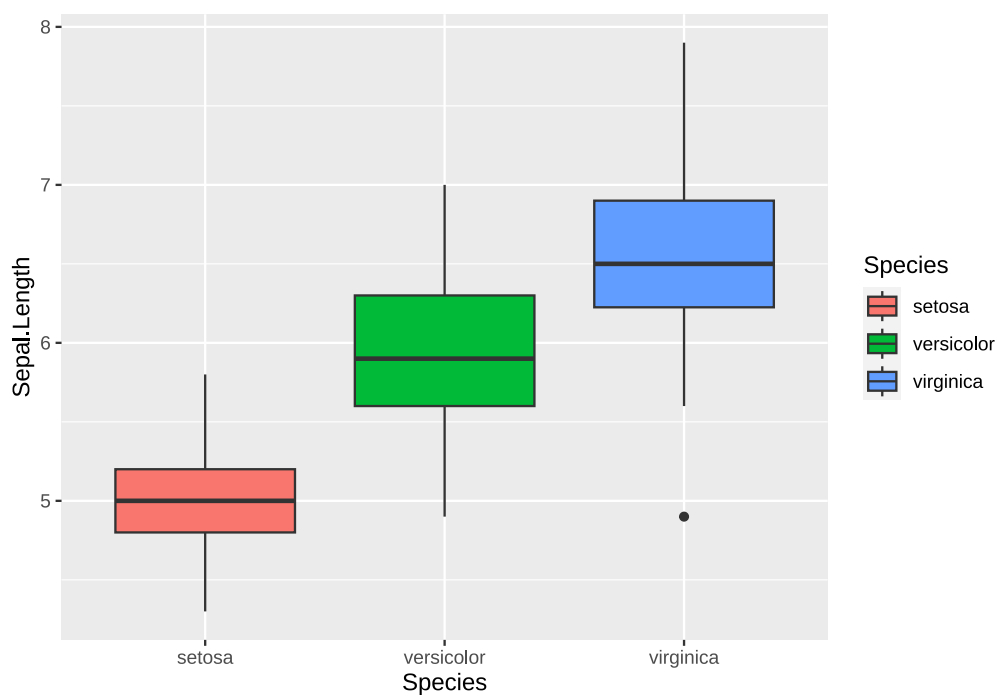
各、種類 (Species) ごとに、Sepal.Width (萼 (がく) 幅) が、どのように分布しているかを示しています。真ん中の太い線が、中央値 (median)、箱が、第一四分位 (Q1) から第三四分位 (Q3)、線と点で表される外れ値も、どのような基準か定められています。(IQR = Q3-Q1, 線は、 $Q3+1.5 \times IQR$ 以下に入っている実際の値までと、 $Q1-1.5 \times IQR$ 以上に入っている実際の値まで。それらに入っていないものが外れ値)。

color を指定すると、枠に色がつき、fill を指定すると、箱の中が塗り潰されます。

```
ggplot(data = df_iris, aes(x = Species, y = Sepal.Length, color = Species)) +  
  geom_boxplot()
```

```
ggplot(data = df_iris, aes(x = Species, y = Sepal.Length, fill = Species)) +  
  geom_boxplot()
```



19.3.1 ヒストグラム Histogram

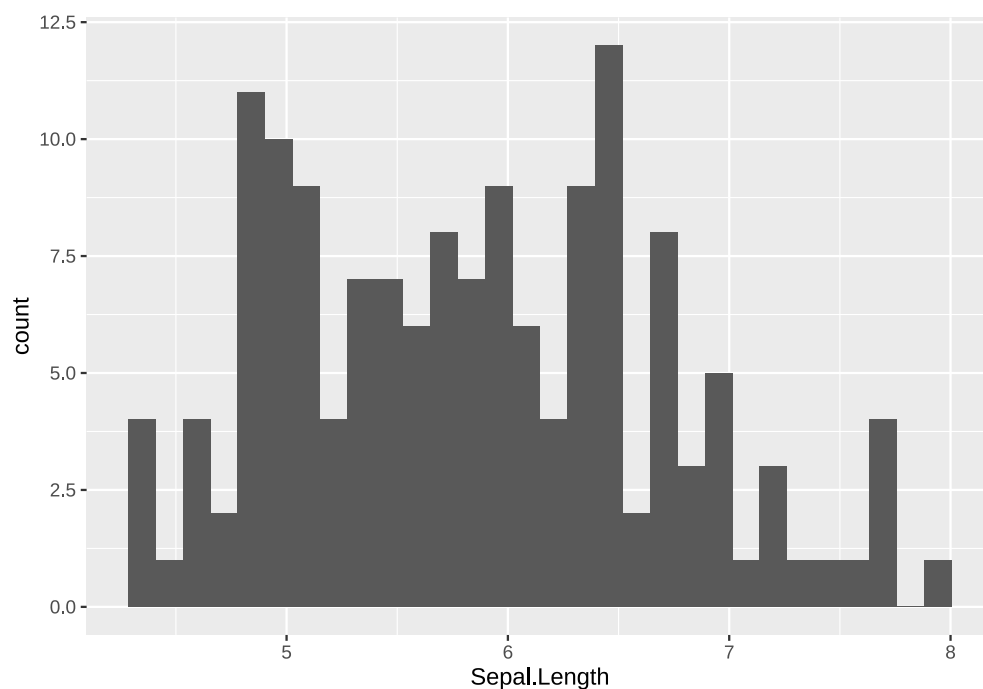
Visualize the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (`geom_histogram()`) display the counts with bars; frequency polygons

(`geom_freqpoly()`) display the counts with lines. Frequency polygons are more suitable when you want to compare the distribution across the levels of a categorical variable.

単一の連続変数の分布を可視化するために、x 軸をビンに分割し、各ビン内の観測値の数を数えます。ヒストグラム (`geom_histogram()`) は、棒で数を表示します。一方、頻度多角形 (`geom_freqpoly()`) は、数を線で表示します。頻度多角形は、カテゴリ変数のレベル間の分布を比較したい場合により適しています。

説明ビデオです。 <https://vimeo.com/221607341>

```
ggplot(data = df_iris, aes(x = Sepal.Length)) +
  geom_histogram()
#> `stat_bin()` using `bins = 30`. Pick better value with
#> `binwidth`.
```

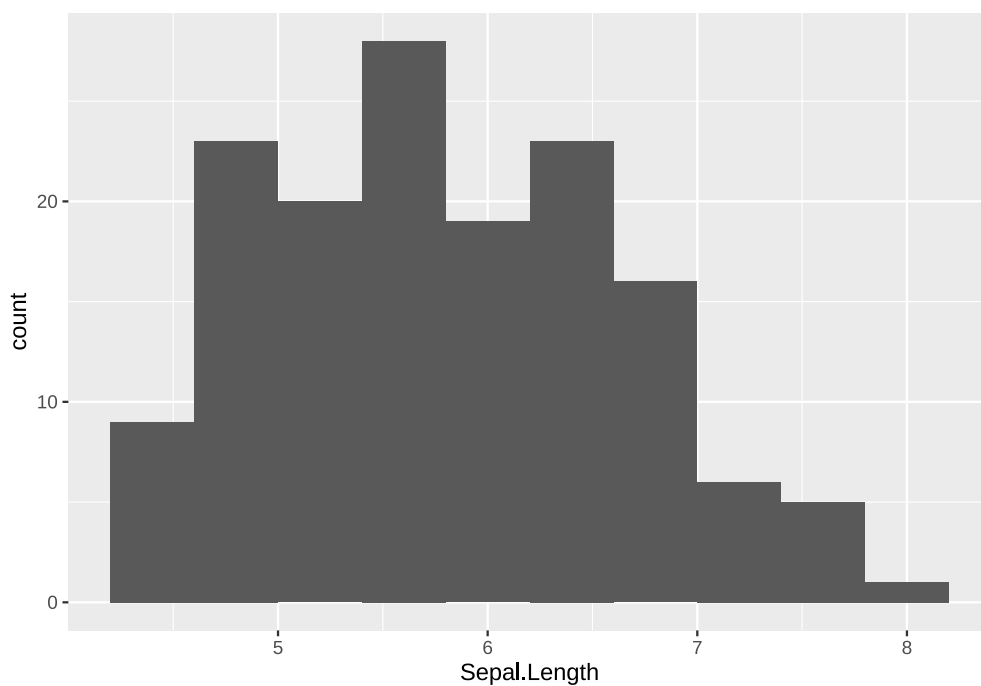


枠 (bins) を幾つに分けるか、または枠の幅を指定するようにこのメッセージが表示されます。

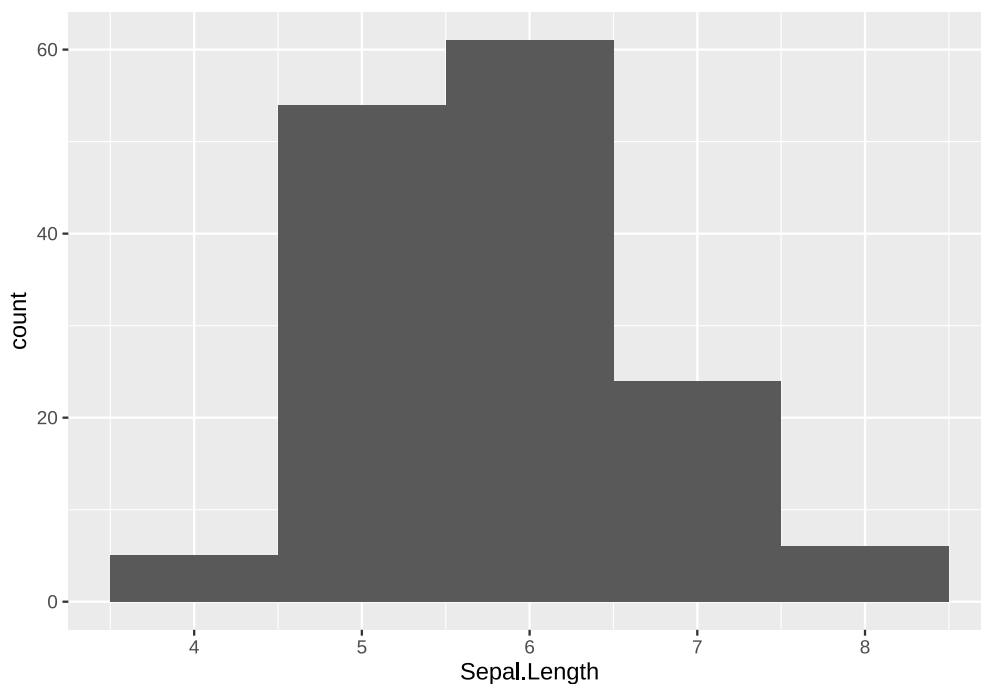
枠 (bins) の数を変更するには `bins = <number>` を使います。幅を指定するときは、`binwidth = <number>` とします。

最初の例では、枠の個数を (初期設定では 30 になっているものを) 10 個とし、二つ目の例では、幅を 1 にしています。

```
ggplot(data = df_iris, aes(x = Sepal.Length)) +
  geom_histogram(bins = 10)
```



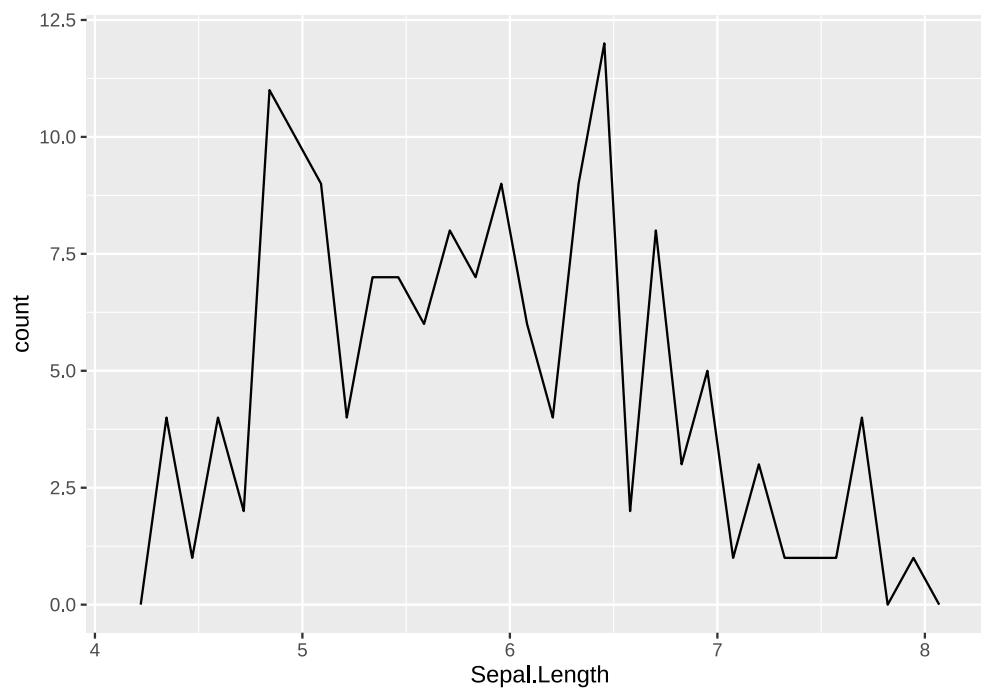
```
ggplot(data = df_iris, aes(x = Sepal.Length)) +  
  geom_histogram(binwidth = 1)
```



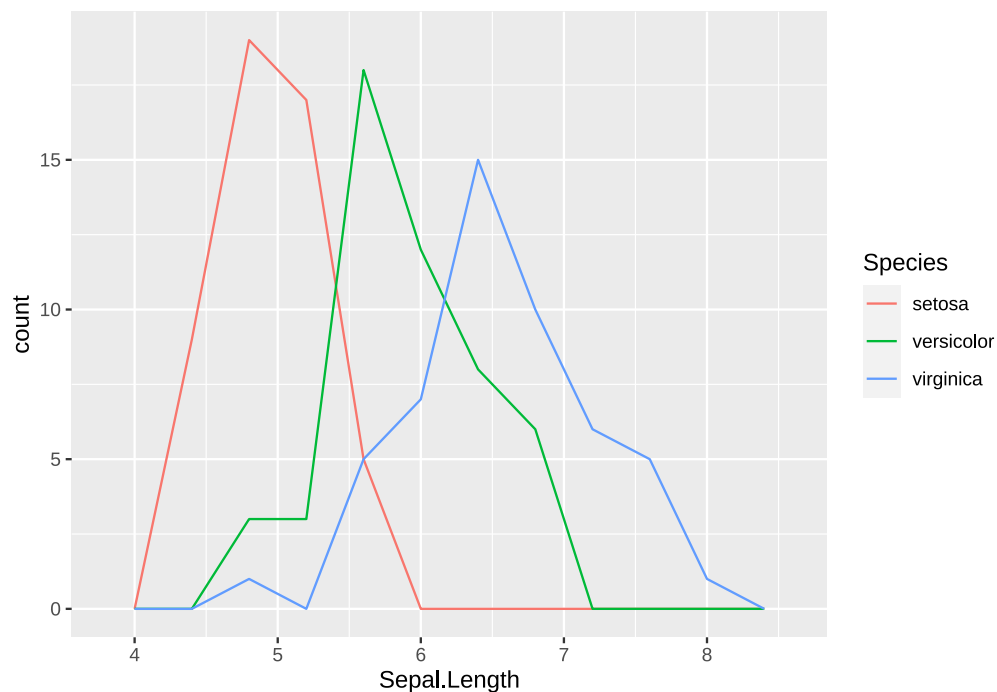
頻度多角形 (`geom_freqpoly()`) を使うと以下ようになります。Species ごとに比べたり、色をつけたりもできます。

```
ggplot(data = df_iris, aes(x = Sepal.Length)) +  
  geom_freqpoly()  
#> `stat_bin()` using `bins = 30`. Pick better value with
```

```
#> `binwidth`.
```

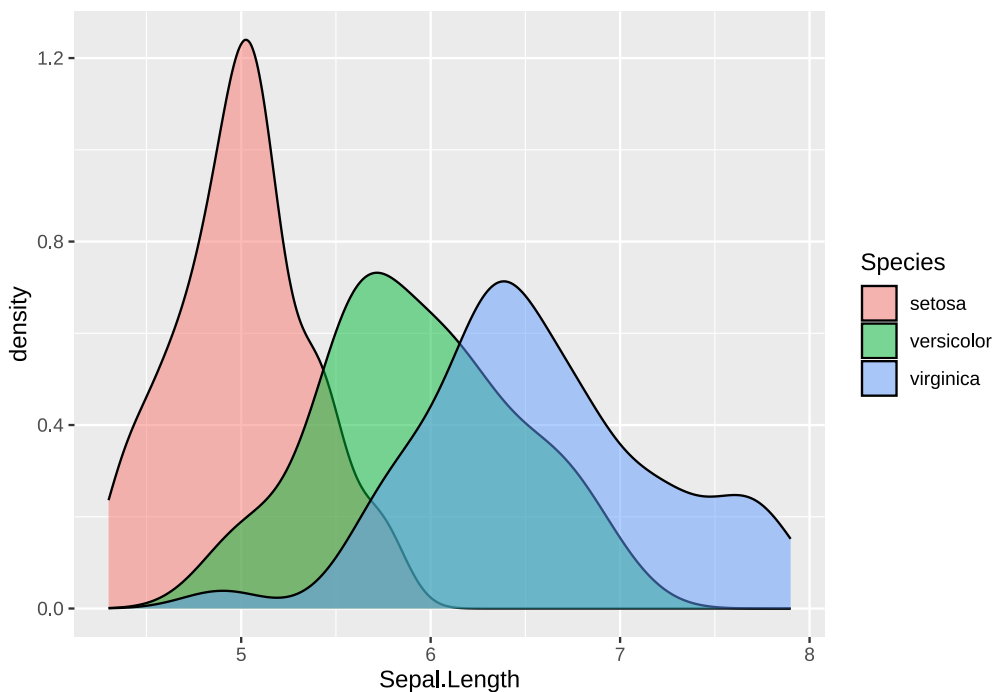


```
ggplot(data = df_iris, aes(x = Sepal.Length, color = Species)) +  
  geom_freqpoly(bins = 10)
```



滑らかな曲線にするときは、density plot を使います。alpha は透明度で 0 から 1 の値で指定します。数が小さい方が薄くなります。color で線の色もあわせて設定することも可能です。いろいろと試してみてください。

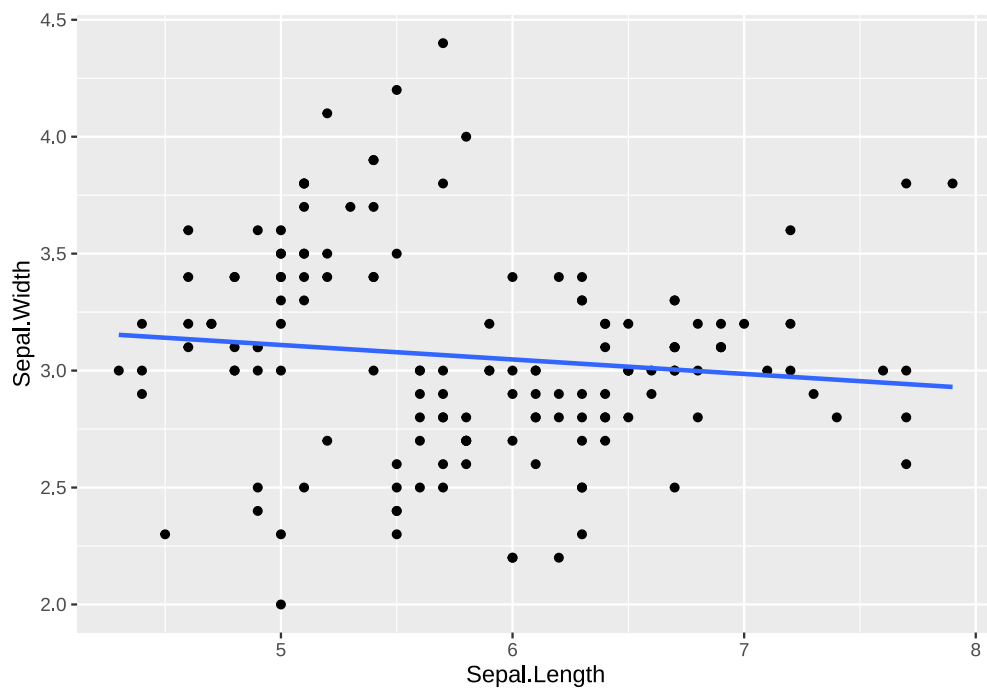
```
ggplot(data = df_iris, aes(x = Sepal.Length, fill = Species)) +  
  geom_density(alpha = 0.5)
```



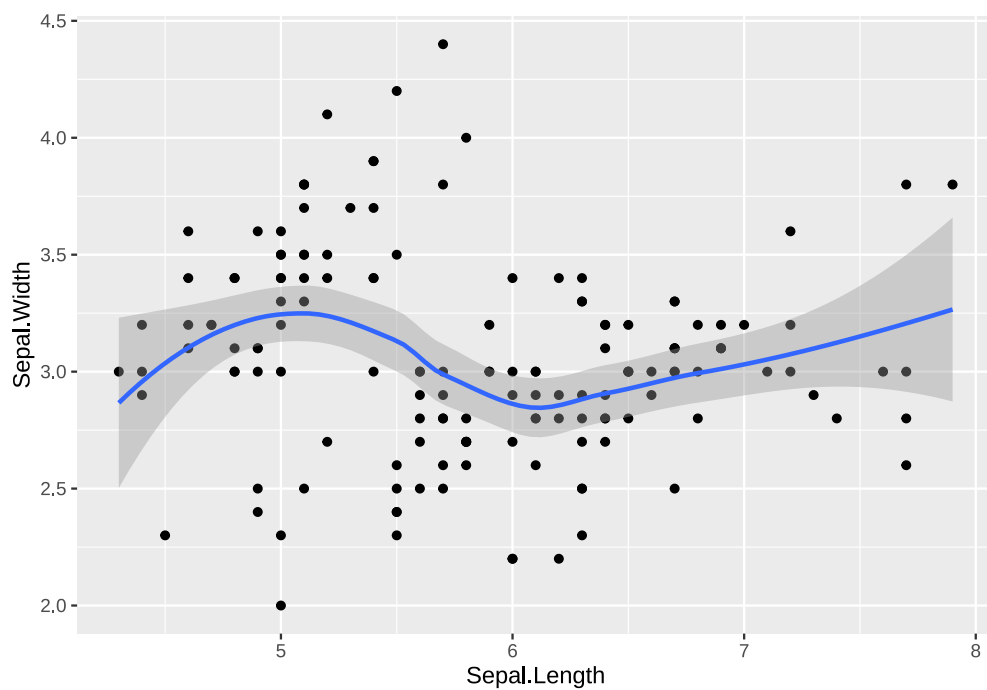
19.3.2 線形モデル Data Modeling

回帰直線を加えたり、他の近似曲線を加えることも可能です。グラフとしても直感的理解を助けますが、統計的な扱いについては、Modeling で説明します。

```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point() +  
  geom_smooth(method = "lm", se = FALSE)
```



```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width)) +  
  geom_point() +  
  geom_smooth()
```



19.4 例から学ぶ ggplot2, I

19.4.0.1 mpg を使って

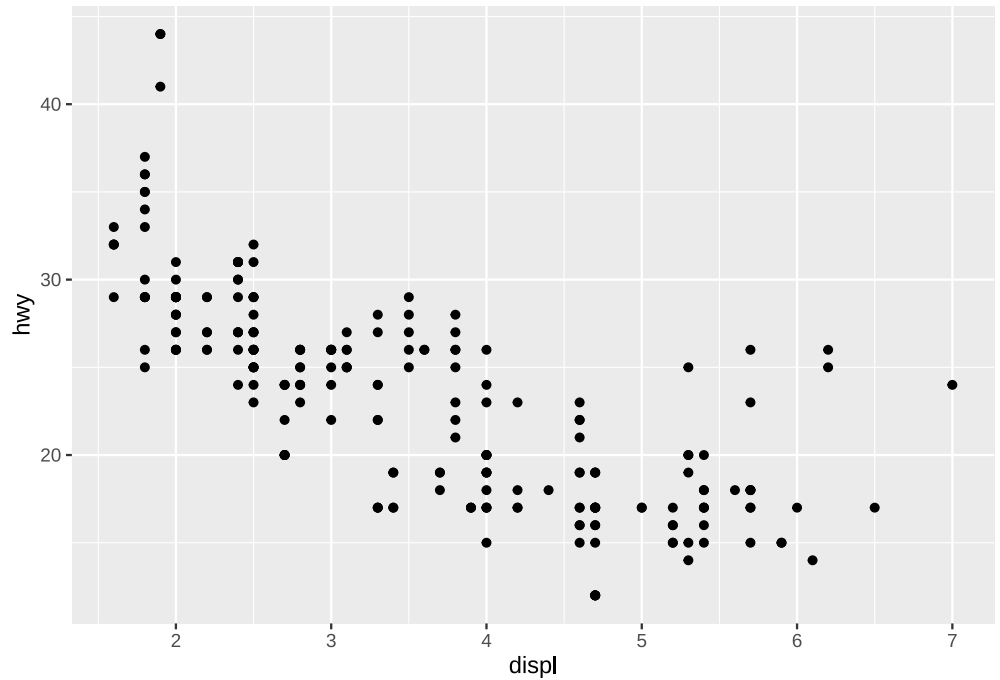
ggplot2 に含まれている、mpg (Fuel economy data from 1999 to 2008 for 38 popular models of cars - 1999 年から 2008 年の 38 の型式の車の年燃費) データを使って、簡単な、散布図と、箱ひげ図を描いてみます。mpg の変数などについては、Help で調べてください。

manufacturer は、メーカー名、model は、型式、displ は、排気量 (単位: リットル)、year は年式、cyl は、気筒数、trans は、オートマかマニュアルか、drv は、f が前輪駆動、r は後輪駆動、4 は四輪駆動、cty は街中での燃費 (1 ガロンで何マイル走るか)、hwy は高速道路での燃費 (1 ガロンで何マイル走るか)、fl 燃料の種類、class タイプ

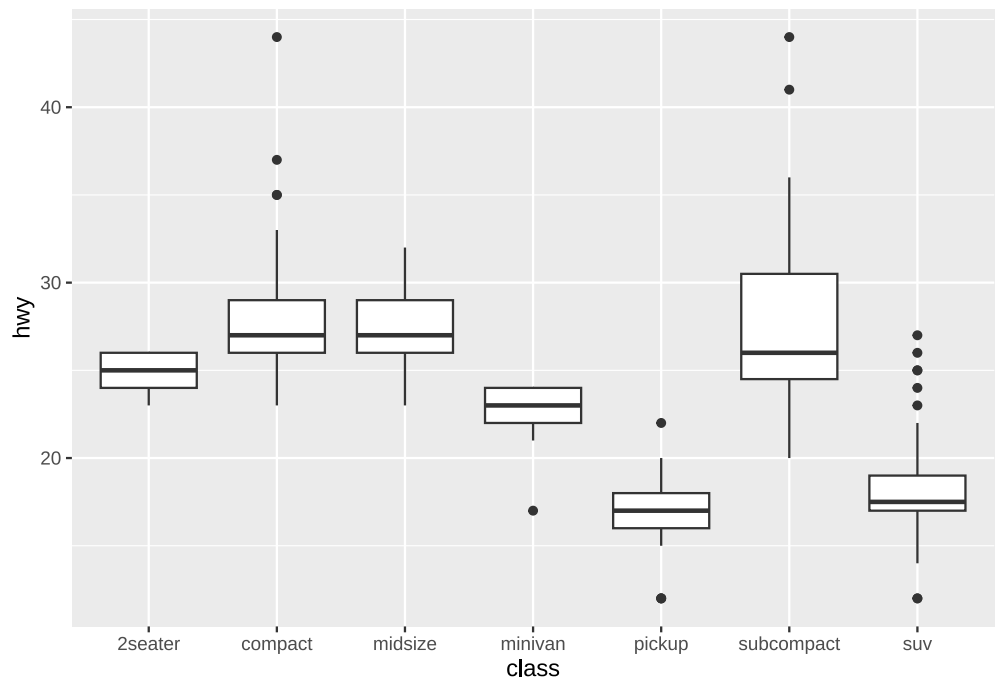
```
df_mpg <- ggplot2::mpg
df_mpg
#> # A tibble: 234 x 11
#>   manufacturer model   displ  year  cyl trans drv   cty
#>   <chr>         <chr>   <dbl> <int> <int> <chr> <chr> <int>
#> 1 audi         a4       1.8  1999   4 auto~ f    18
#> 2 audi         a4       1.8  1999   4 manu~ f    21
#> 3 audi         a4       2    2008   4 manu~ f    20
#> 4 audi         a4       2    2008   4 auto~ f    21
#> 5 audi         a4       2.8  1999   6 auto~ f    16
#> 6 audi         a4       2.8  1999   6 manu~ f    18
#> 7 audi         a4       3.1  2008   6 auto~ f    18
#> 8 audi         a4 quat~ 1.8  1999   4 manu~ 4    18
#> 9 audi         a4 quat~ 1.8  1999   4 auto~ 4    16
#> 10 audi        a4 quat~ 2    2008   4 manu~ 4    20
#> # i 224 more rows
#> # i 3 more variables: hwy <int>, fl <chr>, class <chr>
```

```
glimpse(df_mpg)
#> Rows: 234
#> Columns: 11
#> $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi~
#> $ model         <chr> "a4", "a4", "a4", "a4", "a4", "a4", "~
#> $ displ         <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.~
#> $ year          <int> 1999, 1999, 2008, 2008, 1999, 1999, 2~
#> $ cyl           <int> 4, 4, 4, 4, 6, 6, 6, 4, 4, 4, 4, 6, 6~
#> $ trans         <chr> "auto(15)", "manual(m5)", "manual(m6)~
#> $ drv           <chr> "f", "f", "f", "f", "f", "f", "f", "4~
#> $ cty           <int> 18, 21, 20, 21, 16, 18, 18, 18, 16, 2~
#> $ hwy           <int> 29, 29, 31, 30, 26, 26, 27, 26, 25, 2~
#> $ fl            <chr> "p", "p", "p", "p", "p", "p", "p", "p~
#> $ class         <chr> "compact", "compact", "compact", "com~
```

```
ggplot(data = df_mpg) + geom_point(mapping = aes(x = displ, y = hwy))
```



```
ggplot(data = df_mpg) + geom_boxplot(mapping = aes(x = class, y = hwy))
```



1. `data = df_mpg` でデータを指定します。
2. どのようなグラフにするか、幾何関数 (Geometric Function) を指定します。散布図では、`geom_point()`、箱ひげ図では、`geom_boxplot()` などです。
3. x 軸、y 軸などに対応する変数の写像 (mapping) を指定します。

- 散布図では、`displ` (displacement エンジンの排気量 (単位リットル)) を x 軸に、`hwy` 高速道路で 1 ガロンで走れる距離 (単位マイル) を y 軸に割り当てています。
- 箱ひげ図では、`class` 車の型式を、x 軸に、`hwy` 高速道路で 1 ガロンで走れる距離 (単位マイル) を y 軸に割り当てています。

記号的に書くと、下のようになっています。

```
ggplot(data = <DATA>) + <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

19.5 例から学ぶ ggplot2, II

19.5.1 df_wdi, df_wdi_extra

前の章の Tidyverse で読み込み、概観した、世界開発指標 (World Development Indicators) のデータを使います。参照: WDI のデータ

```
library(tidyverse)
library(WDI)
```

WDI の使い方は、世界銀行の部分で紹介しますが、はじめてのデータサイエンスの例でも紹介したように、データコードを利用して、データを読み込みます。ここでは、出生時の平均寿命と、一人当たりの GDP と、総人口のデータを使います。

- SP.DYN.LE00.IN: Life expectancy at birth, total (years) 出生時の平均寿命
- SP.POP.TOTL: Population, total 総人口
- NY.GDP.PCAP.KD: GDP per capita (constant 2015 US\$) 一人当たりの GDP

次のコードで読み込みます。

```
df_wdi <- WDI(
  country = "all",
  indicator = c(lifeExp = "SP.DYN.LE00.IN", pop = "SP.POP.TOTL",
                gdpPercap = "NY.GDP.PCAP.KD")
)
```

```
#> Rows: 16758 Columns: 7
#> -- Column specification -----
#> Delimiter: ","
#> chr (3): country, iso2c, iso3c
#> dbl (4): year, lifeExp, pop, gdpPercap
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
df_wdi_extra <- WDI(
  country = "all",
  indicator = c(lifeExp = "SP.DYN.LE00.IN", pop = "SP.POP.TOTL",
                gdpPercap = "NY.GDP.PCAP.KD"),
  extra = TRUE
)
```

すこし、追加情報を付加したのもも取得しておきます。

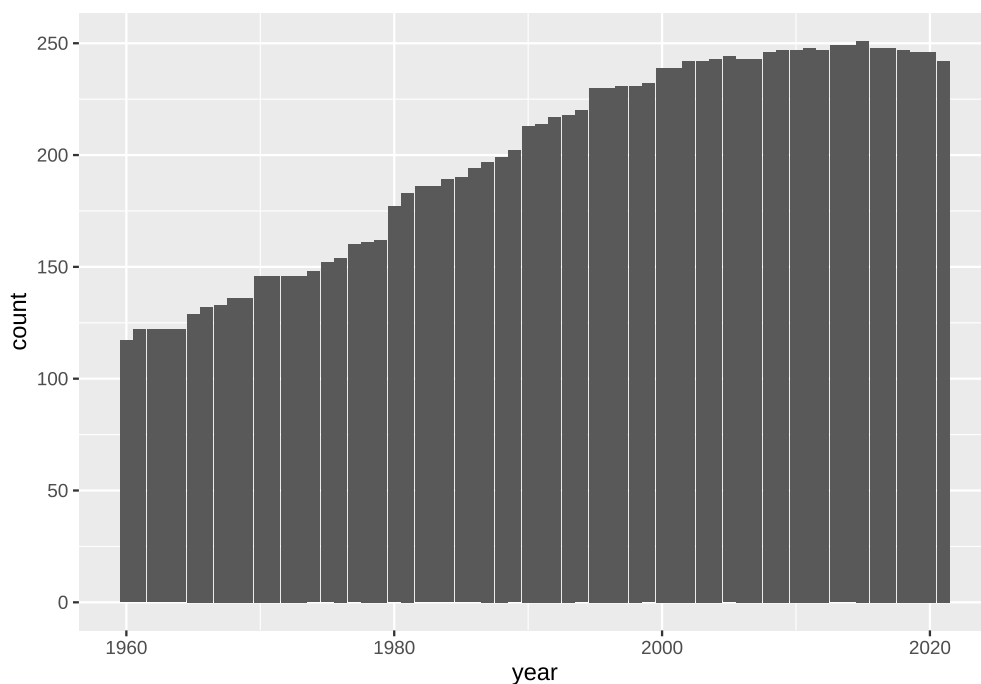
```
#> Rows: 16758 Columns: 15
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (6): year, lifeExp, pop, gdpPercap, longitude, lati...
#> lgl (1): status
#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa
```

```
df_wdi
#> # A tibble: 16,758 x 7
#>   country iso2c iso3c year lifeExp pop gdpPercap
#>   <chr>    <chr> <chr> <dbl> <dbl> <dbl> <dbl>
#> 1 Afghanistan AF AFG 1960 32.5 8622466 NA
#> 2 Afghanistan AF AFG 1961 33.1 8790140 NA
#> 3 Afghanistan AF AFG 1962 33.5 8969047 NA
#> 4 Afghanistan AF AFG 1963 34.0 9157465 NA
#> 5 Afghanistan AF AFG 1964 34.5 9355514 NA
#> 6 Afghanistan AF AFG 1965 35.0 9565147 NA
#> 7 Afghanistan AF AFG 1966 35.5 9783147 NA
#> 8 Afghanistan AF AFG 1967 35.9 10010030 NA
#> 9 Afghanistan AF AFG 1968 36.4 10247780 NA
#> 10 Afghanistan AF AFG 1969 36.9 10494489 NA
#> # i 16,748 more rows
```

19.5.2 棒グラフ (bar graph and column graph)

まずは、年ごと、指標 (indicator) ごとに、どの程度データがあるか見てみましょう。

```
df_wdi |> drop_na(lifeExp, pop, gdpPercap) |>
  ggplot() + geom_bar(aes(year))
```



最初に、三つの indicator がどれも、NA でない行だけを選択していますから、三つの指標がすべて値がある、データを年ごとに数えてグラフにしたものが、上のものになっています。表にすると次のようになっています。

```
df_wdi |> drop_na(lifeExp, pop, gdpPercap) |>
  group_by(year) |> summarize(n = n())
#> # A tibble: 62 x 2
#>   year      n
#>   <dbl> <int>
#> 1  1960    117
#> 2  1961    122
#> 3  1962    122
#> 4  1963    122
#> 5  1964    122
#> 6  1965    129
#> 7  1966    132
#> 8  1967    133
#> 9  1968    136
#> 10 1969    136
#> # i 52 more rows
```

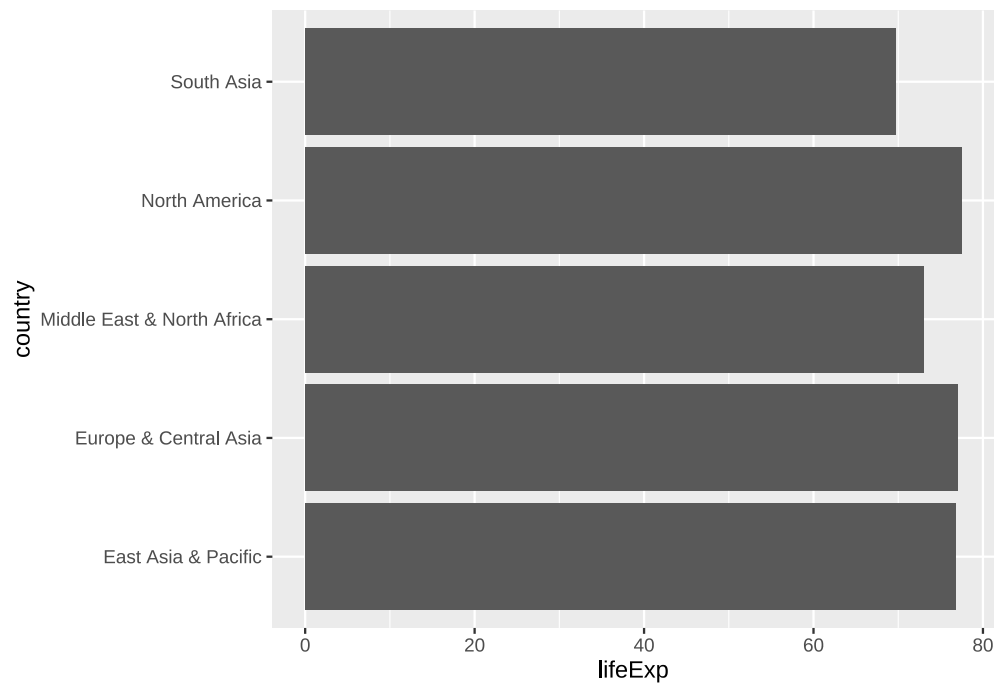
同じような棒グラフですが、それぞれの値をグラフにする場合には、`geom_col()` を使います。

```
regions <- df_wdi_extra |> drop_na(region) |>
  filter(region != "Aggregates") |> distinct(region) |> pull()
```

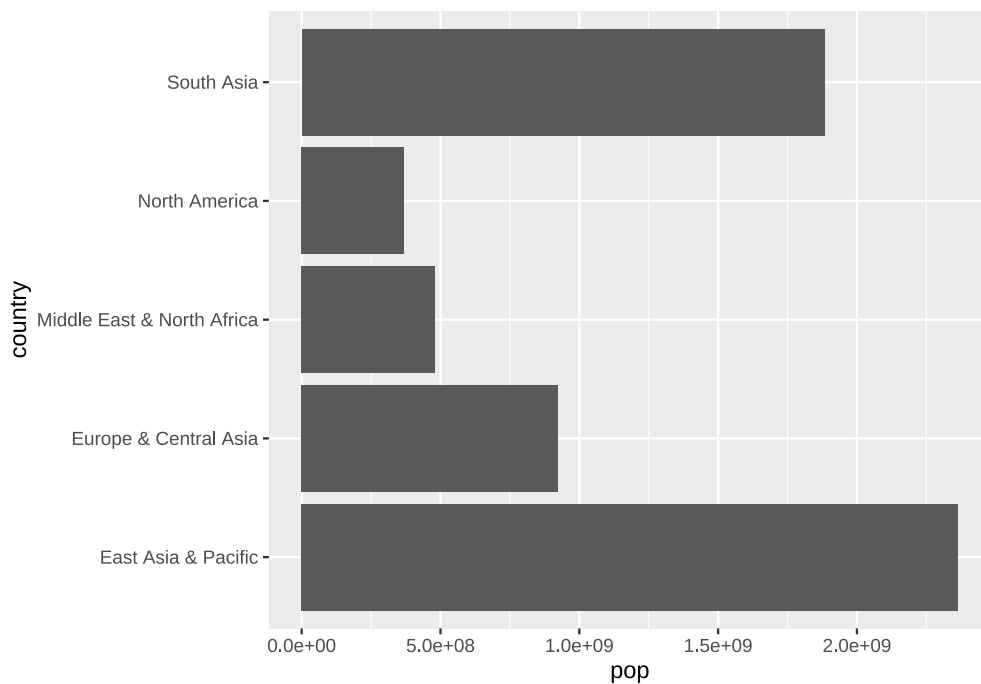
```
regions
```

```
#> [1] "South Asia"  
#> [2] "Europe & Central Asia"  
#> [3] "Middle East & North Africa"  
#> [4] "East Asia & Pacific"  
#> [5] "Sub-Saharan Africa"  
#> [6] "Latin America & Caribbean"  
#> [7] "North America"
```

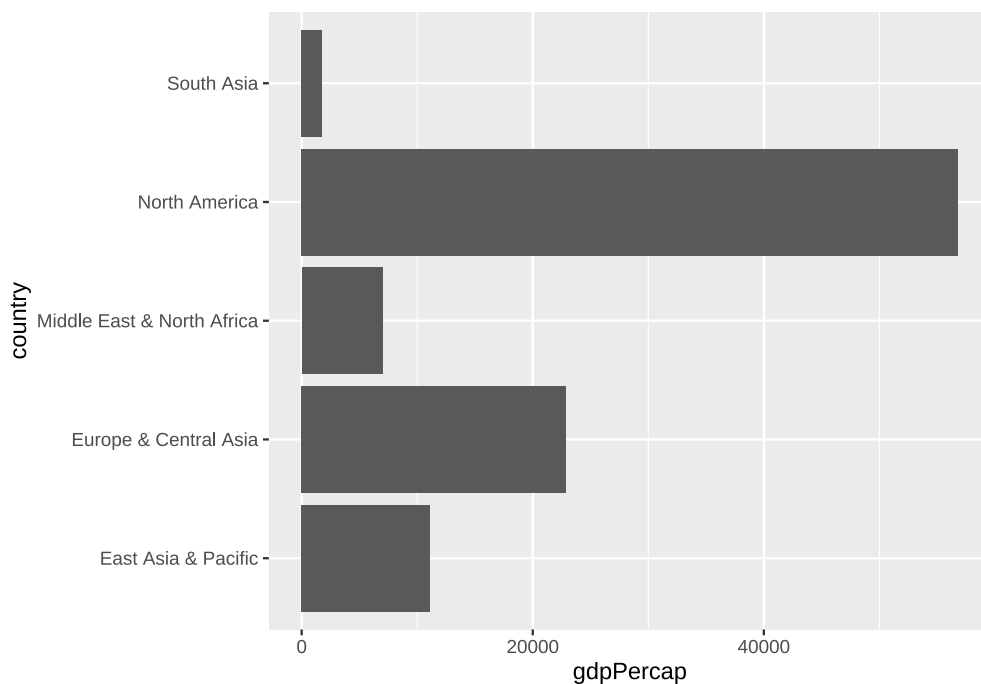
```
df_wdi_extra |> filter(income == "Aggregates", year == 2020) |>  
  filter(country %in% regions) |>  
  ggplot(aes(y = country, x = lifeExp)) + geom_col()
```



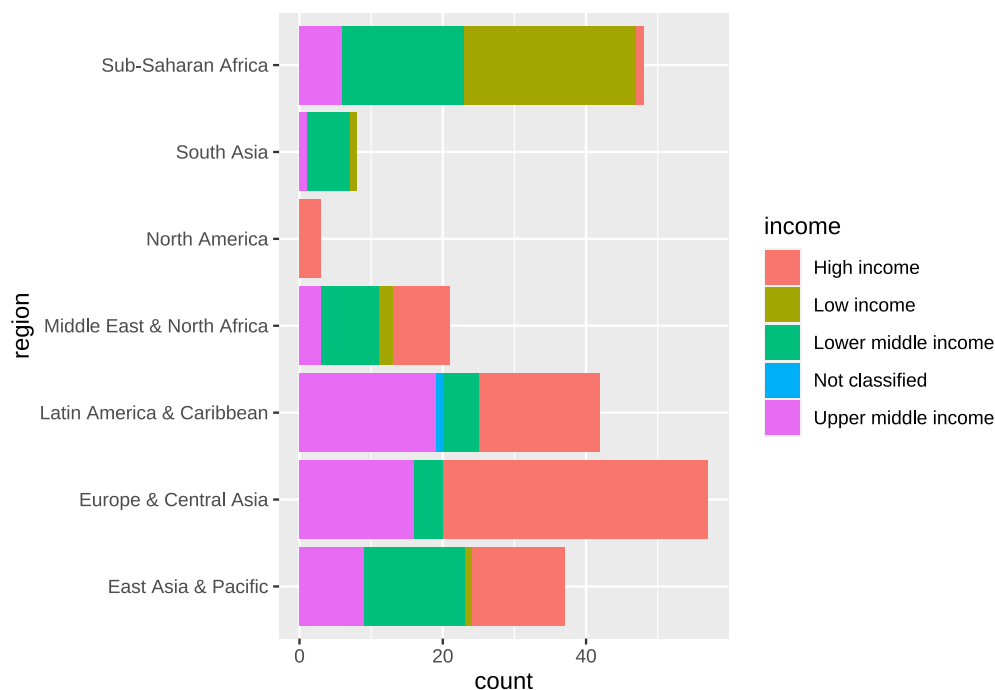
```
df_wdi_extra |> filter(income == "Aggregates", year == 2020) |>  
  filter(country %in% regions) |>  
  ggplot(aes(y = country, x = pop)) + geom_col()
```



```
df_wdi_extra |> filter(income == "Aggregates", year == 2020) |>
  filter(country %in% regions) |>
  ggplot(aes(y = country, x = gdpPercap)) + geom_col()
```



```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(y = region, fill = income)) + geom_bar()
```



19.5.3 備考

最後のグラフを例にとって、いくつかのコメントをしておきます。

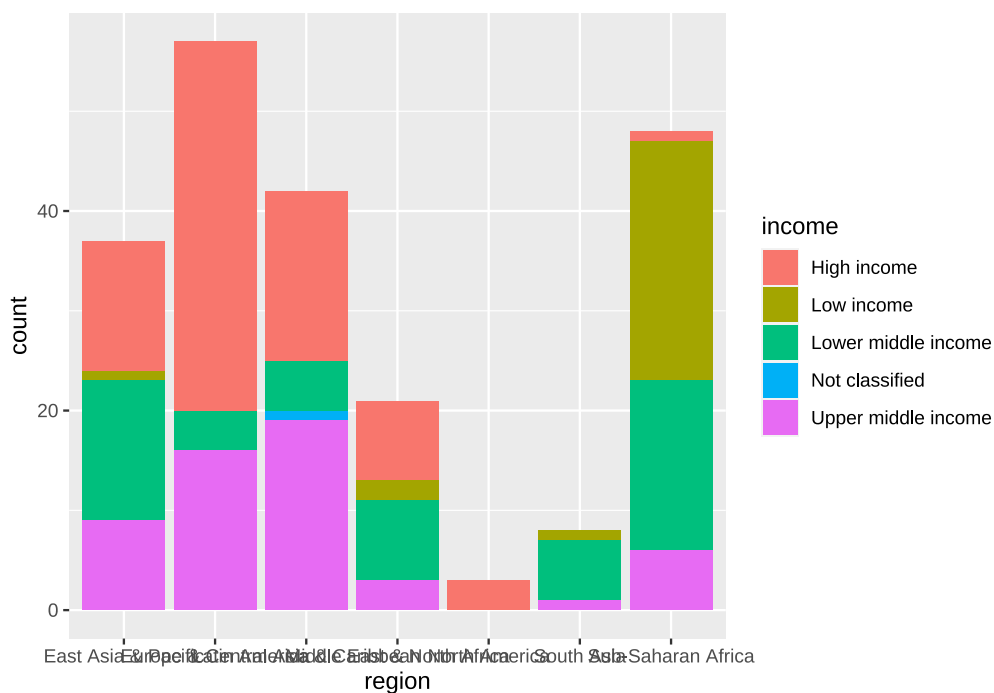
19.5.3.1 概要

地域の一覧を `regions` に入れてありますから、2020 年のデータを取り出し、その中で、それぞれの国ごとのデータではなく、集計してあるもの (Aggregates) を選択してあります。そして、地域ごとのグループ化します。Y 軸に、地域名を取って、`geom_bar()` ですから、データがいくつあるかで棒グラフにしています。そのときに、`fill = income` としていますから、収入レベルごとに、色を変えて塗りつぶし (`fill`) て、描いています。`color` は、枠に色をつけ、`fill` は、塗りつぶしです。

19.5.3.2 縦か横か

棒が横に伸びていますが、縦にすることも可能です。それには、`y = region` と書いてあるものを、`x = region` とすればできます。

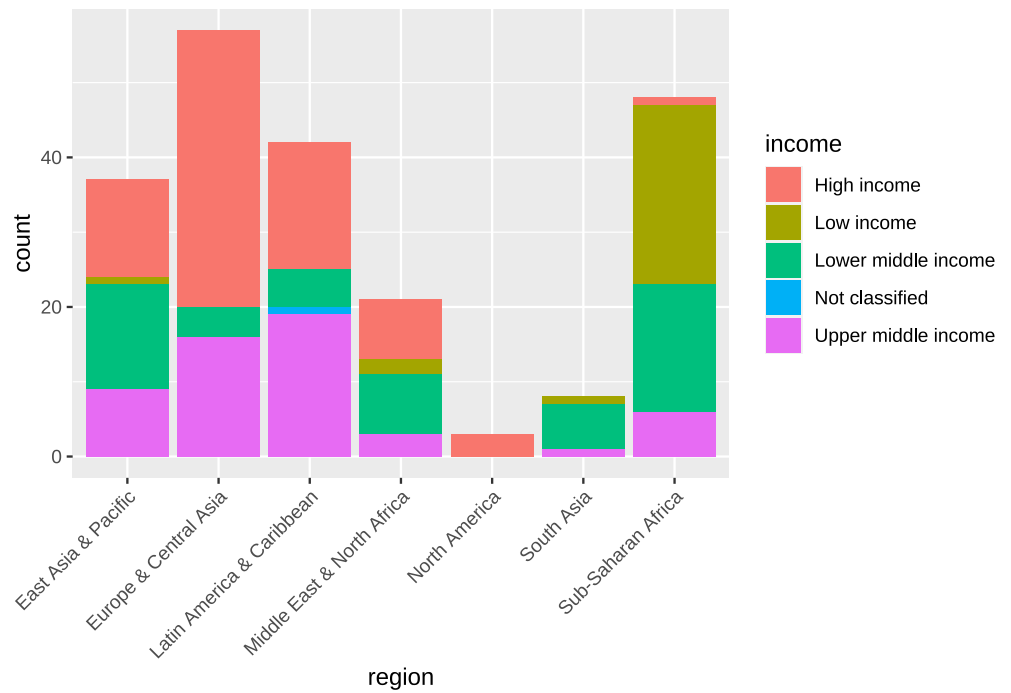
```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar()
```



19.5.3.3 地域名

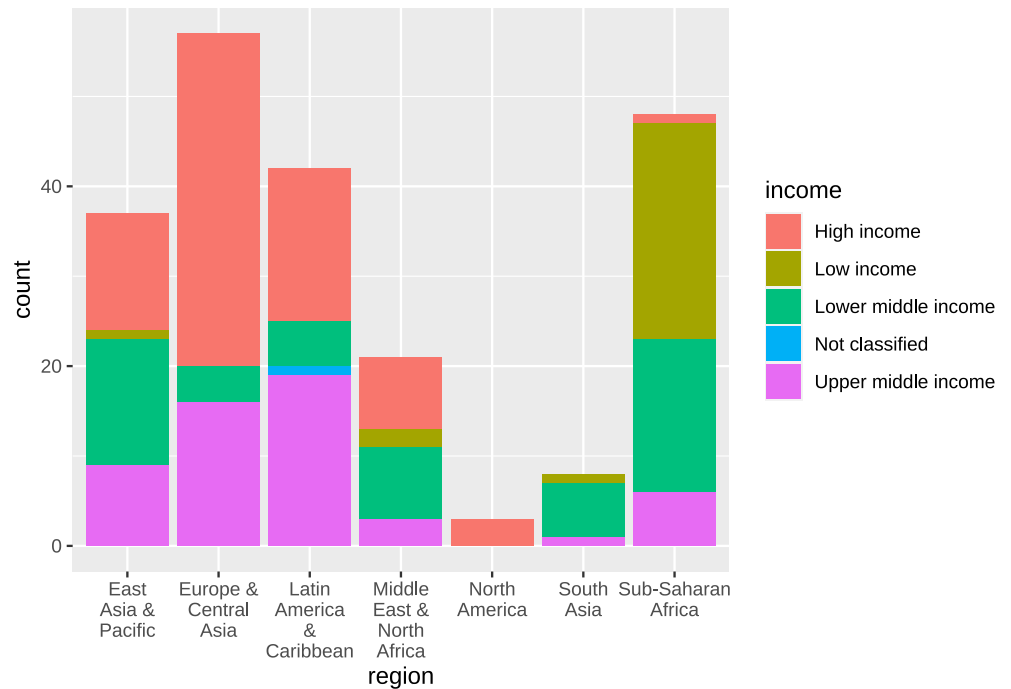
なぜ、横にしたか、見当がついたかと思います。縦にすると、地域名が長くて重なってしまうのですね。もちろん、それを、改善する手がいくつかあります。1つ目は、地域名を縦にしたり、角度を付ける方法。theme を使い、中に角度を書きます。実は、それだけだと、ちょっと重なってしまうので、hjust の変数で調節しています。垂直方向の調節は、vjust です。数はいろいろと変更してみてください。

```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar() +
  theme(axis.text.x = element_text(angle = 45, hjust=1))
```



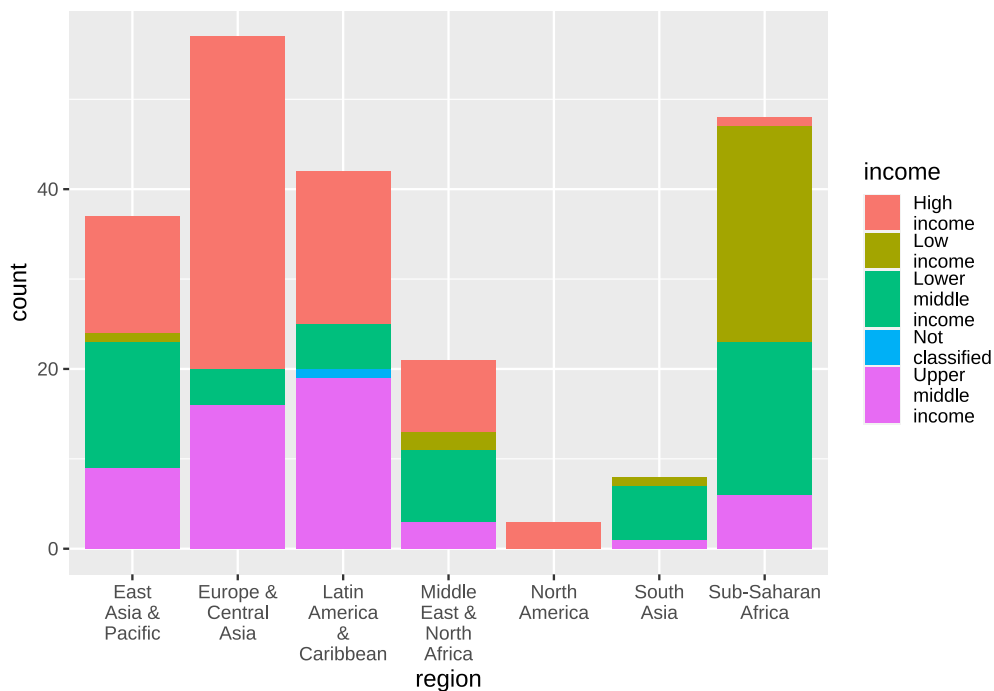
もう一つは折り返す方法です。こちらの方が、良いかもしれません。

```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar() +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8))
```



もし、fill の方の凡例 (legend) も短くしたければ、次のようにします。width のあとの数字は、適切な数を選んでください。

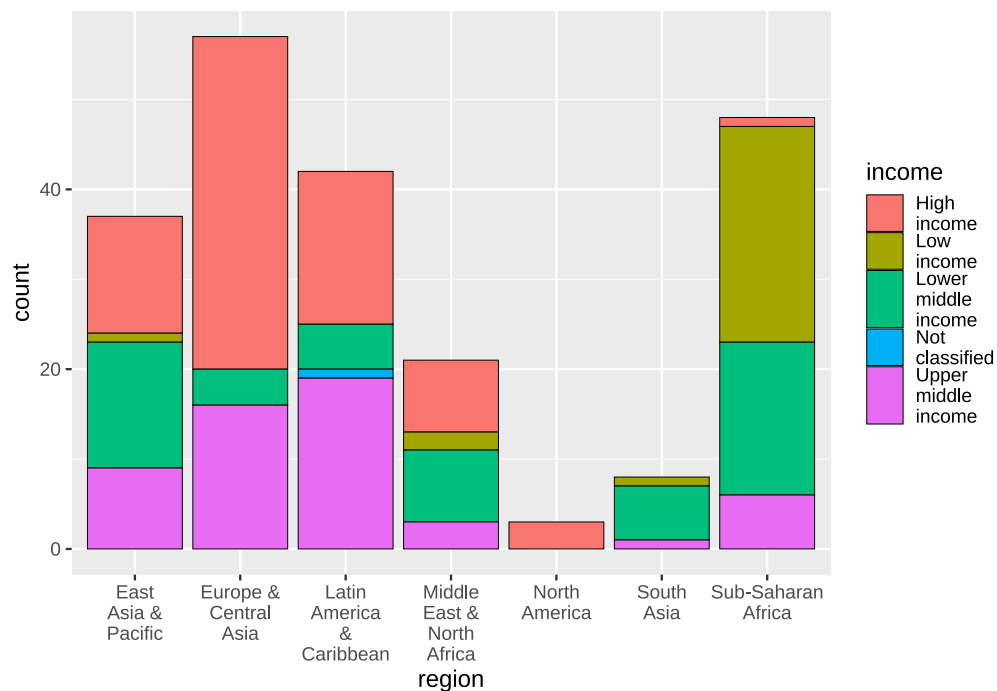

```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar() +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_fill_discrete(labels = function(x) str_wrap(x, width = 8))
```



19.5.3.4 棒グラフに黒枠

変わり目が見にくければ、黒線で囲むことも可能です。どこに、color = "black" を入れるか、注意が必要です。少しその線を細目にするために、linewidth = 0.2 を加えました。

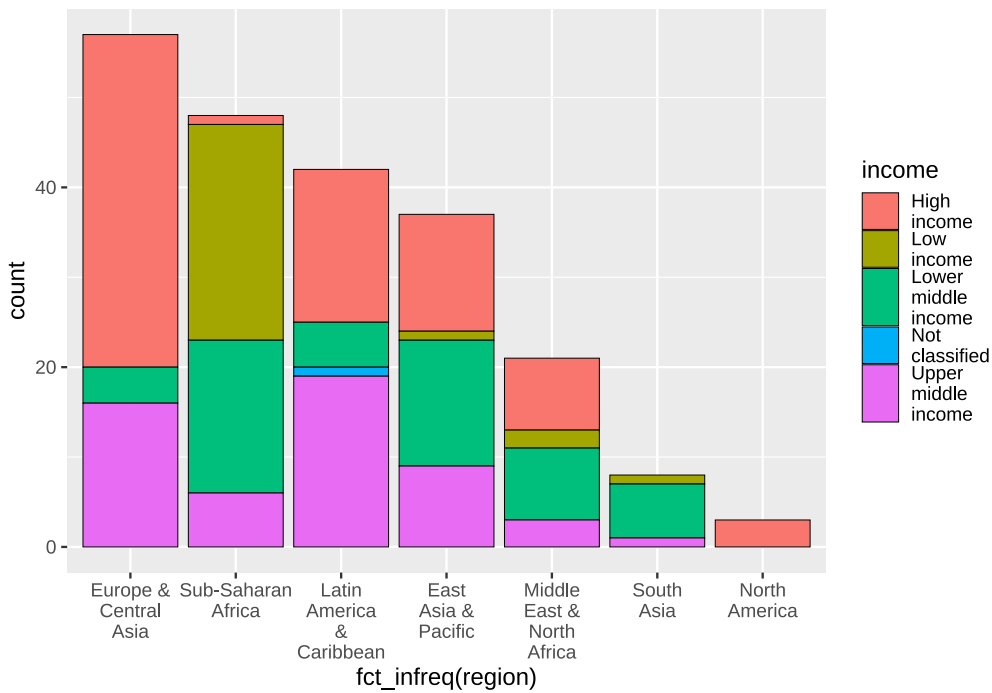
```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar(color = "black", linewidth = 0.2) +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_fill_discrete(labels = function(x) str_wrap(x, width = 8))
```



19.5.3.5 大きい順

順序は、地域のアルファベット順ですから、上のグラフで問題ありませんが、もし、大きい順、すなわち、国の数が大きい順にするとするとどうしたら良いでしょうか。少しだけ、すでに出てきた、Factor をいうものを使います。`fct_infreq(region)` この場合は、度数順に並べてグラフにします。

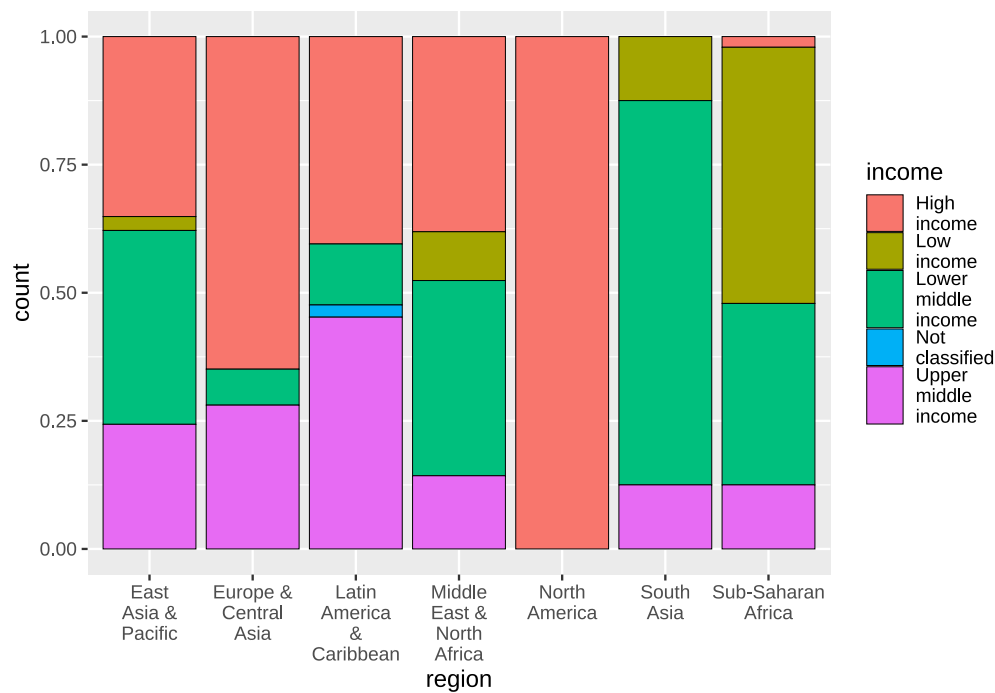
```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = fct_infreq(region), fill = income)) +
  geom_bar(color = "black", linewidth = 0.2) +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_fill_discrete(labels = function(x) str_wrap(x, width = 8))
```



19.5.3.6 割合の表示

積み上げ型になっています。これを、割合で表示するには `position = "fill"` を加えます。すると、各地域ごとの、income level (収入の多寡) による国の数の割合がわかります。

```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar(color = "black", linewidth = 0.2, position = "fill") +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_fill_discrete(labels = function(x) str_wrap(x, width = 8))
```



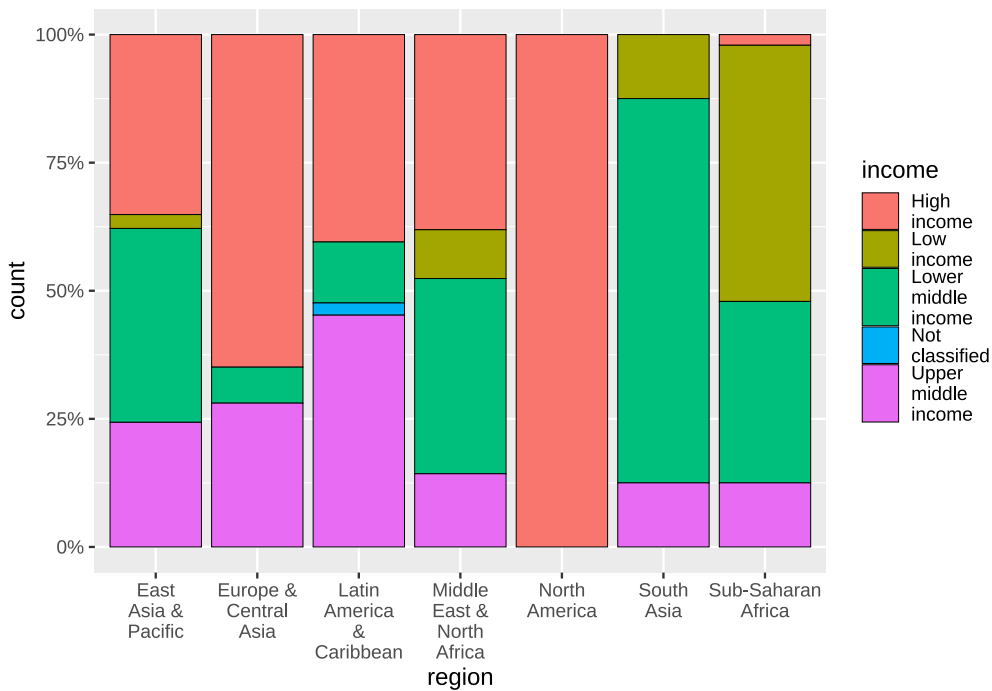
19.5.3.7 縦軸を百分率に

今度は、それを、百分率 (percent) 表示にしてみましょう。

```
labels = scales::percent_format(accuracy = 1)
```

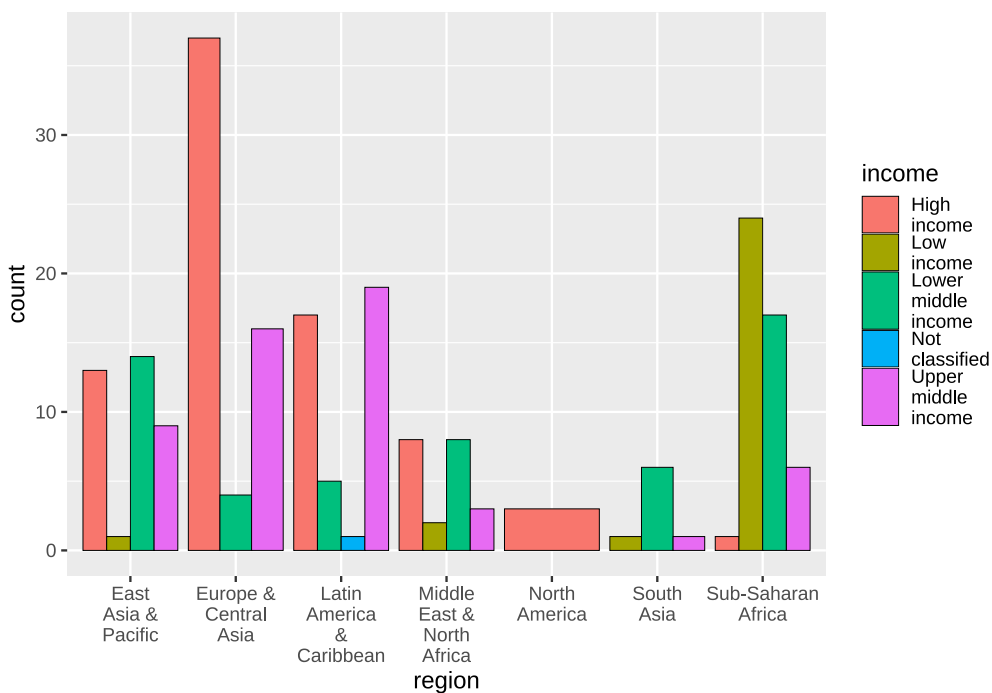
まず、scales は、Tidyverse パッケージの一部ですが、読み込まれてはいないので、加えてあります。library(scales) としてあれば、省略可能です。accuracy = 1 は小数点以下を省略するもので、小数点以下一位までであれば、accuracy = 0.1 とします。

```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar(color = "black", linewidth =
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_y_continuous(labels = scales::label_percent(accuracy = 1)) +
  scale_fill_discrete(labels = function(x) str_wrap(x, width = 8))
```



積み上げてありますが、並べることも可能です。ちょっと数が多いのでみにくいですが。

```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar(color = "black", linewidth = 0.2, position = "dodge") +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_fill_discrete(labels = function(x) str_wrap(x, width = 8))
```



19.5.4 折線グラフ (line graph)

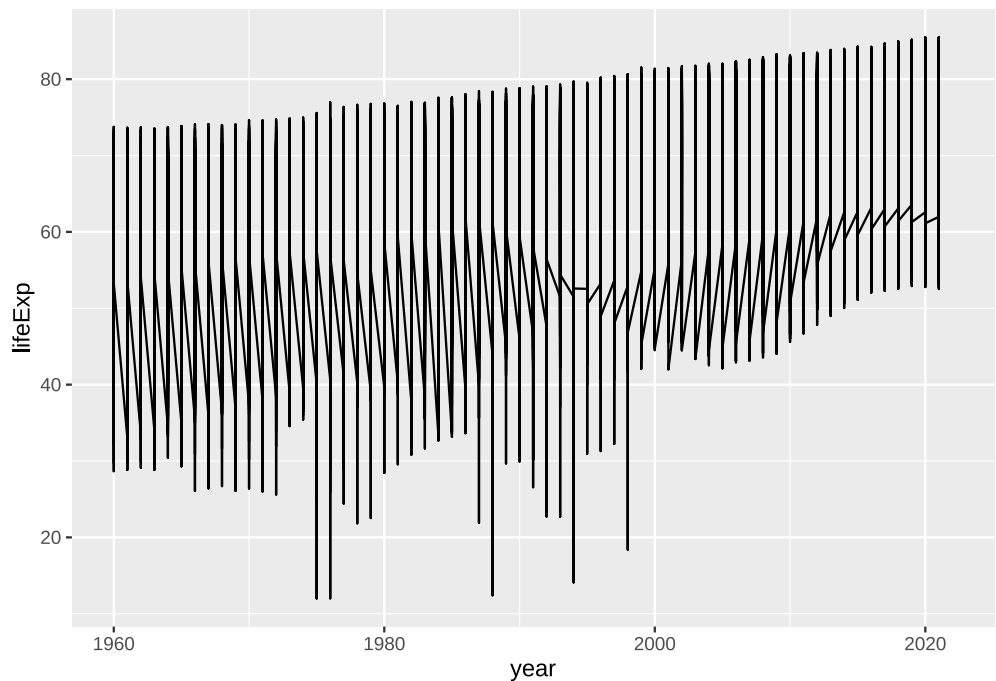
WDI は時系列データですから、折れ線グラフも使います。後ほど紹介しますが、

```
<DATA> |> ggplot(aes(year, lifeExp)) + geom_line()
```

と言った感じです。

まずは、失敗例から。次のコードでグラフが描けるでしょうか。

```
df_wdi |> ggplot(aes(year, lifeExp)) + geom_line()  
#> Warning: Removed 266 rows containing missing values  
#> (`geom_line()`).
```



何が起きているかわかりますか。これは、鋸の刃グラフ (saw-tooth chart) とされる標準的な失敗例です。

```
ggplot(df_wdi, aes(x = year, y = lifeExp)) + geom_boxplot()  
#> Warning: Continuous x aesthetic  
#> i did you forget `aes(group = ...)`?  
#> Warning: Removed 892 rows containing non-finite values  
#> (`stat_boxplot()`).
```

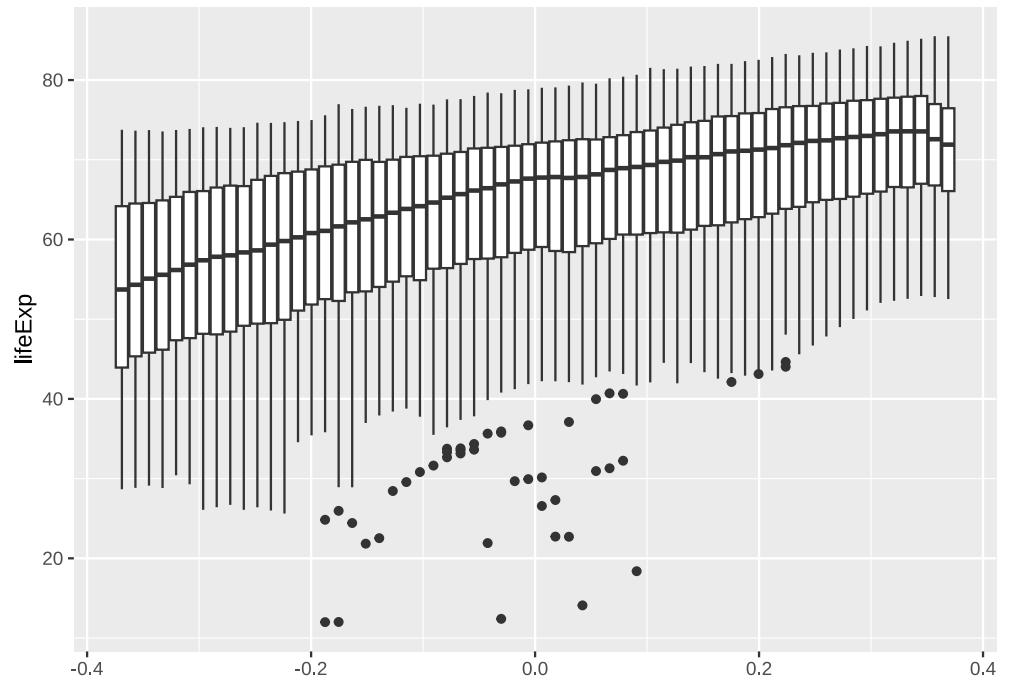


これも期待した箱ひげ図にはなっていません。年は、カテゴリではなく、数値データですね。

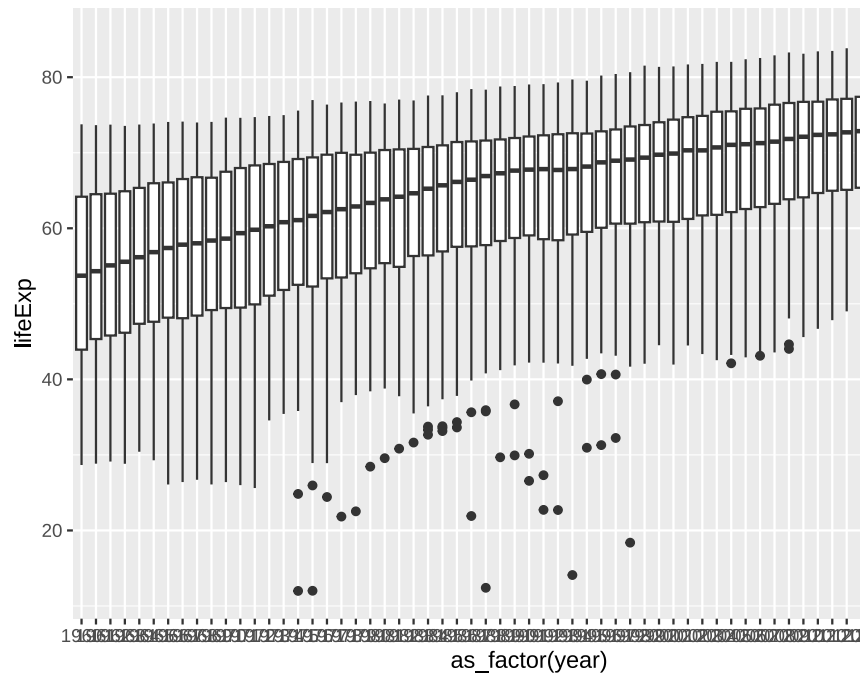
```
typeof(pull(df_wdi, year)) # same as typeof(df$year)
#> [1] "double"
```

次のようにすると少しマシになります。

```
ggplot(df_wdi, aes(y = lifeExp, group = year)) + geom_boxplot()
#> Warning: Removed 892 rows containing non-finite values
#> (`stat_boxplot()`).
```



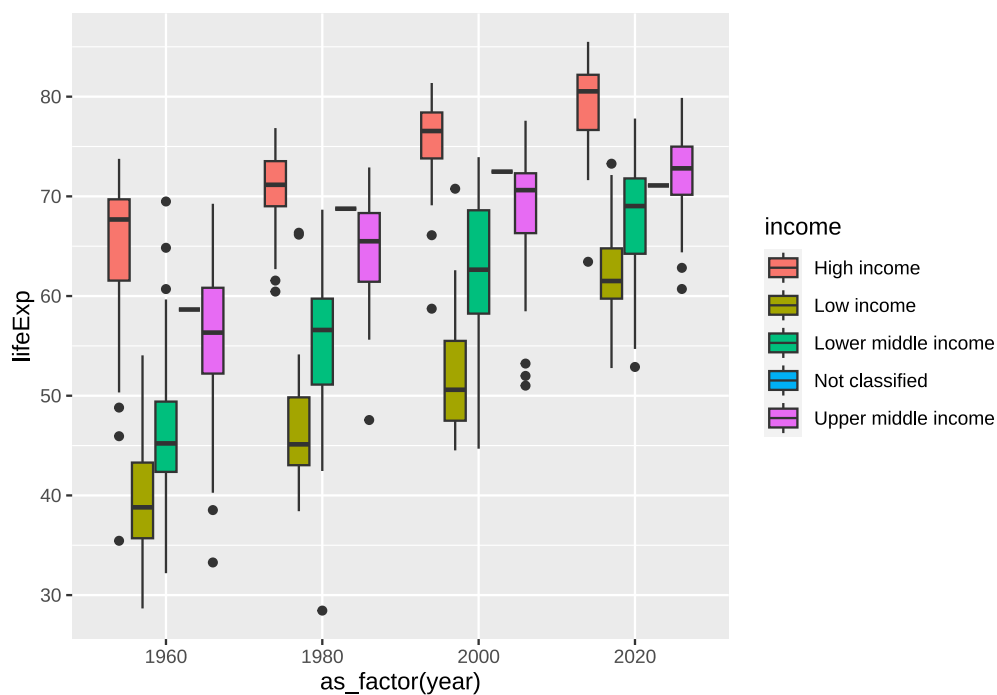
```
ggplot(df_wdi, aes(x = as_factor(year), y = lifeExp)) + geom_boxplot()
#> Warning: Removed 892 rows containing non-finite values
#> (`stat_boxplot()`).
```



■19.5.4.0.1 Box Plot

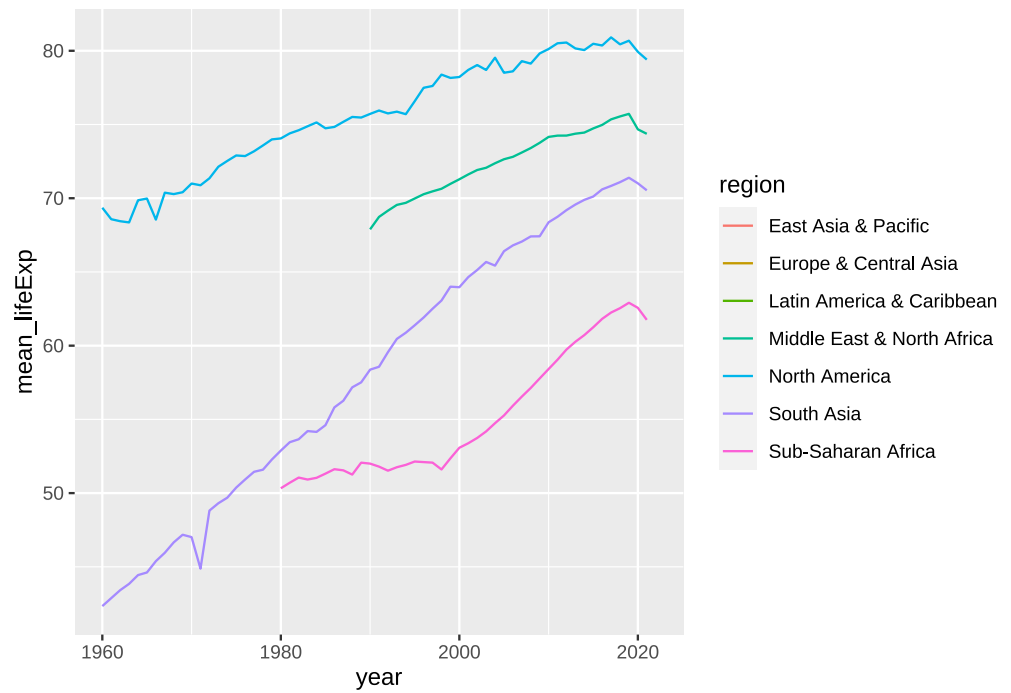
とはいえ、数が多すぎますね。色もつけてみましょう。塗りつぶしは、fill 枠の線に色をつけるのは、color ですから、ここでは、fill を使います。


```
df_wdi_extra |> filter(income != "Aggregates") |>
  filter(year %in% c(1960, 1980, 2000, 2020)) |>
  ggplot(aes(x=as_factor(year), y = lifeExp, fill = income)) +
  geom_boxplot()
#> Warning: Removed 37 rows containing non-finite values
#> (`stat_boxplot()`).
```

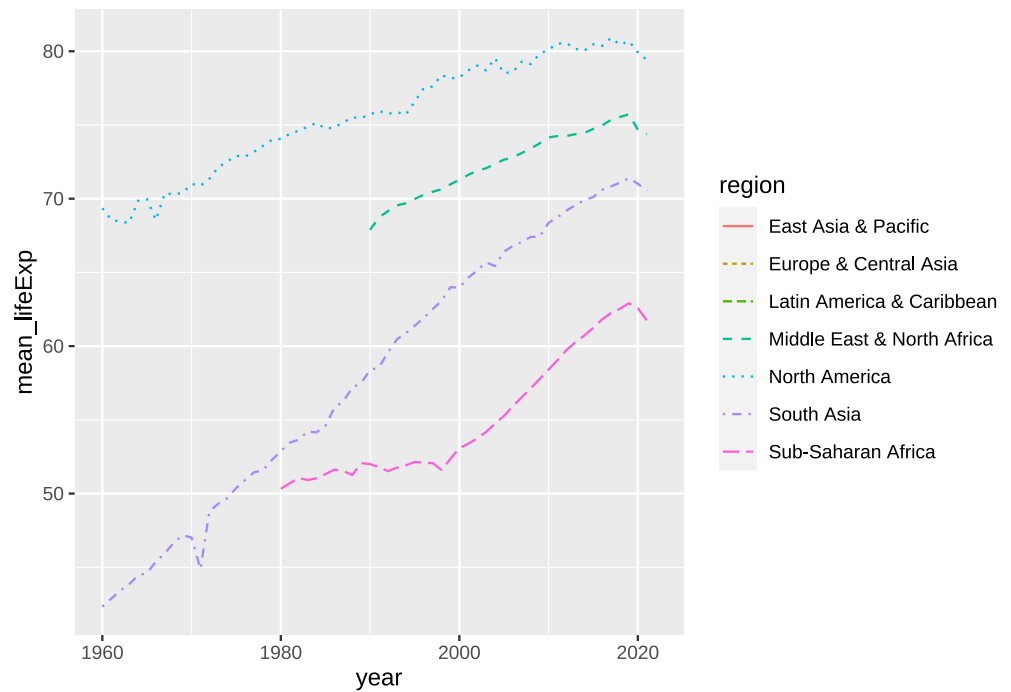


折線グラフの例としては次のようなものがあります。

```
df_lifeExp <- df_wdi_extra |> filter(region != "Aggregates") |>
  group_by(region, year) |>
  summarize(mean_lifeExp = mean(lifeExp), median_lifeExp = median(lifeExp), max_lifeExp = max(lifeExp),
df_lifeExp %>% ggplot(aes(x = year, y = mean_lifeExp, color = region)) +
  geom_line()
#> Warning: Removed 243 rows containing missing values
#> (`geom_line()`).
```

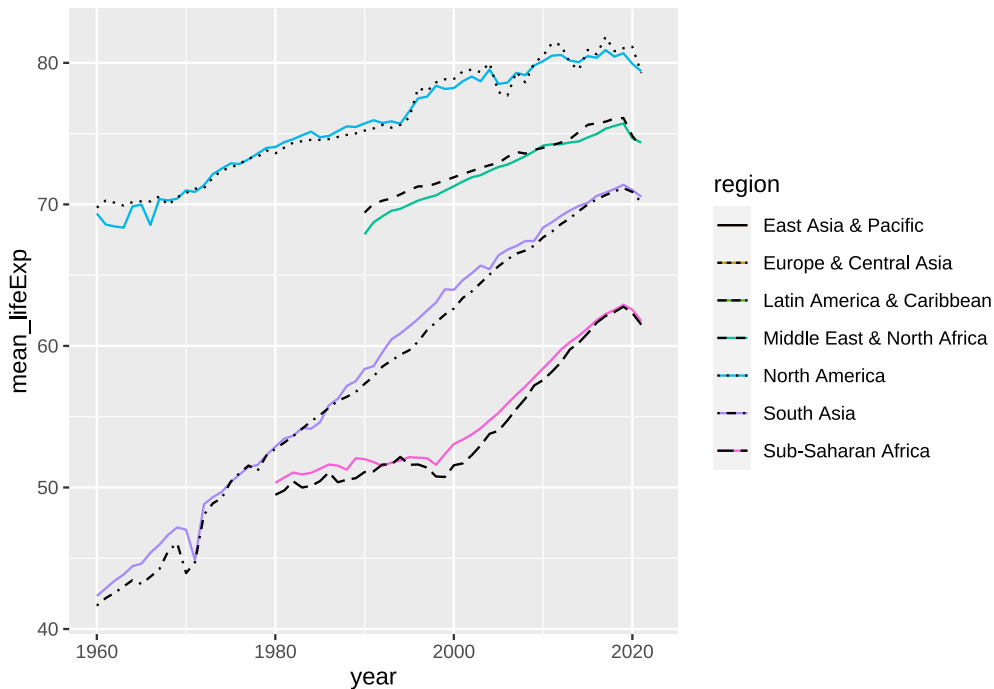


```
df_lifeExp %>% ggplot(aes(x = year, y = mean_lifeExp, color = region, linetype =
  geom_line()
#> Warning: Removed 243 rows containing missing values
#> (`geom_line()`).
```



```
df_lifeExp %>% ggplot() +
  geom_line(aes(x = year, y = mean_lifeExp, color = region)) +
  geom_line(aes(x = year, y = median_lifeExp, linetype = region))
```

```
#> Warning: Removed 243 rows containing missing values (`geom_line()`).
#> Removed 243 rows containing missing values (`geom_line()`).
```



19.6 コメント

19.6.1 参考文献

- Cheat Sheet in RStudio: <https://www.rstudio.com/resources/cheatsheets/>
 - RStudio IED
 - Base R Cheat Sheet
- ‘Quick R’ by DataCamp: <https://www.statmethods.net/management>
- An Introduction to R

19.7 練習

19.7.1 Posit Primers <https://posit.cloud/learn/primers>

1. The Basics – r4ds: Explore, I
 - Visualization Basics
 - Programming Basics

第 20 章

データ読み込み (Import)

20.1 はじめに

データの読み込みは、データサイエンスの最初のステップです。しかし、データを読み込む時に、その後の、データの整形・変形、さらには、視覚化について、ある程度の理解をしているかどうかで、変わってきますので、順番を、Tidyverse での、dplyr をつかった変形と、ggplot2 を使った視覚化のあとにしました。

個々のデータの読み込みは、それぞれのサイトに特徴的なこともありますし、R のパッケージ (API: Application Programming Interface を用いたプログラム群) を用いて読み込む場合などによって、異なりますので、それは、第三部で扱いたいと思います。

20.2 概略

まずは、基本を確認しましょう。

すでに、WDI パッケージを用いて、データを読み込みますが。同様に、他にも、それぞれのサイトが、コンピュータで直接データを取得できるようにするために、提供している、API (Application Programming Interface) を用いたパッケージが存在します。十分な質のものが提供されている場合は、それを使うのが適切だと思います。

そのようなパッケージが提供されていない場合には、データファイルを取得して読み込むことになります。この場合にも、データファイルの URL (universal resource locator) から直接取り込むことができる場合と、一旦、自分のコンピュータにダウンロードして、それから、読み込む場合とがあります。

再現性 (reproducibility) を考えると、URL から、直接読み込む方がよいと思いますが、そうはいかない場合、また、自分で作成した、データファイルもあるでしょうから、その場合は、コンピュータから読み込むことになります。

データの形式はさまざまですが、オープンデータでは、CSV (comma separated values) 形

式のテキストデータ^{*1}か、Microsoft の Excel 形式のデジタルデータのどちらかを提供している場合がほとんどです。ですから、どちらの形式のファイルなのかを確認することが必要です。

```
library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats    1.0.0      v stringr    1.5.0
#> v ggplot2    3.4.3      v tibble     3.2.1
#> v lubridate  1.9.2      v tidyr      1.3.0
#> v purrr      1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()   masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all con
library(readxl)
library(WDI)
library(owidR)
library(wid)
```

上で説明した三つの方法を簡単にまとめておきます。

20.2.1 テキストファイル (CSV など) の読み込み

- `readr` パッケージで読み込みます。tidyverse パッケージ群に含まれる基本パッケージなので、`library(tidyverse)` としてあれば使えます。データ間の分離記号の種類によって使う関数が異なります。
 - `read_csv()`: comma-separated values (CSV) - カンマの場合
 - `read_tsv()`: tab-separated values (TSV) - タブの場合
 - `read_csv2()`: semicolon-separated values with `,` as the decimal mark - セミコロン; の場合で、少数点にカンマが使われている場合 (ヨーロッパでよく使われています)
 - `read_delim()`: delimited files (CSV and TSV are important special cases)
 - 分離記号を指定する場合
 - `read_fwf()`: fixed-width files - 固定幅でデータが並んでいる場合

^{*1} Separator 分離記号として、comma (カンマ) 以外 (space (スペース)、tab (タブ)、bar (縦棒) など) を用いる場合もありますが、それらは、ほとんど問題なく、CSV の場合と同様に読み込みます。これらについては、簡単に言及します。

- `read_table()`: whitespace-separated files - 空白が分離記号に使われている場合
 - `read_log()`: web log files - ログファイルを読み込む場合
 - 例: `df_name <- read_csv("./data/datafile.csv")`
- 詳細は、[readr](#) サイトを参照してください。サイトには、Data Importing CheatSheet もあります。

20.2.2 Excel ファイルの読み込み

- `readxl` パッケージで読み込みます。`tidyverse` パッケージ群に含まれていますから、新たに、インストールする必要はありませんが、基本パッケージではないので、`library(readxl)` としてから使います。
 - 例: `df_name <- read_excel("./data/datafile.xlsx")`
 - Excel ファイルには、xls 形式と、xlsx 形式があります。どちらも、`read_excel` で読み込むことが可能ですが、それぞれの形式に特化した `read_xls` と `read_xlsx` も用意されています。
 - Excel ファイルには、複数のシートが含まれる場合があります。そのときには、シートを、何番目のシートか (例: `sheet = 2`)、または、シート名を指定 (例: `sheet = "name"`) して読み込みます。
 - シートの一部を読み込むような場合もあり、その場合は、指定が必要です。Help や、リンク、または、後から示す例を参照してください。
- 詳細は、[readxl](#) サイトを参照してください。サイトには、Data Importing CheatSheet もあります。

20.2.3 データのダウンロード

- データの URL がわかっている場合は、直接、データを取り込んだり、`data` ディレクトリ (フォルダー) にダウンロードすることができます。ダウンロードしたら、テキストファイルか、Excel ファイルかによって、上で説明した方法を使って読み込んでください。
- 圧縮ファイルになっている場合もあります。圧縮ファイルも R 起動時に読み込まれる `util` パッケージで解凍できますが、基本的に、コンピュータのプログラムを使いますから、ファイルをダブルクリックするなどして、解凍する場合と同じです。
- CSV のようなテキストふぁいと、Excel ファイルのようなデジタルファイルで少しだけ異なります。下に例を書きます。いずれも、`data` ディレクトリ (フォルダー) に保存しています。詳しくは、`download.file` の Help を参照してください。

- テキストファイル (CSV) : `download.file(url = "URL", destfile =
"./data/file_name.csv")`
- デジタルファイル (Excel) : `download.file(url = "URL", destfile =
"./data/file_name.xlsx", mode = "wb")`
 - * `mode = "wb"` デジタルファイルの指定です。Mac の場合には、リソース
ファイルと呼ばれる不可視ファイルを使っているため、この設定がなくて
も、読み込めるようです。
- URL がわかっている、CSV ファイルの場合には、直接、`readr` でも読み込むこと
ができます。
 - 例 : `df_name <- read_csv(URL)`
- URL がわかっていて、`read_csv` などで読み込んだ場合も、保存しておくことをお
勧めします。
 - 保存 : `write_csv(df_name, "./data/df_name.csv")`
 - 読み込み : `df_name <- read_csv("./data/df_name.csv")`

20.2.4 パッケージの利用

- `WDI`, `wir`, `eurostat`, `estatapi` など
 - 例 : `wdi_shortcode <- WDI(indicator = "indicator's name")`
- 毎回、読み込むのは時間もかかりますし、サイト負荷も増えますから、`data ディレ
クトリ (フォルダ)` に書き込み、そこから、読み込むのが適切です。
 - 保存 : `write_csv(wdi_shortcode, "./data/wdi_shortcode.csv")`
 - 読み込み : `wdi_shortcode <- read_csv("./data/wdi_shortcode.csv")`

20.3 データの整形 tidy

R にとって大切な整形されたデータ (tidy data) について学び、Tidyverse パッケ
ージ群の主要なパッケージの一つ `tidyr` について学びます。特に、`pivot_longer`
とその逆変換である、`pivot_wider` の使い方について学びます。

20.3.1 変数、値、観測

データには、変数があり、それぞれの観測における、その変数に関する観測値がありあ
す。それを、表に記したものが、データ表 (Data Table) です。

- 変数 (variable) は、量的または質的な値を、測定することが可能なもの
- 値 (観測値 value) は、観測し計測した時の測定値

- 観測または場合 (observation, case) ある一定のとき、ある一定の条件のもとで、それぞれの変数について行う測定
- データ表 (Tabular data) は、観測に基づく、それぞれの変数の値を表にしたもの

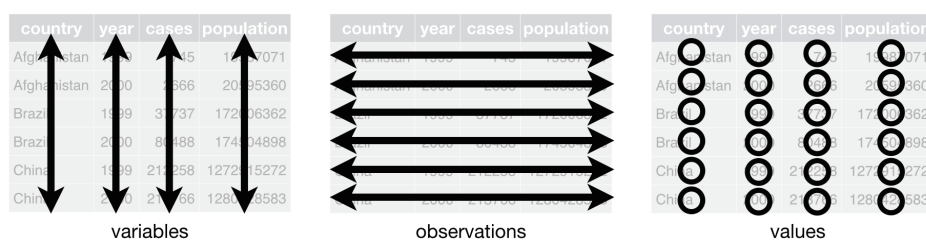
20.3.2 整形済みデータ (tidy data)

整形済みの (データ) 表とは、それぞれの値が表の一つの枠 (ある行ある列) に割り当てられ、変数が列に、それぞれの観測が行に割り当てられたもの

今までは、整形済みのデータを多くみてきましたが、データ表は、必ずしも、整形済みではなく、整形済みのデータが良いデータ表というわけではありません。しかし、**R** で分析をするときには、整形済みのデータ表の概念を理解し、整形済みのデータに `dplyr` などを使って変形しておくことが重要なステップになります。

まとめると、整形済みデータ (表) とは：

1. 各変数が一つの列に対応 (Each variable is in its own column)
2. 各観測・場合が一つの行に対応 (Each observation is in its own row)
3. ある変数のある観測における値が枠 (セル) に書き込まれてる (1,2 から自動的に従うことです) (Each value is in its own cell (this follows from #1 and #2))



Tidyverse の中心的な開発者である H. ウィッカム (Hadley Wickham) は次のように言っています。

“Tidy data sets are all alike; but every messy data set is messy in its own way.”
— Hadley Wickham

訳：整形済みのデータは似たり寄ったりだが、整形されていないデータの乱雑さはそれぞれである。(H. ウィッカム)

“all happy families are all alike; each unhappy family is unhappy in its own way” - Tolstoy’s Anna Karenina

幸せな家族はみな似たり寄ったりだが、不幸な家族は、それぞれに不幸だ。(トルストイ「アンナ・カレーニナ」より)

トルストイの小説「アンナ・カレーニナ」の冒頭の言葉から変形したのですが、実際の世の中を表現するものとして、さまざまな分野で引用されます。データサイエンスでは、

アンナ・カレニナの法則 (Anna Karenina Principle (AKP)) とも呼ばれています。

ある指標に関して、平均値や中央値付近のケースは、似たり寄ったりの状況が多いが、外れ値を取るようなケースは、一つの理由ではなく、さまざまな原因で外れ値となっており、それぞれの原因を考察することは、背景を理解する上で重要だという意味にとれると、わたしは理解しています。

20.3.3 整形済みデータと整形済みでないデータ

`tidyr` パッケージには、例がデータとして含まれていますので、`table1`, `table2`, `table3`, `table4a`, `table4b`, `table5` をまずはみてみましょう。

`table1`

```
#> # A tibble: 6 x 4
#>   country      year cases population
#>   <chr>      <dbl> <dbl>      <dbl>
#> 1 Afghanistan 1999     745  19987071
#> 2 Afghanistan 2000    2666  20595360
#> 3 Brazil      1999  37737  172006362
#> 4 Brazil      2000   80488  174504898
#> 5 China       1999 212258 1272915272
#> 6 China       2000 213766 1280428583
```

`table2`

```
#> # A tibble: 12 x 4
#>   country      year type          count
#>   <chr>      <dbl> <chr>          <dbl>
#> 1 Afghanistan 1999 cases           745
#> 2 Afghanistan 1999 population 19987071
#> 3 Afghanistan 2000 cases           2666
#> 4 Afghanistan 2000 population 20595360
#> 5 Brazil      1999 cases           37737
#> 6 Brazil      1999 population 172006362
#> 7 Brazil      2000 cases           80488
#> 8 Brazil      2000 population 174504898
#> 9 China       1999 cases           212258
#> 10 China      1999 population 1272915272
#> 11 China      2000 cases           213766
#> 12 China      2000 population 1280428583
```

`table3`

```
#> # A tibble: 6 x 3
#>   country      year rate
#>   <chr>      <dbl> <chr>
```

```
#> 1 Afghanistan 1999 745/19987071
#> 2 Afghanistan 2000 2666/20595360
#> 3 Brazil      1999 37737/172006362
#> 4 Brazil      2000 80488/174504898
#> 5 China       1999 212258/1272915272
#> 6 China       2000 213766/1280428583
```

table4a

```
#> # A tibble: 3 x 3
#>   country   `1999` `2000`
#>   <chr>     <dbl> <dbl>
#> 1 Afghanistan 745 2666
#> 2 Brazil      37737 80488
#> 3 China      212258 213766
```

table4b

```
#> # A tibble: 3 x 3
#>   country   `1999`   `2000`
#>   <chr>     <dbl>    <dbl>
#> 1 Afghanistan 19987071 20595360
#> 2 Brazil      172006362 174504898
#> 3 China      1272915272 1280428583
```

table5

```
#> # A tibble: 6 x 4
#>   country century year rate
#>   <chr>     <chr> <chr> <chr>
#> 1 Afghanistan 19 99 745/19987071
#> 2 Afghanistan 20 00 2666/20595360
#> 3 Brazil      19 99 37737/172006362
#> 4 Brazil      20 00 80488/174504898
#> 5 China       19 99 212258/1272915272
#> 6 China       20 00 213766/1280428583
```

どうでしょうか。どれが、整形済みか、どれが整形済みでないかわかりましたか。何を持って、整形済みではないとしているのでしょうか。

ここでは、table1 は整形済み、それ以外は、整形済みでないとして、table1 のように変形するにはどうしたら良いかを考えていきます。

簡単に理由を書いておきます。

- table2: count とありますが、人口 (population) と場合 (case) を一つの物差しでは計れません。

- table3: 割合 (率) が書かれていますが、同時に、この表記で、人口と、場合を同時に表そうとしています。それぞれの観測において、一つの物差しでの計測値ではありません。
- table4a, table4b: どちらも、年毎に別の列になっています。観測値として、同時に計測できるわけではありません。また、これらを、統合する必要もあるでしょう。
- table5: 年が二つに分かれています。このように分けることが便利なこともあるでしょうが、まとまっていた方が自然です。

20.3.4 tidyr による整形の基礎 Pivoting

長型データ (long data) と、幅広型データ (wide data) の変換です。pivoting はある箇所を支点として、回転または折り返すという意味です。Excel にも同様の機能があります。以前は、pivot_longer() は、gather()、pivot_wider() は、spread() という関数が使われていましたから、少し前の文献には、そちらが使われているかもしれません。Posit Primers でもまだ、古いものが使われているようです。

設定値 (arguments) やその初期値 (default) については、Help をみてください。下には、もっとも簡単なもののみを書きます。

- pivot_longer(): Pivot data from wide to long

```
pivot_longer(data = <data>, cols = <column>,
             names_to = <new column>, values_to = <value column>)
```

```
pivot_longer(data = table4a, cols = c(1999,2000),
             names_to = "year", values_to = "population" )
```

- pivot_wider(): Pivot data from long to wide,

```
pivot_wider(data = <data>, names_from = <specified column>,
            values_from = <value column>)
```

```
pivot_wider(data = table2, names_from = type,
            values_from = count)
```

20.3.4.1 例 1: pivot_longer

年 (cols = c("1999", "2000")) を year (names_to = "year") という列にまとめ、それぞれの値を、values_to = "cases" にします。同時に、年が、連続変数の dbl ですから、それを離散変数 (飛び飛びの値をとる) int (整数に変形します)

```
table4a
#> # A tibble: 3 x 3
#>   country   `1999` `2000`
#>   <chr>     <dbl> <dbl>
```

```
#> 1 Afghanistan    745    2666
#> 2 Brazil          37737   80488
#> 3 China           212258  213766
```

```
pivot_longer(data = table4a, cols = c("1999", "2000"),
             names_to = "year", values_to = "cases",
             names_transform = list("year" = as.integer))
```

```
pivot_longer(data = table4a, cols = c("1999", "2000"),
             names_to = "year", values_to = "cases",
             names_transform = list("year" = as.integer))
```

```
#> # A tibble: 6 x 3
#>   country    year cases
#>   <chr>      <int> <dbl>
#> 1 Afghanistan 1999    745
#> 2 Afghanistan 2000   2666
#> 3 Brazil      1999  37737
#> 4 Brazil      2000  80488
#> 5 China       1999 212258
#> 6 China       2000 213766
```

20.3.4.2 例 2: pivot_wider

type (names_from = type) を二種類の変数に分け、値を cases (values_from = count) からとります。

```
table2
#> # A tibble: 12 x 4
#>   country    year type          count
#>   <chr>      <dbl> <chr>          <dbl>
#> 1 Afghanistan 1999 cases            745
#> 2 Afghanistan 1999 population 19987071
#> 3 Afghanistan 2000 cases            2666
#> 4 Afghanistan 2000 population 20595360
#> 5 Brazil      1999 cases            37737
#> 6 Brazil      1999 population 172006362
#> 7 Brazil      2000 cases            80488
#> 8 Brazil      2000 population 174504898
#> 9 China       1999 cases            212258
#> 10 China      1999 population 1272915272
#> 11 China      2000 cases            213766
#> 12 China      2000 population 1280428583
```

```
pivot_wider(data = table2, names_from = type,
```

```

values_from = count)

pivot_wider(data = table2, names_from = type,
             values_from = count)

#> # A tibble: 6 x 4
#>   country      year cases population
#>   <chr>      <dbl> <dbl>      <dbl>
#> 1 Afghanistan 1999     745  19987071
#> 2 Afghanistan 2000    2666  20595360
#> 3 Brazil      1999   37737  172006362
#> 4 Brazil      2000   80488  174504898
#> 5 China       1999  212258 1272915272
#> 6 China       2000  213766 1280428583

```

20.3.4.3 例 3: separate

分け方はいろいろとありますが、ここでは、“/” で分けることにします。正規表現を使うことでかなり複雑な分離方法も可能です。参照

```

table3

#> # A tibble: 6 x 3
#>   country      year rate
#>   <chr>      <dbl> <chr>
#> 1 Afghanistan 1999 745/19987071
#> 2 Afghanistan 2000 2666/20595360
#> 3 Brazil      1999 37737/172006362
#> 4 Brazil      2000 80488/174504898
#> 5 China       1999 212258/1272915272
#> 6 China       2000 213766/1280428583

```

```

separate(table3, rate, c("cases", "population"),
          sep = "/", convert = TRUE)

```

```

separate(table3, rate, c("cases", "population"),
          sep = "/", convert = TRUE)

#> # A tibble: 6 x 4
#>   country      year cases population
#>   <chr>      <dbl> <int>      <int>
#> 1 Afghanistan 1999     745  19987071
#> 2 Afghanistan 2000    2666  20595360
#> 3 Brazil      1999   37737  172006362
#> 4 Brazil      2000   80488  174504898
#> 5 China       1999  212258 1272915272
#> 6 China       2000  213766 1280428583

```

20.3.4.4 例 4: unite

結合します。参照

```
table5
#> # A tibble: 6 x 4
#>   country    century year rate
#>   <chr>      <chr>  <chr> <chr>
#> 1 Afghanistan 19      99    745/19987071
#> 2 Afghanistan 20      00    2666/20595360
#> 3 Brazil      19      99    37737/172006362
#> 4 Brazil      20      00    80488/174504898
#> 5 China       19      99    212258/1272915272
#> 6 China       20      00    213766/1280428583
```

```
table5 %>% unite(col = "year", century, year, sep = "") %>%
  separate(rate, c("cases", "population"), sep = "/",
           convert = TRUE) %>%
  mutate(year = as.integer(year), rate = cases / population)
```

```
table5 %>% unite(col = "year", century, year, sep = "") %>%
  separate(rate, c("cases", "population"), sep = "/",
           convert = TRUE) %>%
  mutate(year = as.integer(year), rate = cases / population)
#> # A tibble: 6 x 5
#>   country    year cases population    rate
#>   <chr>      <int> <int>      <int>    <dbl>
#> 1 Afghanistan 1999     745  19987071 0.0000373
#> 2 Afghanistan 2000    2666  20595360 0.000129
#> 3 Brazil      1999   37737  172006362 0.000219
#> 4 Brazil      2000   80488  174504898 0.000461
#> 5 China       1999  212258  1272915272 0.000167
#> 6 China       2000  213766  1280428583 0.000167
```

20.3.4.5 例 5: bind_rows

二つの表を結合します。行や列の結合：参照

```
table4a; table4b
#> # A tibble: 3 x 3
#>   country    `1999` `2000`
#>   <chr>      <dbl> <dbl>
#> 1 Afghanistan    745    2666
#> 2 Brazil        37737  80488
```

```

#> 3 China      212258 213766
#> # A tibble: 3 x 3
#>   country      `1999`      `2000`
#>   <chr>         <dbl>         <dbl>
#> 1 Afghanistan 19987071     20595360
#> 2 Brazil      172006362    174504898
#> 3 China      1272915272   1280428583

```

```

tables <- list(cases = table4a, population = table4b)
tables %>% bind_rows(.id = "type")
#> # A tibble: 6 x 4
#>   type      country      `1999`      `2000`
#>   <chr>    <chr>         <dbl>         <dbl>
#> 1 cases    Afghanistan      745           2666
#> 2 cases    Brazil           37737          80488
#> 3 cases    China            212258         213766
#> 4 population Afghanistan 19987071     20595360
#> 5 population Brazil      172006362    174504898
#> 6 population China      1272915272   1280428583

```

```

tables <- list(cases = table4a, population = table4b)
tables %>% bind_rows(.id = "type") %>%
  pivot_longer(cols = c("1999", "2000"), names_to = "year") %>%
  pivot_wider(names_from = "type", values_from = "value") %>%
  mutate(year = as.integer(year)) %>% arrange(country)
#> # A tibble: 6 x 4
#>   country      year cases population
#>   <chr>         <int> <dbl>         <dbl>
#> 1 Afghanistan 1999      745     19987071
#> 2 Afghanistan 2000     2666     20595360
#> 3 Brazil      1999    37737    172006362
#> 4 Brazil      2000    80488    174504898
#> 5 China       1999  212258   1272915272
#> 6 China       2000  213766   1280428583

```

20.4 参照: tidyr

- Textbook: R for Data Science, Tidy Data
- Reference: <https://tidyr.tidyverse.org>

20.4.1 RStudio Primers: See References in Moodle at the bottom

4. Tidy Your Data – r4ds: Wrangle, II

- Reshape Data - a bit old
- Separate and Unite
- Join Data Sets

The first component, ‘Reshape Data’ deals with `pivot_longer` and `pivot_wider`. However, it uses an older version of these functions calls `gather` and `spread`.

20.5 二つの表の結合

最初に紹介した、世界開発指標（WDI）は、同じ規格でデータを提供してくれていますし、WDI パッケージを使えば、いくつもの指標について同時に入手することもできます。そのようなデータベースに、広範囲の 1400 以上の分野からの指標が提供され、毎年定められたときに、更新してくれています。その意味でも、初心者にはとても親切なデータベースです。

しかし、一般的に、データサイエンスにおいては、2つの異なる出処（ソース）のデータを一つにまとめて行うことで、世界が広がり、独自の、データサイエンスが始められるとも言えます。

ここでは、そのための基本的な手法を学びます。変形のところで少し書きましたが、Tidyverse パッケージ群の、`dplyr` パッケージでは、一つのデータ表の変形だけではなく、2つ以上のデータ表も扱えます。基本的には、2つのデータ表を一つにすることができれば、それに3つ目のデータ表を加えていけば良いので、2つの場合を説明します。

整理されたデータ表 (tidy data) について上に書きましたが、それぞれのデータ表は、整理されているものとして話を進めます。すでに、上で、`unite` の説明をしてありますから、参考にしてください。

この部分は、Two Table Verbs として、`dplyr` の機能にある部分ですが、ここで紹介します。

- R for Datascience (2ed) Joins: <https://r4ds.hadley.nz/joins#introduction>
- Two Table Verbs: <https://dplyr.tidyverse.org/articles/two-table.html>
- Vignette: <https://cran.r-project.org/web/packages/dplyr/vignettes/two-table.html>
- 早見表 (p.2) Cheat sheet: <https://github.com/rstudio/cheatsheets/blob/main/data-transformation.pdf>

20.5.1 三種類の結合

- 融合結合：適合するものを他から読み込む：Mutating joins, which add new variables to one table from matching rows in another.
- 抽出結合：同一のもの相異なるものを抽出：Filtering joins, which filter observations from one table based on whether or not they match an observation in the other table.
- 集合算結合：データ集合として結合：Set operations, which combine the observations in the data sets as if they were set elements.

20.5.2 融合結合 (mutating joins)

いずれの場合にも両方の表に共通の key を定め、key を媒介として結合する。たとえば、a 列が共通なら、`by = "a"`、a, b 列が共通に存在する場合は、`by = c("a", "b")` とし、x の a と、y の c, x の b と y の d が対応する場合は、`by = c("a" = "c", "b" = "d")` などとする。共通するものが明らかな場合は、省略することも可能。

- `left_join(x,y)`: x をもとに、y を加える
 - 例：`left_join(x,y, by = "a")`
 - b, c 列も共通の場合は、b.1, b.2, c.1, c.2 などとなる。必要なものだけにして結合したほうが安全である。
- `right_join(x,y)`: y をもとに、x を加える
- `full_join(x,y)`: x も y も含める

20.5.3 抽出結合 (Filtering joins)

- `anti_join()`
 - `anti_join(x,y, ...)`: x の行で、y に含まれないもの (return all rows from x without a match in y.)
- `semi_join()`
 - `semi_join(x,y, ...)`: x の行で、y に含まれるもの (return all rows from x with a match in y.)

20.5.4 集合算結合 (Set joins)

- `intersect(x, y)`: return only observations in both x and y
- `union(x, y)`: return unique observations in x and y
- `setdiff(x, y)`: return observations in x, but not in y.

20.5.5 例

以下では、三種類の結合の例を見ます。Data Transformation with dplyr の早見表の2ページ目の例も参照してください。

```
(df_1 <- tibble(A = c('a', 'b', 'c'), B = c('t', 'u', 'v'), C = c(1,2,3)))
#> # A tibble: 3 x 3
#>   A     B     C
#>   <chr> <chr> <dbl>
#> 1 a     t     1
#> 2 b     u     2
#> 3 c     v     3
```

```
(df_2 <- tibble(A = c('a', 'b', 'c'), F = c('t', 'u', 'w'), G = c(3,2,1)))
#> # A tibble: 3 x 3
#>   A     F     G
#>   <chr> <chr> <dbl>
#> 1 a     t     3
#> 2 b     u     2
#> 3 c     w     1
```

```
(df_3 <- tibble(A = c('a', 'C', 'd'), B = c('t', 'u', 'w'), C = c(1,3,4)))
#> # A tibble: 3 x 3
#>   A     B     C
#>   <chr> <chr> <dbl>
#> 1 a     t     1
#> 2 C     u     3
#> 3 d     w     4
```

df_1, df_2 では、A 列は共通ですから、何も指定しないと、by = 'A' を仮定して結合します。

```
left_join(df_1, df_2)
#> Joining with `by = join_by(A)`
#> # A tibble: 3 x 5
#>   A     B     C F     G
#>   <chr> <chr> <dbl> <chr> <dbl>
#> 1 a     t     1 t     3
#> 2 b     u     2 u     2
#> 3 c     v     3 w     1
```

```
left_join(df_1, df_2, by = "A")
#> # A tibble: 3 x 5
#>   A     B     C F     G
```

```
#> <chr> <chr> <dbl> <chr> <dbl>
#> 1 a      t          1 t          3
#> 2 b      u          2 u          2
#> 3 c      v          3 w          1
```

by = c("A", "B"="F") としましたが、B と F では一致していない箇所があります。そこで、一致していない場所は、左側の表 df_1 はそのまま C 列を読み込みますが、右側の表 df_2 は、その部分を NA にして読み込んでいません。つまり、一致した行のみ読み込んでいます。

```
left_join(df_1, df_2, by = c("A", "B"="F"))
#> # A tibble: 3 x 4
#>   A      B      C      G
#>   <chr> <chr> <dbl> <dbl>
#> 1 a      t          1      3
#> 2 b      u          2      2
#> 3 c      v          3     NA
```

A 列も一致していますが、それは指定していないので、それは別と考えて、A.x, A.y と列名を別にして、かつ左側の表をもとにして読み込んでいます。

```
left_join(df_1, df_2, by = c("B"="F"))
#> # A tibble: 3 x 5
#>   A.x      B      C A.y      G
#>   <chr> <chr> <dbl> <chr> <dbl>
#> 1 a      t          1 a          3
#> 2 b      u          2 b          2
#> 3 c      v          3 <NA>      NA
```

roght_join ですから、今度は、右側の表 df_2 をもとにして読み込みます。B 列には、F 列が入っています。

```
right_join(df_1, df_2, by = c("A", "B"="F"))
#> # A tibble: 3 x 4
#>   A      B      C      G
#>   <chr> <chr> <dbl> <dbl>
#> 1 a      t          1      3
#> 2 b      u          2      2
#> 3 c      w          NA      1
```

左側の表 df_1 と右側の表 df_2 が一致していない行は、B と F の値によって分けて、読み込んでいます。

```
full_join(df_1, df_2, by = c("A", "B"="F"))
#> # A tibble: 4 x 4
```

```
#>   A     B     C     G
#>   <chr> <chr> <dbl> <dbl>
#> 1 a     t         1     3
#> 2 b     u         2     2
#> 3 c     v         3     NA
#> 4 c     w         NA     1
```

左側の表と一致しないものに入れ替えています。たとえば、a, b, c は、右側の行でも、t, u, v であるべきときに、違っている部分をチェックするようなときにも使えます。

```
anti_join(df_1, df_2, by = c("A", "B" = "F"))
#> # A tibble: 1 x 3
#>   A     B     C
#>   <chr> <chr> <dbl>
#> 1 c     v         3
```

anti_join とは逆に、一致しているもののみ取り出します。

```
semi_join(df_1, df_2, by = c("A", "B" = "F"))
#> # A tibble: 2 x 3
#>   A     B     C
#>   <chr> <chr> <dbl>
#> 1 a     t         1
#> 2 b     u         2
```

集合算では、同じ列について、intersect は共通な部分を取り出します。

```
intersect(df_1, df_3)
#> # A tibble: 1 x 3
#>   A     B     C
#>   <chr> <chr> <dbl>
#> 1 a     t         1
```

union は、和集合ですから、すべてを含むものを作成します。

```
union(df_1, df_3)
#> # A tibble: 5 x 3
#>   A     B     C
#>   <chr> <chr> <dbl>
#> 1 a     t         1
#> 2 b     u         2
#> 3 c     v         3
#> 4 C     u         3
#> 5 d     w         4
```

setdiff は差集合ですから、df_1 に入っていて、df_3 には入っていないものを取り出し

ます。

```
setdiff(df_1, df_3)
#> # A tibble: 2 x 3
#>   A     B     C
#>   <chr> <chr> <dbl>
#> 1 b     u     2
#> 2 c     v     3
```

第 21 章

数理モデル (Modeling)

21.1 はじめに

データサイエンスにおいて、数理モデルを考えるとどのようなことを意味するのでしょうか。簡単な表現は、データの（主要な）特徴の簡単な概要、またはそれを（いくつかの）数値で表現すること、と言ってもよいと思います。

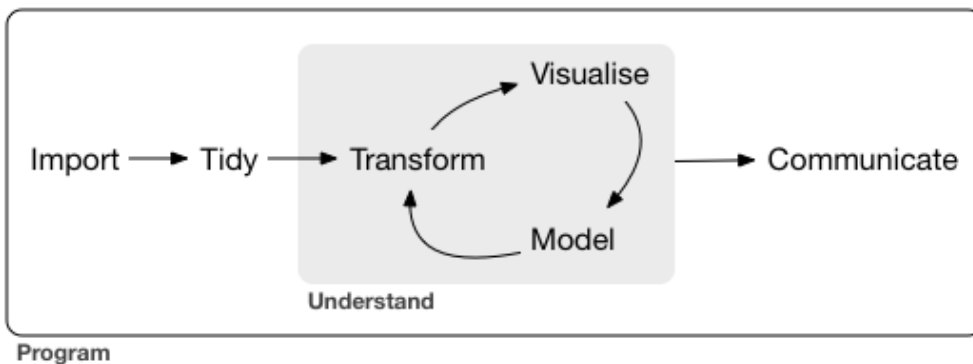


図21.1 image

その意味では、平均値や、中央値、四分位、最大値、最小値などの、代表値と言われるものも、そうですし、正規分布など、特定の分布に近い場合に、それを定める、平均値と、標準偏差の組みも、ひとつの数理モデルを与えと言えるでしょう。

ここでは、もう少し一般的な分布に関して使われる、数理モデルについて学びます。その中でも、基本的な、二変数に関する、線形モデルについてすこし、ていねいにみていきたいと思います。

簡単にいうと、 X と Y という変数があったときに、 X が増えると、 Y も増えるとか、減るとか、 X が増加すると、 Y も増加するとか、減少するとかいう傾向をみたり、それが、どの程度、直線的に、増加、減少しているかを、 $Y = b + m \cdot X$ と、直線の方程式で近似し（おおまかに表し）、それが、どの程度、実際のデータに近いかを表現する（あらかず数値をあたえる）というようなことです。

Y を X の関数で表す（ X を定めると、 Y を計算することができる）とも表現され、 X を

独立変数 (independent variable)、Y を従属変数 (dependent variable) と呼びます。もちろん、X と Y の役割を逆にすることも可能なので、便宜的な名前です。また、統計学では、Y を目的変数 (response variable)、X を説明変数 (explanatory variable) と呼ぶ呼び方もされます。ただ、これらは、X によって、Y が定まるという、因果関係を表すという考え方が背景にあるのですが、傾向や関係性を見るという観点からは、これらの名前を使うことを避けたほうがよいとも言えますので、ここでは、独立変数、従属変数、説明変数、目的変数という言い方は、できるだけ避けたいと思います。

R では、 b (Y 切片) は、intercept (切片) として表示され、 m (直線の傾き) は、(変数 X に関する) slope (傾き) として表示されます。

R を使って、モデルについて表示される、数値の意味も説明したいと思います。数式での表現も書きますが、式を正確に覚えたりするよりも、大まかな意味を理解することを心がけてください。

数理モデルは、非常に多様なだけでなく (主要な) 特徴は、分野ごとに必要度が異なることから、一般論として述べることには限界があります。H. ウィッカム (Hadley Wickham, et. al.) の R for Data Science の標準的な教科書でも、第一版には、基本的なことが含まれていましたが、第二版には、数理モデルは含まれていません。Tidy Modeling with R を参照するようにリンクがついています。

ここでは、オープンデータを用いたデータサイエンス、データ分析を学んでいます。基本的な線形モデルについて紹介しますが、そのモデルが適切かを判断しようとすると、t-検定や、F-検定を適用することが多いのですが、いずれも、正規分布を仮定しています。実際の社会、様々な国や、地域などを比較して検討するとき、そのデータが、正規分布をしているかどうかは、それほど簡単に判断できるわけではありません。また、これも、ある統計的な道具を使って、正規分布に近いことがある程度保証できたとしても、それに、線形モデルを適用して、どの程度適切な結果が得られるか、個人的には判断が難しいと思います。さらに、相関関係やその強さはわかっても、線形モデルなどから、因果関係を導くことはできません。個人的には、線形の関係がみてとれる、程度で、止めておくことが大切ではないかと考えています。安易に、t-検定や、F-検定を用いて、結論を導くのは、差し控えるべきで、ある結果を導くときには、ほかにも、さまざまな根拠をもつことがたいせつだと、個人的には大切だと考えます。

ということで、説明はしますが、自分は、統計がわからないから、データサイエンスはできないなどは、言ってほしくないと思っています。統計の深い知識がなくても、データの様々な特徴を見て取ることは、可能だと思います。

21.2 準備 (Setup)

```
library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
```



```

#> v forcats 1.0.0 v stringr 1.5.0
#> v ggplot2 3.4.3 v tibble 3.2.1
#> v lubridate 1.9.2 v tidyr 1.3.0
#> v purrr 1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag() masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(WDI)
library(readxl)
library(tidymodels)
#> -- Attaching packages ----- tidymodels 1.1.1 --
#> v broom 1.0.5 v rsample 1.2.0
#> v dials 1.2.0 v tune 1.1.2
#> v infer 1.0.5 v workflows 1.1.3
#> v modeldata 1.2.0 v workflowsets 1.0.1
#> v parsnip 1.1.1 v yardstick 1.2.0
#> v recipes 1.0.8
#> -- Conflicts ----- tidymodels_conflicts() --
#> x scales::discard() masks purrr::discard()
#> x dplyr::filter() masks stats::filter()
#> x recipes::fixed() masks stringr::fixed()
#> x dplyr::lag() masks stats::lag()
#> x yardstick::spec() masks readr::spec()
#> x recipes::step() masks stats::step()
#> * Search for functions across packages at https://www.tidymodels.org/find/

```

Tidymodels パッケージを使いますが、これは、Tidyverse と同様に、一つのパッケージではなく、パッケージ群を表します。また、ここでは、ほんの一部しか使いませんが、一応、後々のために、インストールをし、使えるように `library(tidymodels)` で、読み込んでおいてください。興味のあるかたのために、Tidymodels サイトへのリンクを付けておきます。

- tidymodels: <https://www.tidymodels.org>

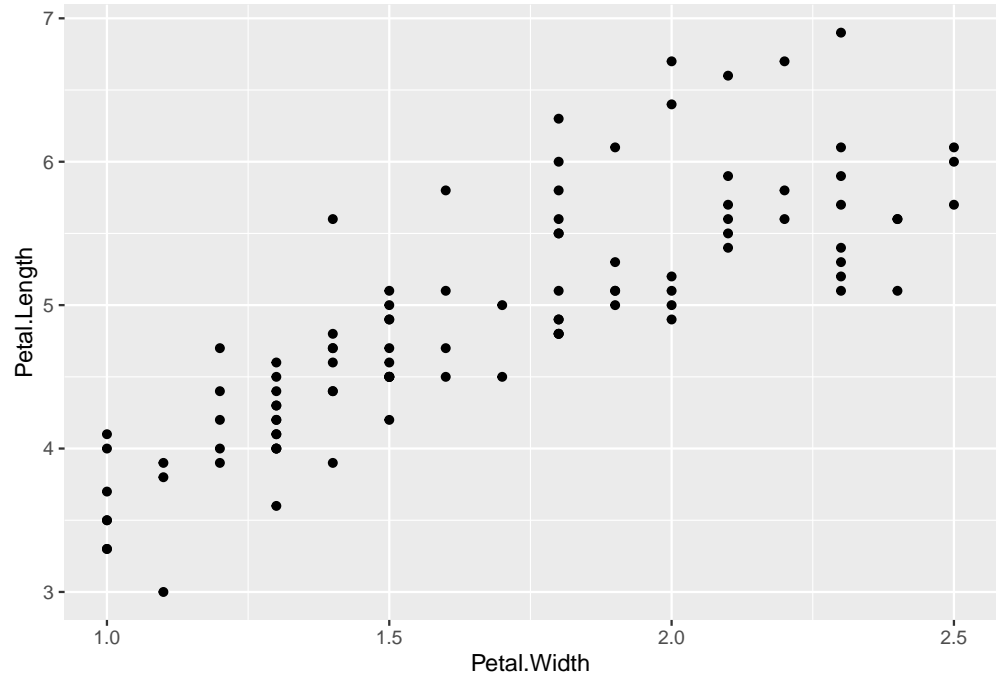
21.3 線形モデル入門

起動時に読み込まれる、`datasets` パッケージの、`iris` データを用いて、概要をみてみます。`iris` には、三種類の、あやめのデータが含まれていますが、ここでは、`virginica` と `versicolor`、すなわち、`setosa` 以外について扱います。

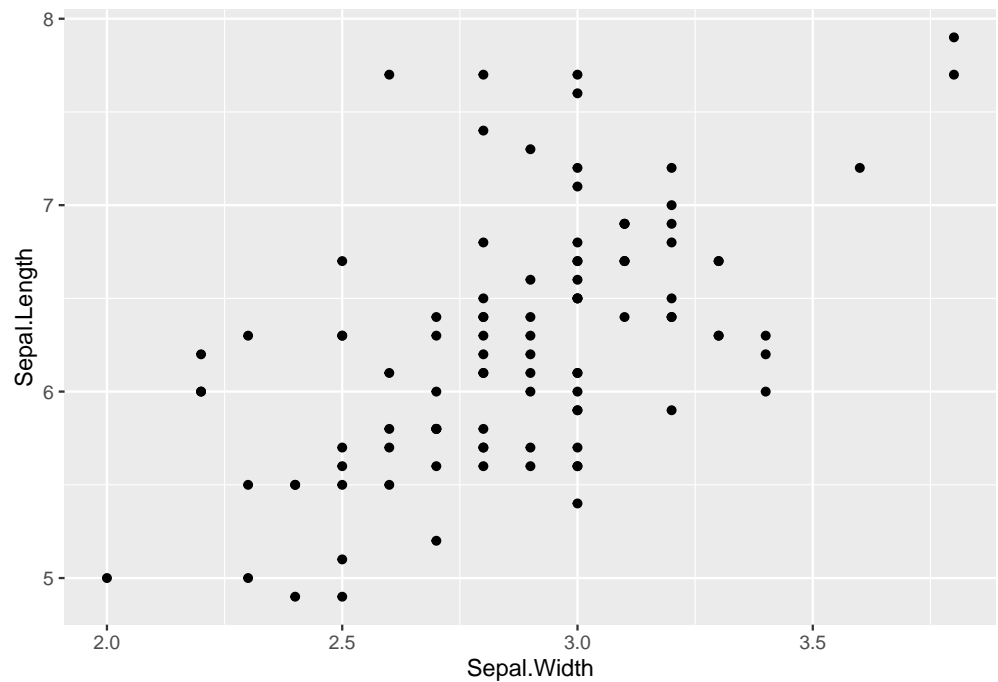
```
df_iris0 <- as_tibble(datasets::iris) %>% filter(Species != "setosa")
```

二つの散布図を見てみます。一つ目は、X 軸が、花弁幅（はなびらのはば）、Y 軸が、花弁長（はなびらの長さ）、二つ目は、X 軸が、萼幅（がくのはば）、Y 軸が、花弁長（がくの長さ）を表した散布図です。

```
df_iris0 |> ggplot(aes(Petal.Width, Petal.Length)) + geom_point()
```



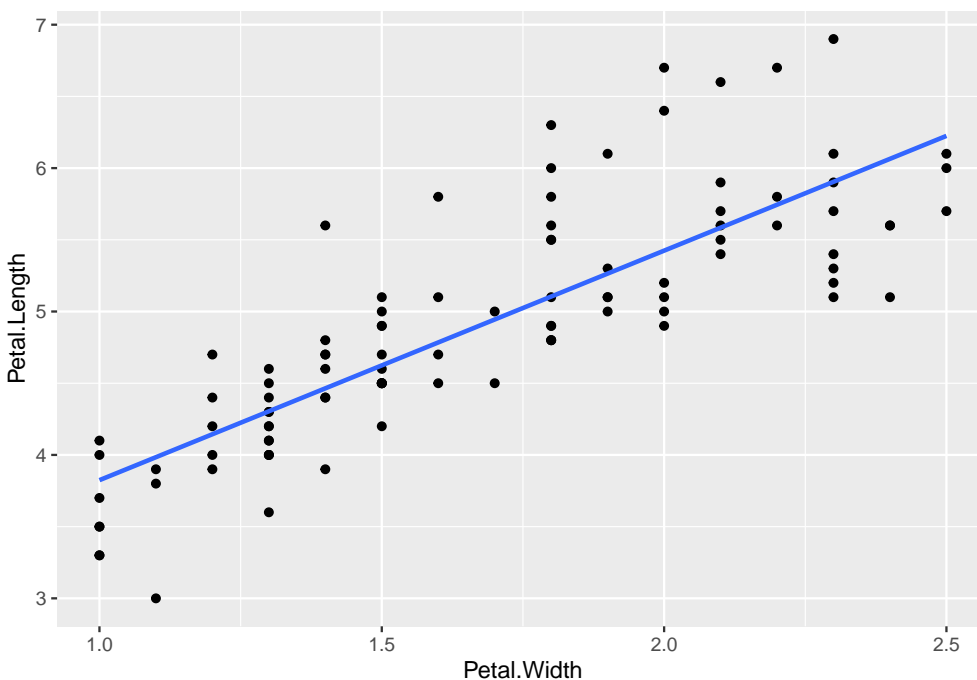
```
df_iris0 |> ggplot(aes(Sepal.Width, Sepal.Length)) + geom_point()
```



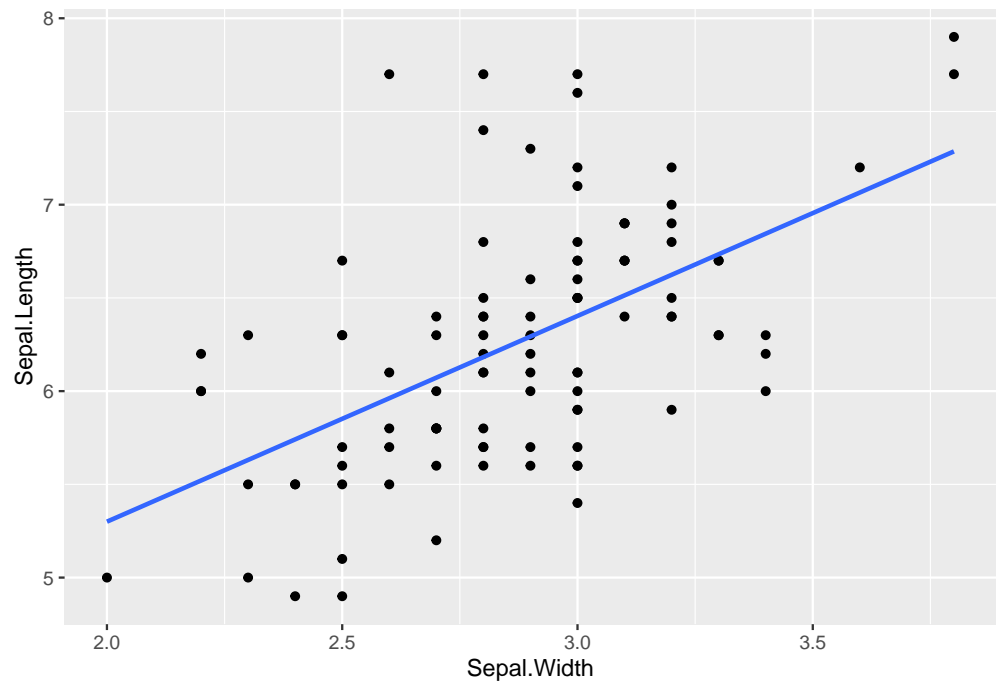
どちらも、幅（はなびらの幅や、がくの幅）が大きくなると、長さ（はなびらの長さや、がくの長さ）が大きくなる傾向があるように見えます。それは、なんとなく直感的にも自然ですよね。種類が同じ植物のはなびらや、がくの形は、だいたい同じ（相似形にちかい）と思われるからです。（直感はとてもたいせつです。直感とあわないことがあれば、立ち止まって考えて見るようにしてください。）しかし、はなびらの散布図は、それが明らかにみてとれるだけでなく、ある直線にそって（または直線的に）増加しているように見えます。がくの散布図では、その傾向が明らかではなく（ぼんやりとしていて）、直線にそって増加しているとまでは言えないように見えます。

すこし、違った、見方をする方もおられるかもしれません。定義を明確にすることで、合意ができるように、それを、数値的に表現するのが、線形モデルを使った表現です。まずは、それぞれに、増加傾向をあらわす直線を描いてみましょう。これは、ある一定の方法で描いたもので、歴史的背景から、回帰直線（regression line）と呼ばれますが、いまは、大体、そんなものだと考えてくださって構いません。

```
df_iris0 |> ggplot(aes(Petal.Width, Petal.Length)) + geom_point() + geom_smooth(method="lm",formula=y~x,
```



```
df_iris0 |> ggplot(aes(Sepal.Width, Sepal.Length)) + geom_point() + geom_smooth(method="lm",formula=y~x,
```



たしかに、この二つの散布図と、直線を見ると、それぞれ（はなびらや、がくの）幅が増加すると長さも増加する傾向にあるようです。そこで、今度は、直線に近い、増加傾向にあると仮定して、適切な直線の方程式を求めてみます。それは、次のようにして求めることができます。

21.3.1 線形モデルの例 (Linear Models by Examples)

21.3.1.1 Linear Model: Petal.Length ~ Petal.Width

はなびらの長さを、はなびらの幅を変数とする、直線の方程式で近似してみます。

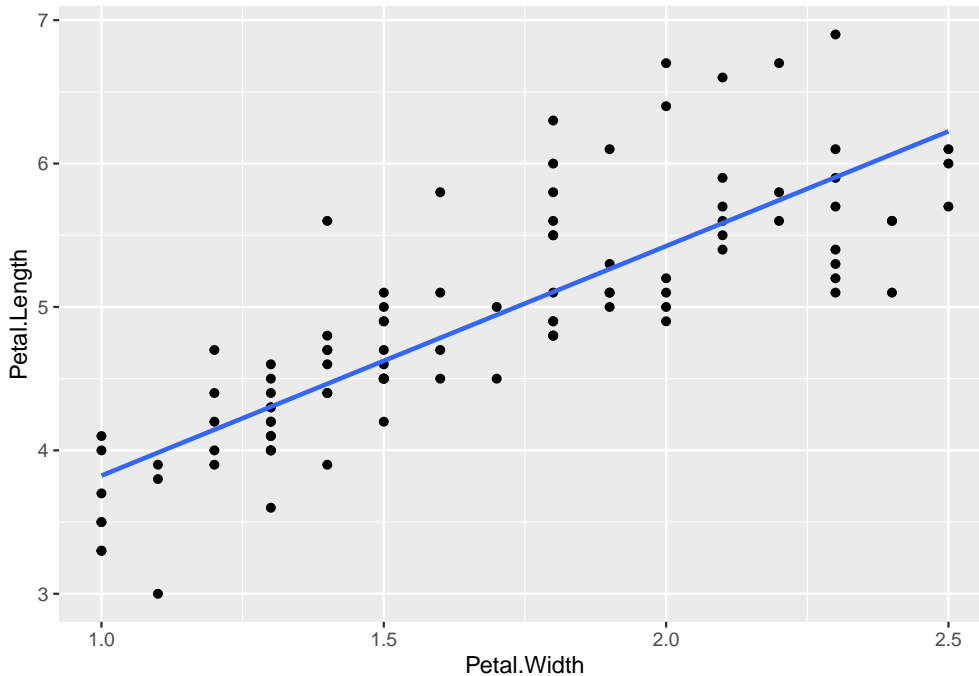
```
df_iris0 |> lm(Petal.Length ~ Petal.Width, data = _)
#>
#> Call:
#> lm(formula = Petal.Length ~ Petal.Width, data = df_iris0)
#>
#> Coefficients:
#> (Intercept) Petal.Width
#>      2.224      1.600
```

Call の部分は、「このような (X を Petal.Width として、Y を Petal.Length の一次式、すなわち、 $Y = b + m \cdot X$ と表わそうと) 線形モデルで考えると」といった、意味です。

Coefficients は、係数の意味です。次に、(Intercept) が、2.224、Petal.Width が 1.600 とありますから、1.600 が一次式で表したときの Petal.Width の係数で、式で書くと次のようになります。

Formula: $\text{Petal.Length} = 2.224 + 1.600 \cdot \text{Petal.Width}$

Petal.Width の範囲が限られていますから、Y 切片は確認できませんが、だいたい見当が付きそうですし、傾きは、X が 1 大きくなると、Y がどれくらい大きくなるかという数ですから、だいたい、見当が付きそうですね。



21.3.1.2 Linear Model: Sepal.Length ~ Sepal.Width

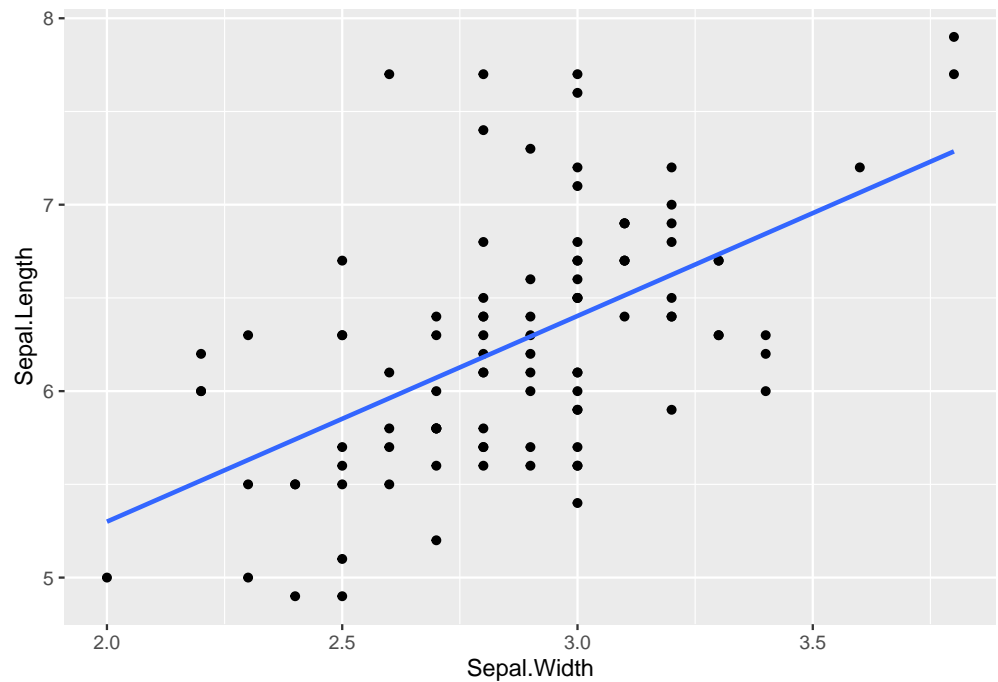
がくの長さを、がくの幅を変数とする、直線の方程式で近似してみます。

```
df_iris0 |> lm(Sepal.Length ~ Sepal.Width, data = _)
#>
#> Call:
#> lm(formula = Sepal.Length ~ Sepal.Width, data = df_iris0)
#>
#> Coefficients:
#> (Intercept) Sepal.Width
#>      3.093      1.103
```

式は、次のようになっていることがわかります。

Formula: $\text{Sepal.Length} = 3.093 + 1.103 \cdot \text{Sepal.Width}$

下のグラフから、だいたいの、Y 切片や、傾きを読み取って、上の式と比較してください。



21.3.1.3 決定係数 (R squared)

日本語では、決定係数と呼ばれることが多いようですが、R squared の意味を簡単に説明します。

まず、決定係数 (R squared) は、0 と 1 の間の値を取ります。モデルの当てはまり具合を表す数とも表現されます。

The coefficient of determination R^2 represents the strength of fit of the model.

このモデル (現在の場合は X の関数として、Y の値をおおまかに表現する線形モデル) によって、どのくらい、実際の値が決まるかをあらわす一つの指標です。この値が、1 のときは、ピッタリ合致していることを意味し、0 のときは、X の値に依存しない、平均値と、変わらないという意味です。たとえば、はなびらモデルでは、下のようになりますから、68%程度決定する、または、適合するという言い方をします。がくモデルの場合には、31%程度決定する、または、適合するという言い方をします。はなびらモデルのほうが、がくモデルよりもより直線的ということを表す数ということになります。

(Multiple) R Squared: a value between 0 and 1, the model's strength. It is a measurement of the model quality. If the value is close to 1, the model quality is high. If it is close to 0, the model quality is low.

Petal.Length ~ Petal.Width: R squared = 0.6779 - 68%

```
df_iris0 |> lm(Petal.Length ~ Petal.Width, data = _) |> summary()
#>
#> Call:
#> lm(formula = Petal.Length ~ Petal.Width, data = df_iris0)
```

```

#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -0.9842 -0.3043 -0.1043  0.2407  1.2755
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   2.2240     0.1926   11.55 <2e-16 ***
#> Petal.Width   1.6003     0.1114   14.36 <2e-16 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.4709 on 98 degrees of freedom
#> Multiple R-squared:  0.6779, Adjusted R-squared:  0.6746
#> F-statistic: 206.3 on 1 and 98 DF,  p-value: < 2.2e-16

```

Sepal.Length ~ Sepal.Width: R squared = 0.3068 - 31%

```

df_iris0 |> lm(Sepal.Length ~ Sepal.Width, data =_) |> summary()
#>
#> Call:
#> lm(formula = Sepal.Length ~ Sepal.Width, data = df_iris0)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -1.0032 -0.3877 -0.0774  0.3200  1.7381
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept)   3.0934     0.4844   6.387 5.70e-09 ***
#> Sepal.Width   1.1033     0.1675   6.585 2.27e-09 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.5547 on 98 degrees of freedom
#> Multiple R-squared:  0.3068, Adjusted R-squared:  0.2997
#> F-statistic: 43.36 on 1 and 98 DF,  p-value: 2.27e-09

```

上のように、summary() 関数をつかうと、Multiple R-squared や、Adjusted R-squared と R squared 値は、出てきますが、現在の場合は、Multiple R-squared が、決定係数だ

と考えてください。

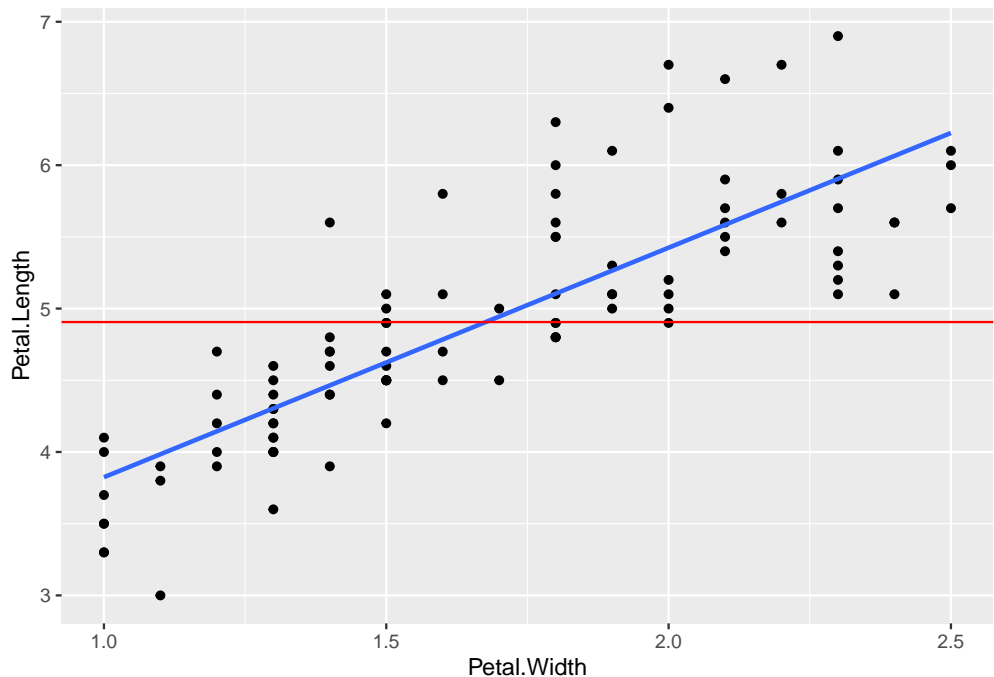
いまは、 Y を、 X の関数として表しましたが、 X ではなく、 X_1 、 X_2 、などとたくさんの変数を使って、表し、全体のデータ（観測値）の個数が、あまり多くないときは、Multiple R-squared よりも、Adjusted R-squared のほうが、より適切になってくるばあいがあるので、このように、2つの数が書かれています。

また、決定係数 (R squared) は、線形モデル以外にも使えますので、モデルの適合性を表す、たいせつな指標のひとつと理解しておいてください。

summary() には、他にもたくさんの、値が出ています。あとで、それぞれについて、簡単に見ていきたいと思います。

決定係数が 1 に近いと良い（強い）モデルで、0 に近いと悪い（弱い）モデルということになりますが、どの程度ならば、良いモデルかは一概に言えません。分野にもよりますが、0.3 より小さければ、あまりよいモデルだとは言えず、0.7 より大きければ、よいモデルのひとつだと考えても良いのではないかと思います。

上の 2つのモデルは、微妙ですが、花びらモデルのほうが、良い（より適合した）モデルだと表現できます。



21.3.2 線形モデルの基礎 (Linear Model Basics) : $y \sim x$

R では、線形モデルを次のように定めます。 $y \sim x$ は、 x の関数で、 y を表すという意味です。パイプを使うときは、その下の式のようになります。パイプは原則として、最初の変数として、一つ前の出力が読み込まれますが、`lm(arg1, arg2)` の場合には、`arg2` 2つ目の変数が、データですから、それを表すために、`data = _` (または、`%>%` の場合には、そこに、ピリオッド (半角)) を書き、そのことを示します。場所指定の `'_'` (または `'?'`)

(place holder) と言います。

```
lm(y~x, data)
```

```
data |> lm(y~x, data=_) # data = をつけないとエラーになります
```

```
data %>% lm(y~x, .)
```

y 切片や、傾きが得られます。(y-intercept, and slope: rate of increase or decrease)

モデルのより詳細な情報は、`summary()` によって得ることができます。パイプを使う場合には、その次のコードとなります。

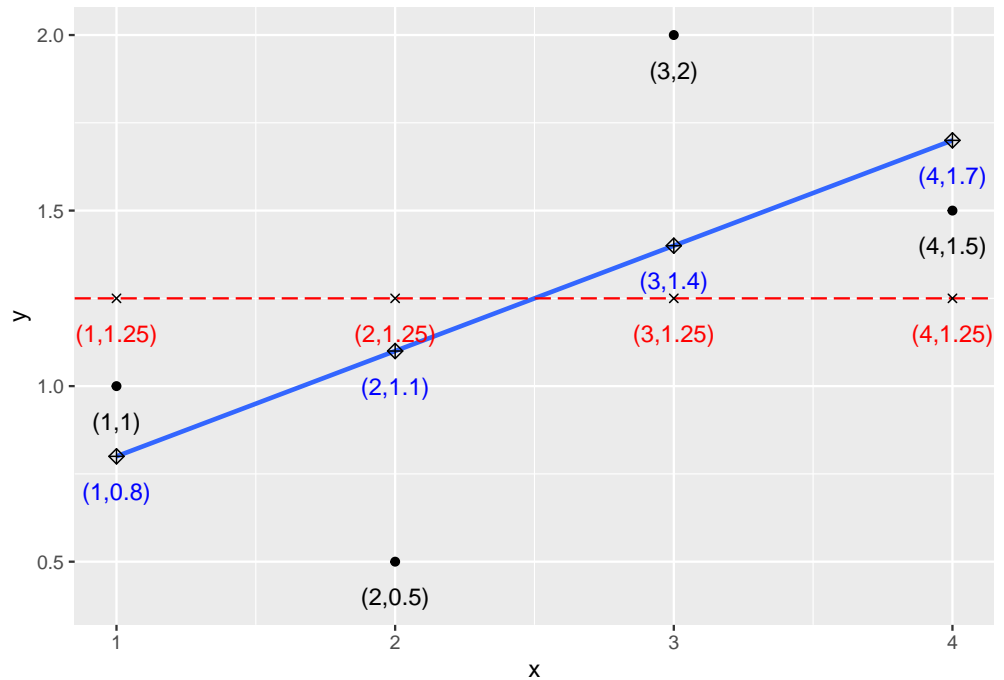
```
summary(lm(y~x, data))
```

```
data |> lm(y~x, .) |> summary()
```

具体的な例から、それぞれの数値がなにを表しているのか見ていきましょう。

```
df1 <- data.frame(x = c(1,2,3,4), y = c(1,0.5,2, 1.5))
ybar <- mean(df1$y)
mod1 <- lm(y~x, df1)
augment(mod1) |> ggplot() + geom_point(aes(x,y)) + geom_smooth(aes(x,y), formula = y~x, method = "lm", s
```

- $(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$: Data points - データの4つの点をあらわす黒点
- \bar{y} : mean of $y = (y_1 + y_2 + y_3 + y_4)/4$. (相加) 平均値を表す赤線
- \hat{y}_i : prediction at x_i , 線形モデルの一次式で与えられる値
 - $(x_1, \hat{y}_1), (x_2, \hat{y}_2), (x_3, \hat{y}_3), (x_4, \hat{y}_4)$ are on the regression line. 回帰直線上で表された点
- $y_1 - \hat{y}_1, y_2 - \hat{y}_2, y_3 - \hat{y}_3, y_4 - \hat{y}_4$ are called residues. 実際の値から、モデルでの予測値を引いたもので、残差と呼ばれます



線形モデルの一次式で与えられる値 (fitted values) は次のようにしても表示できます。

```
mod1$fitted.values
#> 1 2 3 4
#> 0.8 1.1 1.4 1.7
```

21.3.2.1 決定係数 (R Squared)

SS_{tot} は、実際の値から平均値を引いたものを、二乗して足したものです。二乗してあるので、かならず正 (正確には非負) になります。

SS_{res} は、実際の値から (この線形) モデルによって予測したあたいを引いたものを、二乗して足したものです。二乗してあるので、かならず正 (正確には非負) になります。

R^2 は、一番下の式で定義されます。完全に予測が実際の値と一致していると、 SS_{res} は、0 になりますから、その場合は、商の分子が 0 で、 R^2 の値は、1 になります。

線形モデルの、直線の方程式は、直線の中で、 SS_{res} が一番小さくなるものを (最小二乗法という方法で) 選んであり、平均を通る直線から計算したものよりは、小さくなるので、商の部分は、1 以下、すなわち、 R^2 の値は、0 以上 1 以下となります。

$$SS_{tot} = (1 - 1.25)^2 + (0.5 - 1.25)^2 + (2 - 1.25)^2 + (1.5 - 1.25)^2 = 1.25$$

$$SS_{res} = (1 - 0.8)^2 + (0.5 - 1.1)^2 + (2 - 1.4)^2 + (1.5 - 1.7)^2 = 0.8$$

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{0.8}{1.25} = 0.36.$$

実際に、mod1 の `summary()` から値を取り出す方法をいくつか書いておきます。glance

は、`tidymodel` に含まれる、パッケージ `broom` の関数で、モデルの概要 (`summary`) を、表 (この場合は、`tidyverse` の標準の `tibble`) として出力するものです。

```
summary(mod1)$r.squared
```

```
#> [1] 0.36
```

```
mod1 |> summary() |> glance()
```

```
#> # A tibble: 1 x 8
```

```
#>   r.squared adj.r.squared sigma statistic p.value    df
```

```
#>   <dbl>         <dbl> <dbl>    <dbl>  <dbl> <dbl>
```

```
#> 1     0.36         0.0400 0.632    1.12   0.4    1
```

```
#> # i 2 more variables: df.residual <int>, nobs <dbl>
```

```
mod1 |> glance() |> pull(r.squared)
```

```
#> [1] 0.36
```

```
mod1 |> glance() |> select(`R Squared` = r.squared)
```

```
#> # A tibble: 1 x 1
```

```
#>   `R Squared`
```

```
#>   <dbl>
```

```
#> 1     0.36
```

21.3.3 モデル概要

一般的な場合に、それぞれの数値を表す数式 (Mathematical Formula) と簡単な説明を記します。

- $x = c(x_1, x_2, \dots, x_n)$: X 変数 (the X (independent) variable) 例 `Petal.Width`
- $y = c(y_1, y_2, \dots, y_n)$: Y 変数 (the Y (dependent) variable) 例 `Petal.Length`
- $\text{pred} = c(\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n)$: モデル予測値 (the predicted values by a (linear regression) model) 例 $\hat{y}_i = b + m \cdot x_i$

例は、はなびらモデルを使って説明します。

```
df_iris0 |> lm(Petal.Length ~ Petal.Width, data = _) %>% summary()
```

```
#>
```

```
#> Call:
```

```
#> lm(formula = Petal.Length ~ Petal.Width, data = df_iris0)
```

```
#>
```

```
#> Residuals:
```

```
#>   Min      1Q  Median      3Q      Max
```

```
#> -0.9842 -0.3043 -0.1043  0.2407  1.2755
```

```
#>
```

```
#> Coefficients:
```

```
#>           Estimate Std. Error t value Pr(>|t|)
```

```
#> (Intercept)  2.2240    0.1926   11.55 <2e-16 ***
#> Petal.Width  1.6003    0.1114   14.36 <2e-16 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.4709 on 98 degrees of freedom
#> Multiple R-squared:  0.6779, Adjusted R-squared:  0.6746
#> F-statistic: 206.3 on 1 and 98 DF,  p-value: < 2.2e-16
```

21.3.3.1 Call (適用されたモデル)

使われたモデルの定義式 (The function and the parameters used to create the model)

21.3.3.2 Residuals (残差)

残差は、実際の値から、予測値を差し引いた値 (Difference between what the model predicted and the actual value of y): $e_i = y_i - \hat{y}_i$

その、分布の、最小値、第一四分位、中央値、第三四分位、最大値が表示されています。次のようにしても、表示できます。

Tidymodels に含まれる、`broom` パッケージの `augment()` 関数を使うと、予測値 predicted values (pred) または、fitted values は、`.fitted` 列に、残差 residues は、`.resid` に追加されます。(他の列については、さしあたって不要ですが、Helpなどを参照してください。)

```
df_iris0 |> lm(Petal.Length ~ Petal.Width, data = _) |> augment()
#> # A tibble: 100 x 8
#>   Petal.Length Petal.Width .fitted .resid .hat .sigma
#>   <dbl>         <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1         4.7         1.4   4.46  0.236  0.0143  0.473
#> 2         4.5         1.5   4.62 -0.124  0.0117  0.473
#> 3         4.9         1.5   4.62  0.276  0.0117  0.473
#> 4          4          1.3   4.30 -0.304  0.0179  0.472
#> 5         4.6         1.5   4.62 -0.0244 0.0117  0.473
#> 6         4.5         1.3   4.30  0.196  0.0179  0.473
#> 7         4.7         1.6   4.78 -0.0844 0.0103  0.473
#> 8         3.3          1     3.82 -0.524  0.0356  0.470
#> 9         4.6         1.3   4.30  0.296  0.0179  0.472
#> 10        3.9         1.4   4.46 -0.564  0.0143  0.470
#> # i 90 more rows
#> # i 2 more variables: .cooksd <dbl>, .std.resid <dbl>
```

したがってそれを使えば、Residuals の部分の値は、次のようにして得られます。

```
df_iris0 |> lm(Petal.Length ~ Petal.Width, data = _) |> augment() |>
  pull(.resid) |> summary()
#>   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#> -0.9843 -0.3043 -0.1043  0.0000  0.2407  1.2755
```

21.3.3.3 数式による定義

$$x \text{ の平均: } mean(x) = \frac{1}{n} \sum_i x_i$$

$$y \text{ の平均: } mean(y) = \frac{1}{n} \sum_i y_i$$

$$x \text{ の分散: } var(x) = \frac{1}{n} \sum_i (x_i - mean(x))^2$$

$$y \text{ の分散: } var(y) = \frac{1}{n} \sum_i (y_i - mean(y))^2$$

$$x \text{ と } y \text{ の共分散: } cov(x) = \frac{1}{n} \sum_i (x_i - mean(x))(y_i - mean(y))$$

$$x \text{ と } y \text{ の相関係数: } cor(x) = \frac{cov(x, y)}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$\text{回帰直線の係数: } slope = \frac{cov(x, y)}{var(x)} = \frac{cor(x, y)\sqrt{var(y)}}{\sqrt{var(x)}}$$

$$\text{全平方和 total sum of squares: } SS_{tot} = \sum_i (y_i - mean(y))^2$$

$$\begin{aligned} \text{残差平方和 residual sum of squares: } SS_{res} &= \sum_i (y_i - \text{pred}_i)^2 \\ &= \sum_i (y_i - \hat{y}_i)^2 \end{aligned}$$

$$\text{決定係数 R squared: } R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = cor(x, y)^2$$

21.3.3.4 Adjusted R Squared (自由度修正済み決定係数)

$$\text{Adjusted } R^2 = 1 - \frac{(1 - R^2)(n - 1)}{n - k - 1}$$

n : 行数 (観測値の数) number of observations, the number of rows

k : number of variables used for prediction

21.3.3.5 Coefficients (係数)

Tidymodels に含まれる、broom パッケージの tidy() 関数を使うと、係数の部分を表として取り出すことができます。

```
df_iris0 |> lm(Petal.Length ~ Petal.Width, data = _) |> tidy()
#> # A tibble: 2 x 5
#>   term          estimate std.error statistic p.value
#>   <chr>          <dbl>    <dbl>    <dbl>    <dbl>
#> 1 (Intercept)    2.22     0.193    11.5 5.54e-20
#> 2 Petal.Width    1.60     0.111    14.4 7.57e-26
```

ここに現れる、statistics は t-検定の、値で、p.value が最初に定めた値、たとえば、0.05 よりも、小さければ、Intercept や、Petal.Width の係数が、0 である確率は非常に小さいといえます。

21.3.3.6 Other Model Summaries (その他のモデル概要)

Tidymodels に含まれる、broom パッケージの glance() 関数を使うと、残りの値が表示できます。

```
df_iris0 |> lm(Petal.Length ~ Petal.Width, data = _) |> glance()
#> # A tibble: 1 x 12
#>   r.squared adj.r.squared sigma statistic p.value df
#>   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <dbl>
#> 1  0.678      0.675 0.471    206. 7.57e-26  1
#> # i 6 more variables: logLik <dbl>, AIC <dbl>, BIC <dbl>,
#> #   deviance <dbl>, df.residual <int>, nobs <int>
```

r.squared は、R squared (決定係数)、adj.r.squared は、Adjusted R Squared (自由度修正済み決定係数)、sigma は、Residual Standard Error (残差標準誤差)、statistic は、f 検定の F value (F 値)、p.value は、F 分布の p 値です。

最初の 2 つはすでに説明しました。

$$\text{statistic} = \frac{SS_{tot} - SS_{res}}{SS_{res}} \cdot \frac{n - (k + 1)}{k}$$

p 値が、定めた値、例えば、0.05 より小さいと、2 つの変数が、関係しない確率が非常に小さいということになります。

Correlation Constant (相関係数)

式に現れるように、 $R^2 = \text{cor}(x, y)^2$ でしたから、相関係数が分かれば、この場合の決定係数はわかることになります。特に、相関係数の大きな変数の対を見つければ、それから計算する、線形モデルは、強いモデルだということになります。

```
df_iris0 |> select(1:4) |> cor()
#>           Sepal.Length Sepal.Width Petal.Length
#> Sepal.Length  1.0000000  0.5538548  0.8284787
```

```
#> Sepal.Width      0.5538548      1.0000000      0.5198023
#> Petal.Length      0.8284787      0.5198023      1.0000000
#> Petal.Width       0.5937094      0.5662025      0.8233476
#>
#>           Petal.Width
#> Sepal.Length      0.5937094
#> Sepal.Width       0.5662025
#> Petal.Length      0.8233476
#> Petal.Width       1.0000000
```

その自乗を計算することもできます。

この例では、Sepal.Length と、Petal.Length の相関が最も強いことがわかります。

select(1:4) で、最初の四列を取り出していますが、iris のデータは、五つの変数（五列）で五番目は、Species（種類）でしたから、Species の列以外という意味です。すこで、下では、select(-5) としています。どちらでも同じです。Species は、文字列で種類が書かれていましたから、相関係数を求めようとすると、エラーになります。

```
cormat <- df_iris0 |> select(1:4) |> cor()
cormat*cormat
#>
#>           Sepal.Length Sepal.Width Petal.Length
#> Sepal.Length      1.0000000      0.3067552      0.6863769
#> Sepal.Width       0.3067552      1.0000000      0.2701944
#> Petal.Length      0.6863769      0.2701944      1.0000000
#> Petal.Width       0.3524909      0.3205853      0.6779013
#>
#>           Petal.Width
#> Sepal.Length      0.3524909
#> Sepal.Width       0.3205853
#> Petal.Length      0.6779013
#> Petal.Width       1.0000000
```

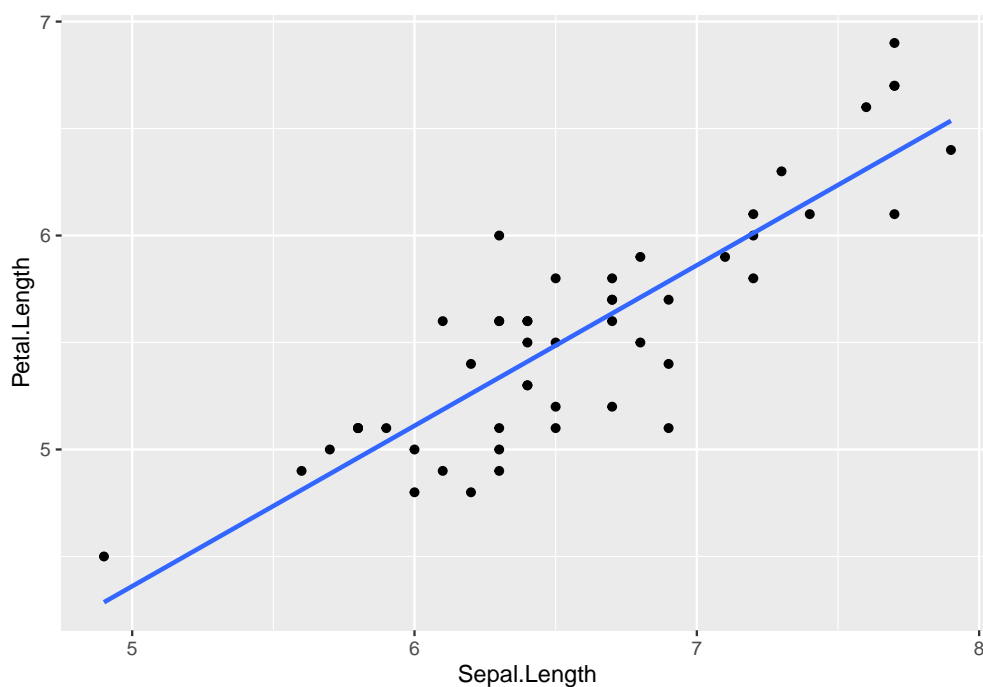
```
df_iris <- datasets::iris
```

```
as_tibble(df_iris) |> filter(Species == "setosa") |> select(-5) %>% cor()
#>
#>           Sepal.Length Sepal.Width Petal.Length
#> Sepal.Length      1.0000000      0.7425467      0.2671758
#> Sepal.Width       0.7425467      1.0000000      0.1777000
#> Petal.Length      0.2671758      0.1777000      1.0000000
#> Petal.Width       0.2780984      0.2327520      0.3316300
#>
#>           Petal.Width
#> Sepal.Length      0.2780984
#> Sepal.Width       0.2327520
#> Petal.Length      0.3316300
#> Petal.Width       1.0000000
```

```
as_tibble(df_iris) |> filter(Species == "virginica") |> select(-5) |> cor()
#>           Sepal.Length Sepal.Width Petal.Length
#> Sepal.Length  1.0000000  0.4572278  0.8642247
#> Sepal.Width   0.4572278  1.0000000  0.4010446
#> Petal.Length  0.8642247  0.4010446  1.0000000
#> Petal.Width   0.2811077  0.5377280  0.3221082
#>           Petal.Width
#> Sepal.Length  0.2811077
#> Sepal.Width   0.5377280
#> Petal.Length  0.3221082
#> Petal.Width   1.0000000
```

```
as_tibble(df_iris) |> filter(Species == "versicolor") |> select(-5) |> cor()
#>           Sepal.Length Sepal.Width Petal.Length
#> Sepal.Length  1.0000000  0.5259107  0.7540490
#> Sepal.Width   0.5259107  1.0000000  0.5605221
#> Petal.Length  0.7540490  0.5605221  1.0000000
#> Petal.Width   0.5464611  0.6639987  0.7866681
#>           Petal.Width
#> Sepal.Length  0.5464611
#> Sepal.Width   0.6639987
#> Petal.Length  0.7866681
#> Petal.Width   1.0000000
```

```
as_tibble(df_iris) |> filter(Species == "virginica") |> ggplot(aes(Sepal.Length,
```



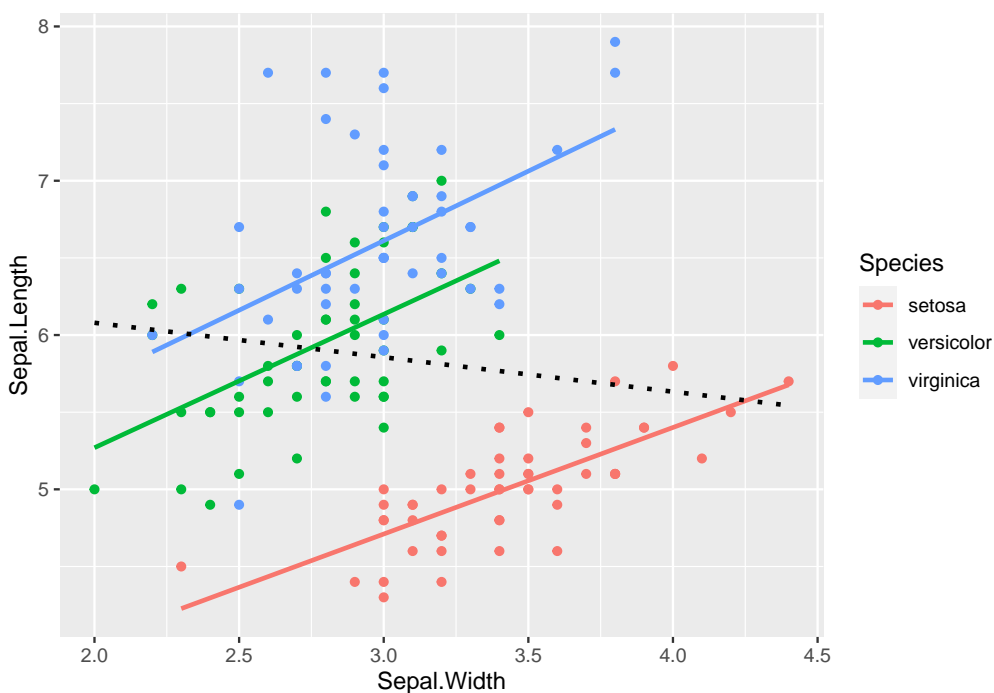

```
as_tibble(df_iris) |> filter(Species == "virginica") |> lm(Petal.Length ~ Sepal.Length, data = _) |> gla
#> [1] 0.8642247
```

2つの変数の場合には、このように、相関係数を見ても、大切です。Correlations of the data suggest the possible strength of linear model $y \sim x$.

```
df_iris |> select(-5) |> cor()
#>           Sepal.Length Sepal.Width Petal.Length
#> Sepal.Length  1.0000000 -0.1175698  0.8717538
#> Sepal.Width  -0.1175698  1.0000000 -0.4284401
#> Petal.Length  0.8717538 -0.4284401  1.0000000
#> Petal.Width   0.8179411 -0.3661259  0.9628654
#>           Petal.Width
#> Sepal.Length  0.8179411
#> Sepal.Width  -0.3661259
#> Petal.Length  0.9628654
#> Petal.Width  1.0000000
```

最後に、もう一つ、iris の例を描いておきます。

```
df_iris |> ggplot(aes(Sepal.Width, Sepal.Length)) +
  geom_point(aes(color = Species)) +
  geom_smooth(aes(color = Species), formula = "y~x", method = "lm", se = FALSE) +
  geom_smooth(formula = "y~x", method = "lm", se = FALSE, color = "black", linetype = "dotted")
```



点線は、全体の、がくモデルについての、回帰直線です。どのようなことがわかりますか。

- 三つの種別ごとで見ると、回帰直線は、右肩上がり、すなわち、萼（がく）幅が大きくなると、萼長が大きくなっている。
- 全体で見ると、回帰直線は、右肩下がり、すなわち、萼（がく）幅が大きくなると、萼長が小さくなっている。

最初に、同じ種類のあやめ (iris) では、おそらく、萼の形などは相似（似た形をしている）だろうから、「萼（がく）幅が大きくなると、萼長が大きくなっていると予想される」と書きました。しかし、上のグラフからみると、これら三種類のあやめ全体ではそうっておらず、三種類個別にみると、予想のようになっていることがわかります。

この場合、Species が交絡因子 (confounder) になっていると言います。

21.3.3.7 ChatGPT への質問と回答

Q. 交絡因子、confounder について、説明してください。

iris の data で、Sepal.Width と Sepal.Length でみると、負の相関になっている、または、回帰直線の傾きが負になっているが、各 Species ごとにみると、傾きが正になっている。この場合も、Species が交絡因子になっていると言えますか。

ChatGPT (2023.9.19)

交絡因子 (confounder) は、2つ以上の変数間の関係を誤解させる要因であり、研究結果に偏りをもたらす可能性がある変数です。交絡因子は、主な関心事（たとえば、Sepal.Width と Sepal.Length の関係）と関係がある可能性があり、その関係を調べる際に考慮する必要があります。

Iris データセットで Sepal.Width と Sepal.Length の関係を見た場合、負の相関が観察されるか、回帰直線の傾きが負になるかもしれません。しかし、各 Species ごとに見た場合、傾きが正になることが示唆されています。

この場合、Species が交絡因子である可能性があります。なぜなら、Species は Sepal.Width と Sepal.Length の間の関係に影響を与えている可能性があります。例えば、異なる種のアイリスの場合、Sepal.Width と Sepal.Length の間の関係が異なるかもしれません。したがって、交絡因子として Species を考慮しないと、Sepal.Width と Sepal.Length の間の関係を適切に解釈することができません。

交絡因子を適切に扱うためには、統計的な手法や分析手法を用いて、交絡因子を制御する必要があります。例えば、多変量解析や回帰分析において、Species を説明変数としてモデルに含めることができます。そうすることで、Species の効果を考慮した Sepal.Width と Sepal.Length の関係をより正確に評価できます。

したがって、交絡因子として Species を考慮することは重要です。それにより、変数間の関係をより正確に理解し、結果の解釈に偏りが生じないようにすることができます。

21.4 世界開発指標 (WDI) からの例

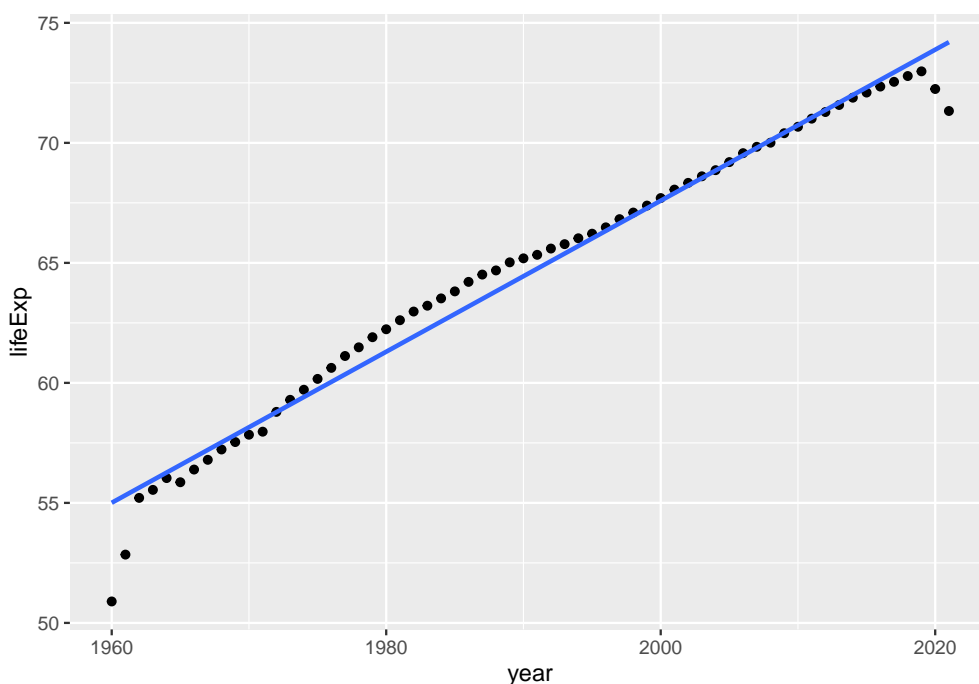
21.4.1 出生時の平均寿命 (life expectancy at birth)

- SP.DYN.LE00.IN: Life expectancy at birth, total (years)

```
wdi_lifeExp <- WDI(indicator = c(lifeExp = "SP.DYN.LE00.IN"))
```

```
#> Rows: 16758 Columns: 5
#> -- Column specification -----
#> Delimiter: ","
#> chr (3): country, iso2c, iso3c
#> dbl (2): year, lifeExp
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
wdi_lifeExp |> filter(country == "World") |> drop_na(lifeExp) |>
  ggplot(aes(year, lifeExp)) + geom_point() +
  geom_smooth(method = "lm", se = FALSE)
#> `geom_smooth()` using formula = 'y ~ x'
```



```
wdi_lifeExp |> lm(lifeExp ~ year, data = _) |> summary()
#>
#> Call:
#> lm(formula = lifeExp ~ year, data = wdi_lifeExp)
#>
#> Residuals:
#>     Min       1Q   Median       3Q      Max
#> -51.173  -7.150   1.741   7.794  18.782
```

```
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -5.383e+02  8.583e+00  -62.71  <2e-16 ***
#> year         3.027e-01  4.312e-03   70.20  <2e-16 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 9.707 on 15864 degrees of freedom
#> (892 observations deleted due to missingness)
#> Multiple R-squared:  0.237, Adjusted R-squared:  0.237
#> F-statistic: 4928 on 1 and 15864 DF, p-value: < 2.2e-16
```

$$lifeExp \sim -538.3 + 0.3027 \cdot year$$

世界の平均寿命は、毎年、ほぼ、0.3 年ほど増加しているという傾向にあるということのようです。ただ、適合度は、0.237 ですから、あまり良くはありません。これを使って、将来について予測するのは、適切ではないかもしれません。

Each year, life expectancy at birth increases approximately 0.3 years. R-squared of this model is 0.237, and the model explains 24%.

しかしグラフをみると、1960 年ごろと、1920 年以降は、ちょっと違った動きをしています。これを、除くと、ほぼ一直線上にあるように見えます。年を 1962 年から、2019 年までに、制限してみしてみましょう。

```
wdi_lifeExp %>% filter(country == "World", year >= 1962, year <= 2019) %>% drop_na()
#>
#> Call:
#> lm(formula = lifeExp ~ year, data = .)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -1.00776 -0.29546 -0.04527  0.37540  0.82117
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -5.537e+02  7.841e+00  -70.62  <2e-16 ***
#> year         3.107e-01  3.939e-03   78.89  <2e-16 ***
#> ---
#> Signif. codes:
```

```
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 0.5022 on 56 degrees of freedom
#> Multiple R-squared: 0.9911, Adjusted R-squared: 0.9909
#> F-statistic: 6224 on 1 and 56 DF, p-value: < 2.2e-16
```

今度は、決定係数が 0.9911 になりました。グラフをみた直感とあっていますね。ですから、1962 年から、2019 年までの間は、世界の出生時平均寿命は、毎年、0.31 年ほどずつ増加していると結論することができそうです。

実際には、調査対象の国がどのように変化しているかなど、他の要因も調べる必要があると思います。

また、なぜ、1960 年から、1962 年は、大きく変化しているのか、2020 年で減少している理由はなんだろうか。コロナウイルス感染症のせいだろうかなど、考えることが広がっていくかもしれません。次々と新しい問いが出てくること。それが素晴らしいことです。ほんの少しわかったことによって、扉が開いていく。そのような経験をたくさんしていただきたいと思います。

21.4.2 BRICS の出生時

今度は、BRICS についてみてみましょう。最初は、BRIC または、BRICs と、4 カ国で、ブラジル、ロシア、インド、中国を指していましたが、南アフリカが急成長したこともあり、南アフリカを加えて、現在は、BRICS と言っているようです。2023 年の首脳会議で、さらに、6 カ国（アルゼンチン、エジプト、エチオピア、イラン、サウジアラビア、アラブ首長国連邦）が加わって、かなり大きな勢力となりつつあるようです。どのぐらいの規模なのか、GDP（国内総生産）、Population（人口）、Oil Production（原油生成量）、Exports of Goods（製品輸出）などが世界のどの程度をしめるのかも、そして、これらの指標の増加・または減少の傾向などについても、調べてみると良いかもしれませんね。

決定係数を求めてみましょう。下から、0.47 ですから、まあまあです。

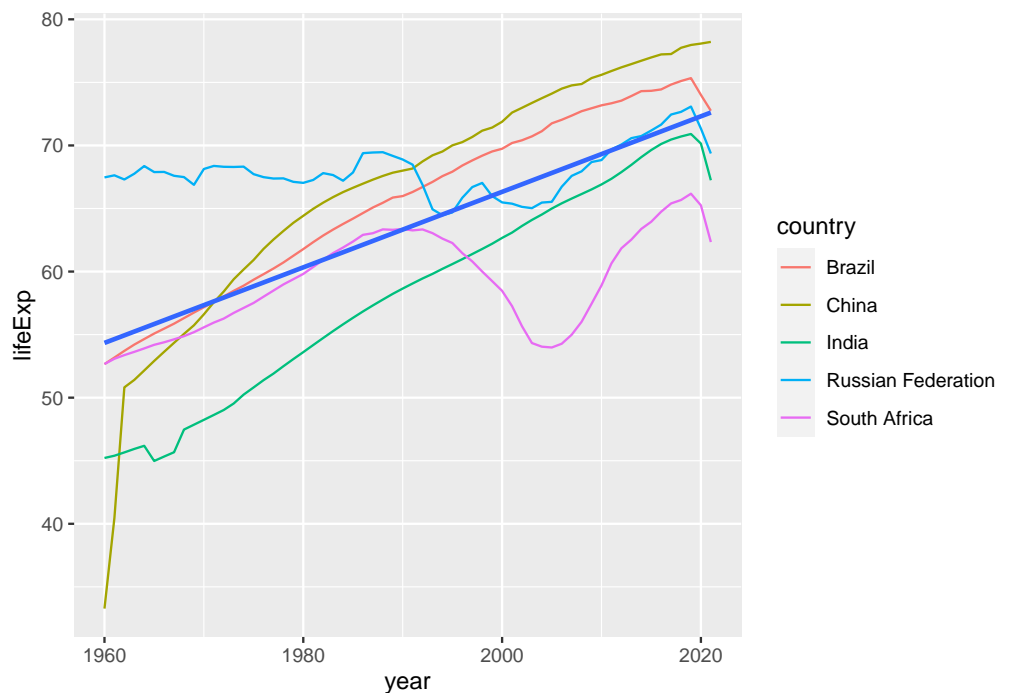
```
mod_brics <- wdi_lifeExp |>
  filter(country %in% c("Brazil", "Russian Federation", "India", "China", "South Africa")) |>
  drop_na(lifeExp) |>
  lm(lifeExp ~ year, data = _) %>% summary()
mod_brics$r.squared
#> [1] 0.4717945
```

そこで、4 カ国の、出生時平均寿命のグラフを描いてみましょう。回帰直線は、データポイント（点）から計算されるものですから、`geom_point()` を使った方が良いかもしれませんが、ある程度長い期間の時系列データですから、`geom_line()` で、折れ線グラフを書いてみました。`geom_line()` の中にある、`color = country` を省略するとどうなるでしょうか。ぜひ試してみてください。`geom_point()` にかえたグラフも描いてみるこ

をお勧めします。

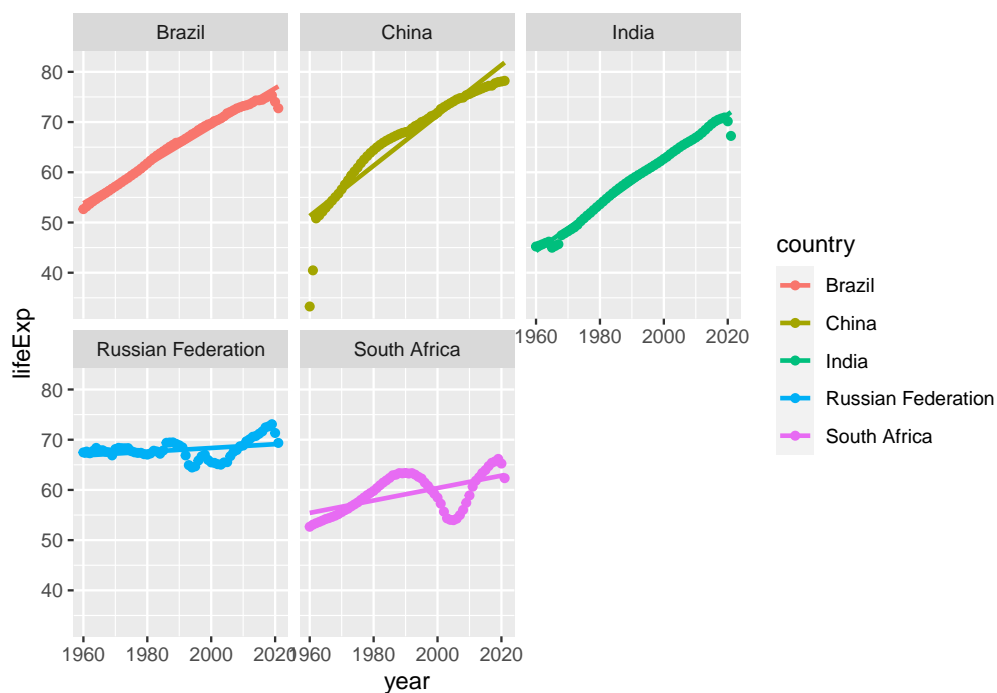
```
wdi_lifeExp |>
  filter(country %in% c("Brazil", "Russian Federation", "India", "China", "South A
  drop_na(lifeExp) |>
  ggplot(aes(year, lifeExp)) + geom_line(aes(color = country)) +
  geom_smooth(formula = y~x, method = "lm", se = FALSE)
```

国ごとにかなり異なることもわかりますね。それも、かなりの変化があります。この場合には、いろいろなモデルのさまざまな値をみるよりも、グラフをていねいにみていくことが大切のように思います。



一応、`geom_point()` に変更し、それぞれの国ごとに、回帰直線を描いたグラフを付けておきます。

```
wdi_lifeExp |>
  filter(country %in% c("Brazil", "Russian Federation", "India", "China", "South A
  drop_na(lifeExp) |>
  ggplot(aes(year, lifeExp, color = country)) + geom_point() +
  geom_smooth(formula = y~x, method = "lm", se = FALSE) + facet_wrap(~country)
```



以下では、それぞれの国ごとに、モデルを作成し tidy や、glance で、その、基本的なデータを、取り出して、表にする方法を紹介します。このようにして、一度に、国ごとのモデルの、さまざまな値を求めることができます。結果だけを出力することも可能ですが、ステップごとの状況も確認できるようにしておきます。

```
country_model <- function(df) {
  lm(lifeExp ~ year, data = df)
}

by_country <- wdi_lifeExp |>
  filter(country %in% c("Brazil", "Russian Federation", "India", "China", "South Africa")) |>
  drop_na(lifeExp) |> group_by(country) |> nest()

by_country
#> # A tibble: 5 x 2
#> # Groups:   country [5]
#>   country      data
#>   <chr>        <list>
#> 1 Brazil      <tibble [62 x 4]>
#> 2 China      <tibble [62 x 4]>
#> 3 India      <tibble [62 x 4]>
#> 4 Russian Federation <tibble [62 x 4]>
#> 5 South Africa <tibble [62 x 4]>
```

```

model_by_country <- by_country %>%
  mutate(model = map(data, country_model))
model_by_country
#> # A tibble: 5 x 3
#> # Groups:   country [5]
#>   country      data      model
#>   <chr>      <list>    <list>
#> 1 Brazil    <tibble [62 x 4]> <lm>
#> 2 China    <tibble [62 x 4]> <lm>
#> 3 India    <tibble [62 x 4]> <lm>
#> 4 Russian Federation <tibble [62 x 4]> <lm>
#> 5 South Africa <tibble [62 x 4]> <lm>

```

```

model_by_country |>
  mutate(tidied = map(model, tidy)) |>
  unnest(tidied)
#> # A tibble: 10 x 8
#> # Groups:   country [5]
#>   country data      model term estimate std.error statistic
#>   <chr> <list> <lis> <chr> <dbl> <dbl> <dbl>
#> 1 Brazil <tibble> <lm> (Int~ -6.94e+2 13.2 -52.7
#> 2 Brazil <tibble> <lm> year 3.81e-1 0.00661 57.7
#> 3 China <tibble> <lm> (Int~ -9.30e+2 46.8 -19.9
#> 4 China <tibble> <lm> year 5.01e-1 0.0235 21.3
#> 5 India <tibble> <lm> (Int~ -8.44e+2 12.1 -69.5
#> 6 India <tibble> <lm> year 4.53e-1 0.00610 74.3
#> 7 Russia~ <tibble> <lm> (Int~ -7.12e+0 26.0 -0.273
#> 8 Russia~ <tibble> <lm> year 3.77e-2 0.0131 2.88
#> 9 South ~ <tibble> <lm> (Int~ -1.87e+2 45.5 -4.12
#> 10 South ~ <tibble> <lm> year 1.24e-1 0.0228 5.42
#> # i 1 more variable: p.value <dbl>

```

```

model_by_country |>
  mutate(glanced = map(model, broom::glance)) |>
  unnest(glanced)
#> # A tibble: 5 x 15
#> # Groups:   country [5]
#>   country      data      model r.squared adj.r.squared sigma
#>   <chr>      <list>    <lis> <dbl> <dbl> <dbl>
#> 1 Brazil    <tibble> <lm> 0.982 0.982 0.932
#> 2 China    <tibble> <lm> 0.883 0.881 3.32
#> 3 India    <tibble> <lm> 0.989 0.989 0.860

```



```
#> 4 Russian Feder~ <tibble> <lm>      0.122      0.107 1.84
#> 5 South Africa <tibble> <lm>      0.329      0.317 3.22
#> # i 9 more variables: statistic <dbl>, p.value <dbl>,
#> #   df <dbl>, logLik <dbl>, AIC <dbl>, BIC <dbl>,
#> #   deviance <dbl>, df.residual <int>, nobs <int>
```

21.5 数理モデルに関する参考文献

- Tidy Modeling with R : tidymodel パッケージ群の教科書
 - <https://www.tmwr.org>
- Tidymodels パッケージ群
 - <https://www.tidymodels.org>
 - broom: <https://cran.r-project.org/web/packages/broom/index.html>
 - Introduction to broom: <https://cran.r-project.org/web/packages/broom/vignettes/broom.html>
- 統計学の時間
 - <https://bellcurve.jp/statistics/course/>

21.5.1 モデル概要に現れる数値について

- r-statistics.co by Selva Prabhakaran:
 - <http://r-statistics.co/Linear-Regression.html>
- Meaning Behind Each Section of Summary()
 - <https://www.learnbymarketing.com/tutorials/explaining-the-lm-summary-in-r/>

第 22 章

情報共有 (Communicate)

22.1 はじめに

データサイエンスでは、再現性 (reproducibility) や、プログラムの文書化 (literate programming) が大切で、そのために、記録をとり、RMarkdown を活用することをすでに述べました。

ここでは、データサイエンスを通して得た情報・発見を共有するためのレポートや、スライドなどの作成についてまとめておきます。基本的には、RMarkdown を中心として書きます。

また、Quarto の活用についても、書きます。

22.2 R Markdown について

すでに、RMarkdown の章で、簡単な使い方を紹介しました。また、この文書も含め、共有している、Posit Cloud のファイルなどで、RMarkdown についてある程度理解しておられると思います。

最初には、RNotebook という、RMarkdown の一つの形式について紹介しました。まず、RStudio で、メニューから File の New File を選択すると、上から順に、Quarto Document、Quarto Presentation、R Notebook、R Markdown と表示されます。Quarto は、新しい形式で、新しく機能も追加されている途中で、R Markdown と共通の機能が使えますので、まずは、R Markdown の説明をし、あとから Quarto について紹介します。

22.3 レポートとスライド

22.3.1 RNotebook/RMarkdown

メニューから、File の New File から RNotebook を選択すると、一番上の部分 (YAML と呼びます) には、

```

---
title: "R Notebook"
output: html_notebook
---
```

とあります。この三つのダッシュの部分、YAML が、RMarkdown の設定部分で、この場合には、title と output とあり、output には、html_notebook とあります。YAML は、書き方の規則が正確に決まっています。字下げの規則もありますので、簡単には変更できませんが、title の引用符の間に、タイトルを入れて、文書を書き始めます。

RMarkdown を選択すると、Document (文書) が選択されている初期設定窓が出て、そこに Title (文書タイトル) と、Author (著者名) を入力するようになっており、右の枠には、HTML、PDF、Word と表示されます。HTML を選択したまま、Title (ここでは「文書名」と Author (ここでは「あなたの名前」) を入力すると、YAML は次のようになっていると思います。

```

---
title: "文書名"
author: "あなたの名前"
date: "2023-09-22"
output: html_document
---
```

ここで、RNotebook で、HTML 出力をした場合と、RMarkdown の HTML 出力の場合の違いを簡単に説明しておきます。

初期設定窓からは、Presentation (スライド)、Shiny、From Template と選択できます。Shiny は、Web アプリのようなものです。スライドについては、あとから説明しますが、いろいろと試してみるのもよいと思います。

22.3.1.1 Visual Editor

RStudio には、左上に、Source と Visual という Editor (エディター) の選択ができます。Visual Editor (ビジュアル・エディタ) は、2021 年末に登場したのですが、HTML や、MD といったスクリプト言語をご存知の方は、別ですが、Visual Editor の方が楽だと思います。

22.3.1.2 メニュー

Visual ボタンの右からは、B (太字)、I (斜字)、6 つのレベルの表題、箇条書き、番号付き箇条書き、リンク、画像挿入ボタンがあり、その右には、Format (形式選択メニュー)、Insert (挿入メニュー) があり、高機能ではありませんが、直感的に、編集が可能になっています。

その一つ上の行でも、いくつかの機能が選択できるようになっていますが、RNotebook の場合には、Preview と表示され、RMarkdown の場合は、knit と表示されているボタ

ンがあり、その右には、出力設定のためのギアマーク、そして、C マークの Code Chunk (コードチャンク) のボタン、その右には、Run ボタンなどがあります。

Preview ボタンを押すと、HTML ファイルが生成されます。別ウインドウに表示されるかと思いますが、Preview ボタンの右のギアマーク (の右の三角マーク) から、Preview in Viewer Pane を選択すると右下の Viewer に表示されます。

22.3.1.3 Code Chunk

この中に、コードを書くのですが、右上にギアマークと、下向きの三角と、右向きの三角があります。コードチャンクに名前をつけたり、メッセージの表示・非表示、注意 (Warnings) の表示・非表示の切り替えなど、さらには、コードを表示する・しない、実行する・しないなども、選択できるようになっています。詳しくは、下に、Chunk Option とありますから、そのリンク先をみてください。

Code Chunk の一番右の、右向き三角を押すと、コードが実行されます。それより上のコードチャンクに、たとえば、library(tidyverse) などの設定があり、それを実行していないと、エラーになる場合もありますから、一番上から実行するというときには、下向きの三角印を押します。

22.3.1.4 PDF への出力

RNotebook でも、RMarkdown でも、初期設定では、HTML 文書が生成されますが、RNotebook では、Preview ボタン、RMarkdown では、knit ボタンの右にある、三角を押すと、そこから、PDF と Word が選択できるようになっています。RNotebook の場合には、HTML も選択できます。

PDF を選択すれば、PDF に出力されるはずですが、初期設定が必要です。RStudio で R 中の動作確認をみていただくのがよいと思います。

ただ、フォントの都合で、日本語が正しく表示されない場合があります。その時は、下の YAML を参照してください。

22.3.1.5 Word への出力

文書にエラーがなければ、knit ボタン、または、Preview ボタンの右の三角から Word を選べば、Word ファイルが生成されます。

また、テンプレートを使って、次のようにすることも可能です。テンプレートは、一度、knit を使って、生成した、docx ファイルの、スタイルを編集したものを、template.docx として、Project フォルダの文書の同じところに入れてください。

```
---  
word_document:  
  reference_docx: "template.docx"  
---
```

22.3.2 スライド

Presentation につかう、スライドも、いろいろな形式が使えます。しかし、共通に、一つだけ、重要な点があります。

それは、ページおくりです。Heading 2 のところで、ページが変わります。または、三つのダッシュで、次のページに行くことも可能です。

詳細は下の参考文献をご覧ください。

22.3.2.1 PowerPoint への出力

Word と同様にテンプレートを使って次のようにすることも可能です。

```
powerpoint_presentation:  
  reference_doc: my-styles.pptx
```

R Markdown いくつかの Output

```
title: "Testing R Markdown Formats"
```

```
author: "ID Your Name"
```

```
header-includes:
```

```
- \usepackage{xeCJK}  
- \setCJKmainfont{ipaexm.ttf}  
- \setCJKsansfont{ipaexg.ttf}  
- \setCJKmonofont{ipaexg.ttf}
```

```
output:
```

```
html_notebook: default  
html_document: default  
pdf_document: default  
  latex_engine: xelatex  
word_document: default  
powerpoint_presentation: default  
ioslides_presentation: default
```

<https://ds-sl.github.io/intro2r/Rmarkdown-J.html>

22.4 Quarto について

リンク : <https://quarto.org>

Document (文書)、HTML を選択すると、YAML は次のようになっています。

```
---  
title: "Untitled"  
format: html  
editor: visual  
---
```

初期設定画面で Author (著者名) を入れることも可能です。editor: visual となっていますから、最初から、Visual Editor が開いています。しかし、Source を押せば、Source Editor に変更することも可能です。

あとは、ほとんど、RMarkdown の場合と同じですが、多少違う部分もありますので、上のリンクを参照してください。

Presentation (スライド) では、Reveal JS (HTML)、Beamer (PDF)、PowerPoint (Microsoft PowerPoint または、OpenOffice) いくつか選択ができますが、あとから変更することもできますので、最初は、Reveal JS で、あとから、発表にあったものに変更するのもよいと思います。

22.5 参考文献

- R Markdown クックブック : <https://gedevan-aleksizde.github.io/rmarkdown-cookbook/>
 - 英語版 : <https://bookdown.org/yihui/rmarkdown-cookbook/>
- R Markdown: The Definitive Guide: <https://bookdown.org/yihui/rmarkdown/>

第 23 章

復習

簡単に、`tidyverse` の基本でもある、`dplyr` による変形、`ggplot2` による視覚化、`readr` などによるデータの読み込み、`tidyr` によって、R で使いやすいデータにすることや、データの結合、さらには、少しだけ、数理モデルについてもみてきました。

すでに、はじめてのデータサイエンスでこれらを紹介していますから、復習の意味を、こめて、そこでの一つ一つについて、確認してみましょう。

また、`dplyr` と、`ggplot2` のところで、使った、`gapminder` のデータを、実際の、WDI のデータに置き換えて、分析をしてみましょう。

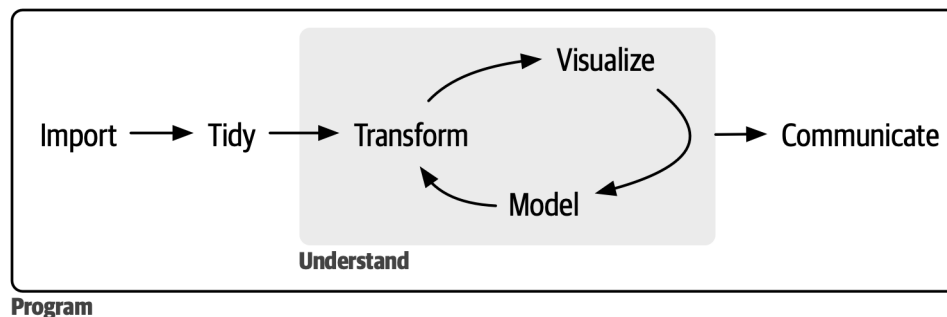
23.1 はじめてのデータサイエンスについて

23.1.1 データサイエンスの実際

データから情報を得るときには、大体次のような手順をとります。

1. 準備 Setup
2. データを取得 Import data
3. データ構造の確認 View data
4. 必要に応じて整形 Transform data
5. 視覚化 Visualize data
6. データを理解 Understand data
7. レポートなどにまとめる Communicate data

下の図は R for Data Science に掲載されている図です。よく、表現されていると思います。詳細は、少しずつ説明します。



はじめにに書きましたが、基本的には、問いをもちデータを取得し、視覚化などを通して、データを理解し、さらに問いを深めるサイクルが、データサイエンスの核だと思います。

R を使った分析の一つの例を、見て行きます。一つ一つのコード（コンピュータ・プログラム）の簡単な説明は、加えますが、あまりそれに捉われず、「データサイエンスとは何か？」を考えながら、雰囲気味わってください。

23.1.2 R のパッケージを活用

23.1.2.1 準備 Setup

世界銀行（World Bank）の、世界開発指標（WDI: World Development Indicators）の一つの、GDP（Gross Domestic Product 国内総生産）のデータから始めます。GDP にも何種類かの尺度がありますが、次のものを見てみます。

- NY.GDP.MKTP.CD: GDP (current US\$)

NY.GDP.MKTP.CD は、データコードと言われるもので、世界開発指標（WDI）には、一つずつ決まっています。

World Development Indicators のサイトの下にある、Data Themes（テーマ）からテーマを選択し、下にスクロールすると、Code をみることができます。ちなみに、ここで利用する NY.GDP.MKTP.CD: GDP (current US\$) は、テーマの Economy（経済）の、一番上にあります。

経済用語の英語はよく知らないという方は、ブラウザ（Edge, Google Chrome, Safari など）の翻訳機能を使うのも良いでしょう。ただ、そのページの対話型の機能（interactive function）を利用するときは、翻訳機能を OFF にする必要がある場合もありますので、注意してください。

エラーメッセージを調べるときなどに、英語のほうが情報がたくさん得られますから、言語を、英語に変更しておきます。

R には、WDI のデータを取得する R のツール（パッケージ）WDI がありますから、それを使います。また、データを取り扱うための基本的なツール（パッケージ）tidyverse を使いますので、次のコードで、これらを読み込みます。`#`> 以下は、コードを実行すると、表示される情報（出力）です。以下同様です。

```

Sys.setenv(LANG = "en") # 言語を英語に
library(tidyverse)      # tidyverse パッケージを読み込みます
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats    1.0.0      v stringr    1.5.0
#> v ggplot2    3.4.3      v tibble     3.2.1
#> v lubridate  1.9.2      v tidyr      1.3.0
#> v purrr      1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(WDI)            # WDI パッケージを読み込みます

```

データを保存する場所を作成しておくことをお勧めします。保存しておくときは、このディレクトリを使います。

```
dir.create("./data")
```

23.1.2.2 データ取得 Import data

データを取得します。少し時間がかかります。取得したデータに、`df_gdp` などと、わかりやすい名前をつけます。`df` は data frame の略で、R で標準的なデータの形式です。

```
df_gdp <- WDI(country = "all",
              indicator = c(gdp = "NY.GDP.MKTP.CD"),
              extra = TRUE)
```

このコードで、全ての国の GDP を取得できます。GDP の値は、`NY.GDP.MKTP.CD` という名前の列にありますが、覚えやすいように、`gdp` という名前に変更しておきます。`extra = TRUE` とすることによって、それぞれの国についての情報などが追加されます。

23.1.2.3 データ構造の確認

最初の数行だけを見るには、`head(df_gdp)` とします。

```

head(df_gdp)
#> # A tibble: 6 x 13
#>   country iso2c iso3c year   gdp status lastupdated region
#>   <chr>    <chr> <chr> <dbl> <dbl> <lgl> <date>    <chr>
#> 1 Afghan~ AF    AFG    1963 7.51e8 NA    2023-07-25 South~
#> 2 Afghan~ AF    AFG    1962 5.47e8 NA    2023-07-25 South~
#> 3 Afghan~ AF    AFG    1961 5.49e8 NA    2023-07-25 South~
#> 4 Afghan~ AF    AFG    1960 5.38e8 NA    2023-07-25 South~

```

```
#> 5 Afghan~ AF    AFG    2003 4.54e9 NA    2023-07-25 South~
#> 6 Afghan~ AF    AFG    2002 3.85e9 NA    2023-07-25 South~
#> # i 5 more variables: capital <chr>, longitude <dbl>,
#> #   latitude <dbl>, income <chr>, lending <chr>
```

データの構造を見るときには、`str(df_gdp)` もよく使われます。今度は、列が縦に並んで表示されます。

```
str(df_gdp)
#> spc_tbl_ [16,758 x 13] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
#> $ country      : chr [1:16758] "Afghanistan" "Afghanistan" "Afghanistan" "Afgha
#> $ iso2c        : chr [1:16758] "AF" "AF" "AF" "AF" ...
#> $ iso3c        : chr [1:16758] "AFG" "AFG" "AFG" "AFG" ...
#> $ year         : num [1:16758] 1963 1962 1961 1960 2003 ...
#> $ gdp          : num [1:16758] 7.51e+08 5.47e+08 5.49e+08 5.38e+08 4.54e+09 ...
#> $ status       : logi [1:16758] NA NA NA NA NA NA ...
#> $ lastupdated: Date[1:16758], format: "2023-07-25" ...
#> $ region      : chr [1:16758] "South Asia" "South Asia" "South Asia" "South As
#> $ capital      : chr [1:16758] "Kabul" "Kabul" "Kabul" "Kabul" ...
#> $ longitude    : num [1:16758] 69.2 69.2 69.2 69.2 69.2 ...
#> $ latitude     : num [1:16758] 34.5 34.5 34.5 34.5 34.5 ...
#> $ income       : chr [1:16758] "Low income" "Low income" "Low income" "Low inco
#> $ lending      : chr [1:16758] "IDA" "IDA" "IDA" "IDA" ...
#> - attr(*, "spec")=
#> .. cols(
#> ..   country = col_character(),
#> ..   iso2c = col_character(),
#> ..   iso3c = col_character(),
#> ..   year = col_double(),
#> ..   gdp = col_double(),
#> ..   status = col_logical(),
#> ..   lastupdated = col_date(format = ""),
#> ..   region = col_character(),
#> ..   capital = col_character(),
#> ..   longitude = col_double(),
#> ..   latitude = col_double(),
#> ..   income = col_character(),
#> ..   lending = col_character()
#> .. )
#> - attr(*, "problems")=<externalptr>
```

概要 (`summary(df_gdp)`) からもある程度わかります。

```
summary(df_gdp)
#>   country          iso2c          iso3c
#> Length:16758      Length:16758      Length:16758
#> Class :character  Class :character  Class :character
#> Mode  :character  Mode  :character  Mode  :character
#>
#>
#>
#>
#>   year          gdp          status
#> Min.   :1960   Min.   :8.825e+06   Mode:logical
#> 1st Qu.:1975   1st Qu.:2.523e+09   NA's:16758
#> Median :1991   Median :1.843e+10
#> Mean   :1991   Mean   :1.207e+12
#> 3rd Qu.:2007   3rd Qu.:2.244e+11
#> Max.   :2022   Max.   :1.006e+14
#>
#>      NA's   :3393
#> lastupdated      region          capital
#> Min.   :2023-07-25 Length:16758      Length:16758
#> 1st Qu.:2023-07-25 Class :character  Class :character
#> Median :2023-07-25 Mode  :character  Mode  :character
#> Mean   :2023-07-25
#> 3rd Qu.:2023-07-25
#> Max.   :2023-07-25
#>
#> longitude      latitude      income
#> Min.   :-175.22   Min.   :-41.286   Length:16758
#> 1st Qu.: -15.18   1st Qu.:  4.174   Class :character
#> Median :  19.54   Median : 17.277   Mode  :character
#> Mean   :  19.16   Mean   : 18.740
#> 3rd Qu.:  50.53   3rd Qu.: 39.715
#> Max.   : 179.09   Max.   : 64.184
#> NA's   :3528     NA's   :3528
#>
#>   lending
#> Length:16758
#> Class :character
#> Mode  :character
#>
#>
#>
#>
```

国のリストをみてみましょう。とても長いリストの中には、地域名も含まれています。

```
df_gdp |> distinct(country) |> pull()
#> [1] "Afghanistan"
#> [2] "Africa Eastern and Southern"
#> [3] "Africa Western and Central"
#> [4] "Albania"
#> [5] "Algeria"
#> [6] "American Samoa"
#> [7] "Andorra"
#> [8] "Angola"
#> [9] "Antigua and Barbuda"
#> [10] "Arab World"
#> [11] "Argentina"
#> [12] "Armenia"
#> [13] "Aruba"
#> [14] "Australia"
#> [15] "Austria"
#> [16] "Azerbaijan"
#> [17] "Bahamas, The"
#> [18] "Bahrain"
#> [19] "Bangladesh"
#> [20] "Barbados"
#> [21] "Belarus"
#> [22] "Belgium"
#> [23] "Belize"
#> [24] "Benin"
#> [25] "Bermuda"
#> [26] "Bhutan"
#> [27] "Bolivia"
#> [28] "Bosnia and Herzegovina"
#> [29] "Botswana"
#> [30] "Brazil"
#> [31] "British Virgin Islands"
#> [32] "Brunei Darussalam"
#> [33] "Bulgaria"
#> [34] "Burkina Faso"
#> [35] "Burundi"
#> [36] "Cabo Verde"
#> [37] "Cambodia"
#> [38] "Cameroon"
#> [39] "Canada"
```

```
#> [40] "Caribbean small states"  
#> [41] "Cayman Islands"  
#> [42] "Central African Republic"  
#> [43] "Central Europe and the Baltics"  
#> [44] "Chad"  
#> [45] "Channel Islands"  
#> [46] "Chile"  
#> [47] "China"  
#> [48] "Colombia"  
#> [49] "Comoros"  
#> [50] "Congo, Dem. Rep."  
#> [51] "Congo, Rep."  
#> [52] "Costa Rica"  
#> [53] "Cote d'Ivoire"  
#> [54] "Croatia"  
#> [55] "Cuba"  
#> [56] "Curacao"  
#> [57] "Cyprus"  
#> [58] "Czechia"  
#> [59] "Denmark"  
#> [60] "Djibouti"  
#> [61] "Dominica"  
#> [62] "Dominican Republic"  
#> [63] "Early-demographic dividend"  
#> [64] "East Asia & Pacific"  
#> [65] "East Asia & Pacific (excluding high income)"  
#> [66] "East Asia & Pacific (IDA & IBRD countries)"  
#> [67] "Ecuador"  
#> [68] "Egypt, Arab Rep."  
#> [69] "El Salvador"  
#> [70] "Equatorial Guinea"  
#> [71] "Eritrea"  
#> [72] "Estonia"  
#> [73] "Eswatini"  
#> [74] "Ethiopia"  
#> [75] "Euro area"  
#> [76] "Europe & Central Asia"  
#> [77] "Europe & Central Asia (excluding high income)"  
#> [78] "Europe & Central Asia (IDA & IBRD countries)"  
#> [79] "European Union"  
#> [80] "Faroe Islands"
```

```
#> [81] "Fiji"  
#> [82] "Finland"  
#> [83] "Fragile and conflict affected situations"  
#> [84] "France"  
#> [85] "French Polynesia"  
#> [86] "Gabon"  
#> [87] "Gambia, The"  
#> [88] "Georgia"  
#> [89] "Germany"  
#> [90] "Ghana"  
#> [91] "Gibraltar"  
#> [92] "Greece"  
#> [93] "Greenland"  
#> [94] "Grenada"  
#> [95] "Guam"  
#> [96] "Guatemala"  
#> [97] "Guinea"  
#> [98] "Guinea-Bissau"  
#> [99] "Guyana"  
#> [100] "Haiti"  
#> [101] "Heavily indebted poor countries (HIPC)"  
#> [102] "High income"  
#> [103] "Honduras"  
#> [104] "Hong Kong SAR, China"  
#> [105] "Hungary"  
#> [106] "IBRD only"  
#> [107] "Iceland"  
#> [108] "IDA & IBRD total"  
#> [109] "IDA blend"  
#> [110] "IDA only"  
#> [111] "IDA total"  
#> [112] "India"  
#> [113] "Indonesia"  
#> [114] "Iran, Islamic Rep."  
#> [115] "Iraq"  
#> [116] "Ireland"  
#> [117] "Isle of Man"  
#> [118] "Israel"  
#> [119] "Italy"  
#> [120] "Jamaica"  
#> [121] "Japan"
```



```
#> [122] "Jordan"
#> [123] "Kazakhstan"
#> [124] "Kenya"
#> [125] "Kiribati"
#> [126] "Korea, Dem. People's Rep."
#> [127] "Korea, Rep."
#> [128] "Kosovo"
#> [129] "Kuwait"
#> [130] "Kyrgyz Republic"
#> [131] "Lao PDR"
#> [132] "Late-demographic dividend"
#> [133] "Latin America & Caribbean"
#> [134] "Latin America & Caribbean (excluding high income)"
#> [135] "Latin America & the Caribbean (IDA & IBRD countries)"
#> [136] "Latvia"
#> [137] "Least developed countries: UN classification"
#> [138] "Lebanon"
#> [139] "Lesotho"
#> [140] "Liberia"
#> [141] "Libya"
#> [142] "Liechtenstein"
#> [143] "Lithuania"
#> [144] "Low & middle income"
#> [145] "Low income"
#> [146] "Lower middle income"
#> [147] "Luxembourg"
#> [148] "Macao SAR, China"
#> [149] "Madagascar"
#> [150] "Malawi"
#> [151] "Malaysia"
#> [152] "Maldives"
#> [153] "Mali"
#> [154] "Malta"
#> [155] "Marshall Islands"
#> [156] "Mauritania"
#> [157] "Mauritius"
#> [158] "Mexico"
#> [159] "Micronesia, Fed. Sts."
#> [160] "Middle East & North Africa"
#> [161] "Middle East & North Africa (excluding high income)"
#> [162] "Middle East & North Africa (IDA & IBRD countries)"
```

```
#> [163] "Middle income"  
#> [164] "Moldova"  
#> [165] "Monaco"  
#> [166] "Mongolia"  
#> [167] "Montenegro"  
#> [168] "Morocco"  
#> [169] "Mozambique"  
#> [170] "Myanmar"  
#> [171] "Namibia"  
#> [172] "Nauru"  
#> [173] "Nepal"  
#> [174] "Netherlands"  
#> [175] "New Caledonia"  
#> [176] "New Zealand"  
#> [177] "Nicaragua"  
#> [178] "Niger"  
#> [179] "Nigeria"  
#> [180] "North America"  
#> [181] "North Macedonia"  
#> [182] "Northern Mariana Islands"  
#> [183] "Norway"  
#> [184] "Not classified"  
#> [185] "OECD members"  
#> [186] "Oman"  
#> [187] "Other small states"  
#> [188] "Pacific island small states"  
#> [189] "Pakistan"  
#> [190] "Palau"  
#> [191] "Panama"  
#> [192] "Papua New Guinea"  
#> [193] "Paraguay"  
#> [194] "Peru"  
#> [195] "Philippines"  
#> [196] "Poland"  
#> [197] "Portugal"  
#> [198] "Post-demographic dividend"  
#> [199] "Pre-demographic dividend"  
#> [200] "Puerto Rico"  
#> [201] "Qatar"  
#> [202] "Romania"  
#> [203] "Russian Federation"
```

```
#> [204] "Rwanda"  
#> [205] "Samoa"  
#> [206] "San Marino"  
#> [207] "Sao Tome and Principe"  
#> [208] "Saudi Arabia"  
#> [209] "Senegal"  
#> [210] "Serbia"  
#> [211] "Seychelles"  
#> [212] "Sierra Leone"  
#> [213] "Singapore"  
#> [214] "Sint Maarten (Dutch part)"  
#> [215] "Slovak Republic"  
#> [216] "Slovenia"  
#> [217] "Small states"  
#> [218] "Solomon Islands"  
#> [219] "Somalia"  
#> [220] "South Africa"  
#> [221] "South Asia"  
#> [222] "South Asia (IDA & IBRD)"  
#> [223] "South Sudan"  
#> [224] "Spain"  
#> [225] "Sri Lanka"  
#> [226] "St. Kitts and Nevis"  
#> [227] "St. Lucia"  
#> [228] "St. Martin (French part)"  
#> [229] "St. Vincent and the Grenadines"  
#> [230] "Sub-Saharan Africa"  
#> [231] "Sub-Saharan Africa (excluding high income)"  
#> [232] "Sub-Saharan Africa (IDA & IBRD countries)"  
#> [233] "Sudan"  
#> [234] "Suriname"  
#> [235] "Sweden"  
#> [236] "Switzerland"  
#> [237] "Syrian Arab Republic"  
#> [238] "Tajikistan"  
#> [239] "Tanzania"  
#> [240] "Thailand"  
#> [241] "Timor-Leste"  
#> [242] "Togo"  
#> [243] "Tonga"  
#> [244] "Trinidad and Tobago"
```

```
#> [245] "Tunisia"  
#> [246] "Turkiye"  
#> [247] "Turkmenistan"  
#> [248] "Turks and Caicos Islands"  
#> [249] "Tuvalu"  
#> [250] "Uganda"  
#> [251] "Ukraine"  
#> [252] "United Arab Emirates"  
#> [253] "United Kingdom"  
#> [254] "United States"  
#> [255] "Upper middle income"  
#> [256] "Uruguay"  
#> [257] "Uzbekistan"  
#> [258] "Vanuatu"  
#> [259] "Venezuela, RB"  
#> [260] "Vietnam"  
#> [261] "Virgin Islands (U.S.)"  
#> [262] "West Bank and Gaza"  
#> [263] "World"  
#> [264] "Yemen, Rep."  
#> [265] "Zambia"  
#> [266] "Zimbabwe"
```

今回は下のように、`|>` (パイプと呼びます) で繋げてコードを書きました。

```
df_gdp |> distinct(country) |> pull()
```

最初は、データ、その中の、異なる国を選択して、書き出してくださいというものです。

これは、

```
pull(distinct(df_gdp, country))
```

と同じです。どんどん、かっこの中に入れ子になって複雑になるので、一つ一つのステップを、順に書いたものが、最初のものになります。

```
df_gdp |> head()
```

```
df_gdp |> str()
```

なども可能です。かっこの中に最初に入るものが直前のもの、ここでは、データになっています。

23.1.2.4 必要に応じて整形 Transform data

変数が多いので、日本の部分だけ `filter` を使って選択します。 `country` が `Japan` と一致する場合のみを選択するときは、 `==` を使います。数値ではないので、引用符をつけます。

す。半角を使ってください。

```
df_gdp |> filter(country == "Japan")
#> # A tibble: 63 x 13
#>   country iso2c iso3c year    gdp status lastupdated
#>   <chr>   <chr> <chr> <dbl> <dbl> <lgl> <date>
#> 1 Japan   JP    JPN  2022 4.23e12 NA    2023-07-25
#> 2 Japan   JP    JPN  2021 5.01e12 NA    2023-07-25
#> 3 Japan   JP    JPN  2020 5.05e12 NA    2023-07-25
#> 4 Japan   JP    JPN  2019 5.12e12 NA    2023-07-25
#> 5 Japan   JP    JPN  2018 5.04e12 NA    2023-07-25
#> 6 Japan   JP    JPN  2017 4.93e12 NA    2023-07-25
#> 7 Japan   JP    JPN  2016 5.00e12 NA    2023-07-25
#> 8 Japan   JP    JPN  2015 4.44e12 NA    2023-07-25
#> 9 Japan   JP    JPN  2014 4.90e12 NA    2023-07-25
#> 10 Japan  JP    JPN  2013 5.21e12 NA    2023-07-25
#> # i 53 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> #   longitude <dbl>, latitude <dbl>, income <chr>,
#> #   lending <chr>
```

```
df_gdp |> filter(country == "Japan") |> head(2)
#> # A tibble: 2 x 13
#>   country iso2c iso3c year    gdp status lastupdated
#>   <chr>   <chr> <chr> <dbl> <dbl> <lgl> <date>
#> 1 Japan   JP    JPN  2022 4.23e12 NA    2023-07-25
#> 2 Japan   JP    JPN  2021 5.01e12 NA    2023-07-25
#> # i 6 more variables: region <chr>, capital <chr>,
#> #   longitude <dbl>, latitude <dbl>, income <chr>,
#> #   lending <chr>
```

2行目の、gdp の、4.940878e+12 は、Scientific notation とされるもので、

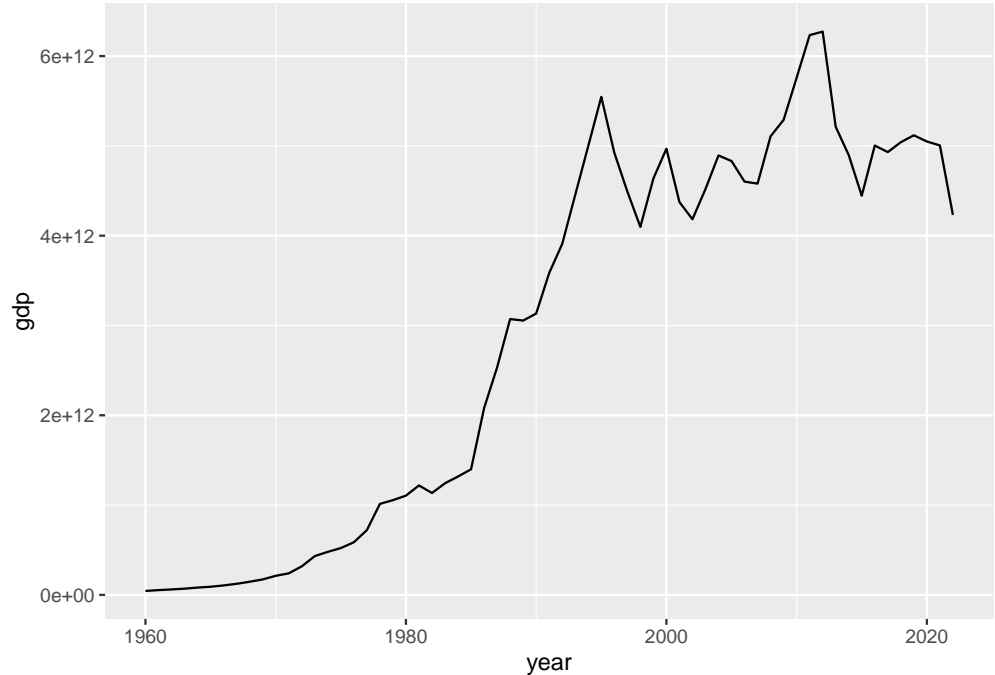
$$4.940878 \times 10^{12} = 4,940,887,800,000$$

を意味します。e+3 は千 (thousand)、e+6 は百万 (million)、e+9 は、10 億 (billion)、e+12 は、兆 (trillion) ですから、日本の、2021 年の GDP は、約 5 兆ドルとなります。

23.1.2.5 視覚化 data visualization

```
df_gdp |> filter(country == "Japan") |>
  ggplot(aes(x = year, y = gdp)) + geom_line()
```

■23.1.2.5.1 Fig 1. 日本の GDP の経年変化を折線グラフ (line graph)



```
df_gdp |> filter(country == "Japan") |>
  ggplot(aes(x = year, y = gdp)) + geom_line()
```

日本を選択したときに、それに名前をつけておいて、それを使うこともできますが、名前がどんどん増えるので、それに続けて、コードを書いていく方法をとっています。

```
ggplot(aes(x = year, y = gdp)) + geom_line()
```

の部分が、グラフを描く部分で、「x 軸を、year、y 軸を、gdp として、それを、折線グラフで描いてください」というコードです。

Warning: [38;5;238mRemoved 1 row containing missing values

と表示されています。値がない年があることを言っています。2022 年のデータがないことがわかっていますから、最初から削除してこくことも可能です。

23.1.2.6 データの理解 Understand data

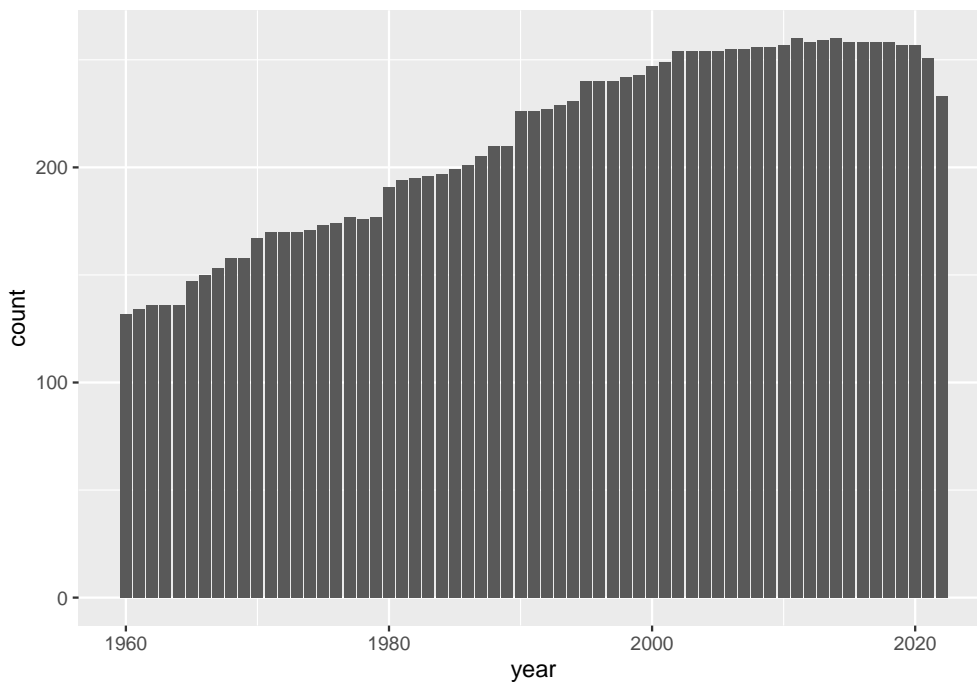
視覚化によって見えてくるものがいくつもありますね。どんなことがわかりますか。気づいたこと (observation) をあげてみましょう。

コードを描くことではなく、この部分が、データサイエンスの核の部分です。気づいたことを列挙してみましょう。

23.1.2.7 さまざまな視覚化

■23.1.2.7.1 Fig 2. 各年ごとのデータの数 `summary(df_gdp)` で、データ自体は、1960年から2022年までのようですが、日本も、2022年のデータはありませんでしたから、年によって、どの程度データがあるか、調べてみます。

```
df_gdp |> drop_na(gdp) |> ggplot(aes(x = year)) + geom_bar()
```



```
df_gdp |> drop_na(gdp) |> ggplot(aes(x = year)) + geom_bar()
```

バー・グラフを使いますが、`gdp` の値が、欠損値 (NA: not available) のデータを削除してから、グラフを描きます。

■23.1.2.7.2 2021年のGDPの降順での表示 (1) 最新の2021年のデータはすべてあるわけではなさそうですが、`gdp` の値が大きい順に並べてみましょう。

```
df_gdp |> filter(year == 2021) |> drop_na(gdp) |> arrange(desc(gdp))
```

```
#> # A tibble: 251 x 13
#>   country      iso2c iso3c  year    gdp status lastupdated
#>   <chr>        <chr> <chr> <dbl>  <dbl> <lgl> <date>
#> 1 World        1W    WLD   2021 9.69e13 NA    2023-07-25
#> 2 High income  XD    <NA>   2021 6.00e13 NA    2023-07-25
#> 3 OECD members OE    OED   2021 5.84e13 NA    2023-07-25
#> 4 Post-demogr~ V4    PST   2021 5.51e13 NA    2023-07-25
#> 5 IDA & IBRD ~ ZT  IBT   2021 3.82e13 NA    2023-07-25
#> 6 Low & middl~ XO    LMY   2021 3.65e13 NA    2023-07-25
```

```
#> 7 Middle inco~ XP MIC 2021 3.61e13 NA 2023-07-25
#> 8 IBRD only XF IBD 2021 3.56e13 NA 2023-07-25
#> 9 East Asia &~ Z4 EAS 2021 3.11e13 NA 2023-07-25
#> 10 Upper middl~ XT <NA> 2021 2.85e13 NA 2023-07-25
#> # i 241 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> # longitude <dbl>, latitude <dbl>, income <chr>,
#> # lending <chr>
```

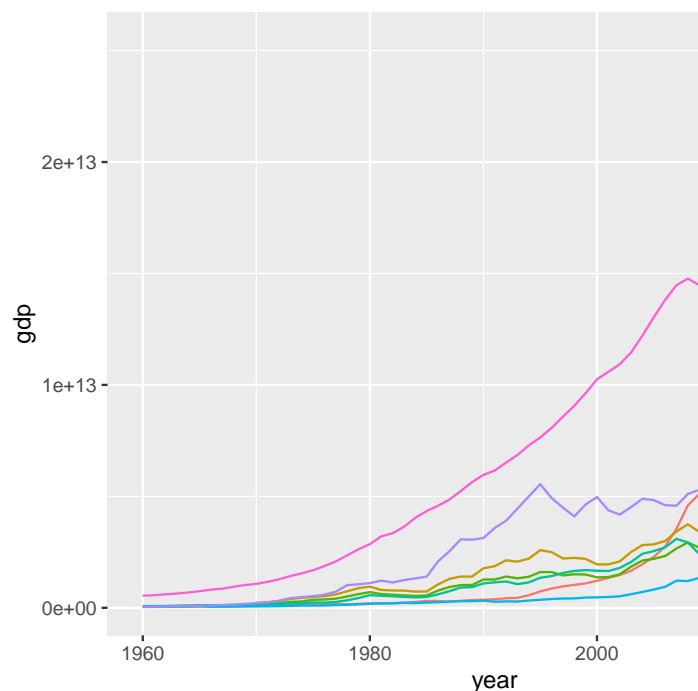
■23.1.2.7.3 2021 年の GDP の降順での表示 (2) 最初に、World と表示され、グループや、カテゴリーのデータもあるようですから、それを、まず、削除することが必要です。region の列を見ると、World などは、Aggregates となっているので、そのようなものを削除すればよさそうです。数値の大きい順に並べたいので、desc 降順 (descending order) にします。

```
df_gdp |> filter(year == 2021, region != "Aggregates") |>
  drop_na(gdp) |> arrange(desc(gdp))
#> # A tibble: 202 x 13
#>   country      iso2c iso3c year   gdp status lastupdated
#>   <chr>         <chr> <chr> <dbl> <dbl> <lgl> <date>
#> 1 United Stat~ US    USA  2021 2.33e13 NA 2023-07-25
#> 2 China        CN    CHN  2021 1.78e13 NA 2023-07-25
#> 3 Japan        JP    JPN  2021 5.01e12 NA 2023-07-25
#> 4 Germany      DE    DEU  2021 4.26e12 NA 2023-07-25
#> 5 India        IN    IND  2021 3.15e12 NA 2023-07-25
#> 6 United King~ GB    GBR  2021 3.12e12 NA 2023-07-25
#> 7 France       FR    FRA  2021 2.96e12 NA 2023-07-25
#> 8 Italy         IT    ITA  2021 2.11e12 NA 2023-07-25
#> 9 Canada       CA    CAN  2021 2.00e12 NA 2023-07-25
#> 10 Russian Fed~ RU    RUS  2021 1.84e12 NA 2023-07-25
#> # i 192 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> # longitude <dbl>, latitude <dbl>, income <chr>,
#> # lending <chr>
```

これは、グラフではありませんが、これも一つの視覚化とも考えられないことはありません。

上位7カ国は、United States, China, Japan, Germany, India, United Kingdom, France であることがわかりました。8番目は、Italy でここまでが、GDP が2兆ドルを越している国となります。


```
df_gdp |> filter(iso2c %in% c("US", "CN", "JP", "DE", "IN", "GB", "FR")) |>
  ggplot(aes(x = year, y = gdp, col = iso2c)) + geom_line()
#> Warning: Removed 10 rows containing missing values
#> (`geom_line()`).
```



■23.1.2.7.4 Fig 3. 2021年時のGDP上位7カ国のGDP経年変化

```
df_gdp |> filter(iso2c %in% c("US", "CN", "JP", "DE", "IN", "GB", "FR")) |>
  ggplot(aes(x = year, y = gdp, col = iso2c)) + geom_line()
```

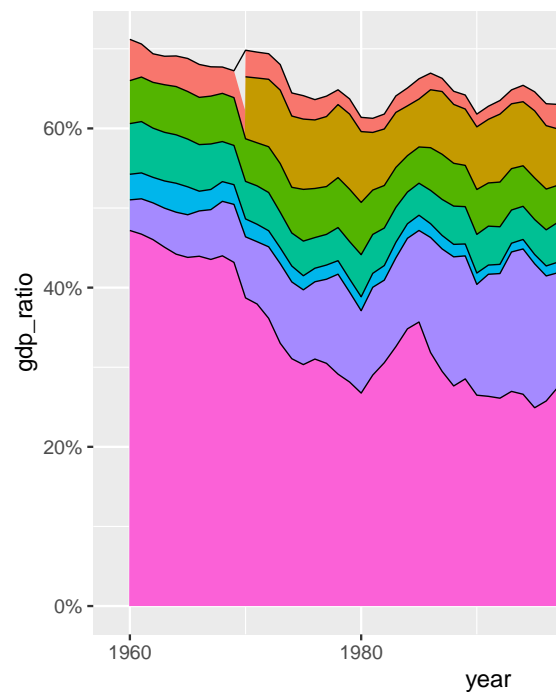
ここでは、最初に、`filter` を使って、7カ国のデータを選択しています。そのときには、`%in%` として、国名を、`combine` するという意味で、`c()` とひとまとめにします。数字ではなく、文字なので、引用符で囲んでいます。この場合は、`single quote` でも構いませんが、半角を使ってください。

このグラフからは、どのようなことがわかりますか。気づいたことを書いてみましょう。

もう少し、このようなグラフをみてみたいというような、メモも大切です。

```
df_gdp |>
  filter(region != "Aggregates") |> drop_na(gdp) |>
  group_by(year) |> mutate(gdp_ratio = gdp/sum(gdp)) |> ungroup() |>
  filter(iso2c %in% c("US", "CN", "JP", "DE", "IN", "GB", "FR")) |>
```

```
ggplot(aes(x = year, y = gdp_ratio, fill = iso2c)) + geom_area() +
geom_line(col = "black", position = "stack", linewidth = 0.3) +
scale_y_continuous(labels = scales::percent_format(accuracy = 1))
```



■23.1.2.7.5 Fig 4. 世界の GDP における割合(1)

まず、下の部分が新しいですが、ここでは、年毎にグループにして、その上で、新しい `dgp_ratio` という名前の列を追加し、その `gdp` の値を、`gdp` 合計で割っています。すなわち、世界の、GDP における割合が計算されています。

```
group_by(year) |> mutate(gdp_ratio = gdp/sum(gdp)) |> ungroup() |>
```

下の部分では、`geom_area` を使って、`fill=iso2c` により、`iso2c` ごとに、違う色を塗って、`position = "stack"` により、積み上げ型の、グラフを描き、境目がわかりやすいように、0.3 の太さの黒の線を描いてください。また、`y` 軸は、小数点以下を省いたパーセント表示に変えてください。というコードです。

```
ggplot(aes(x = year, y = gdp_ratio, fill = iso2c)) + geom_area() +
geom_line(col = "black", position = "stack", linewidth = 0.3) +
scale_y_continuous(labels = scales::percent_format(accuracy = 1))
```

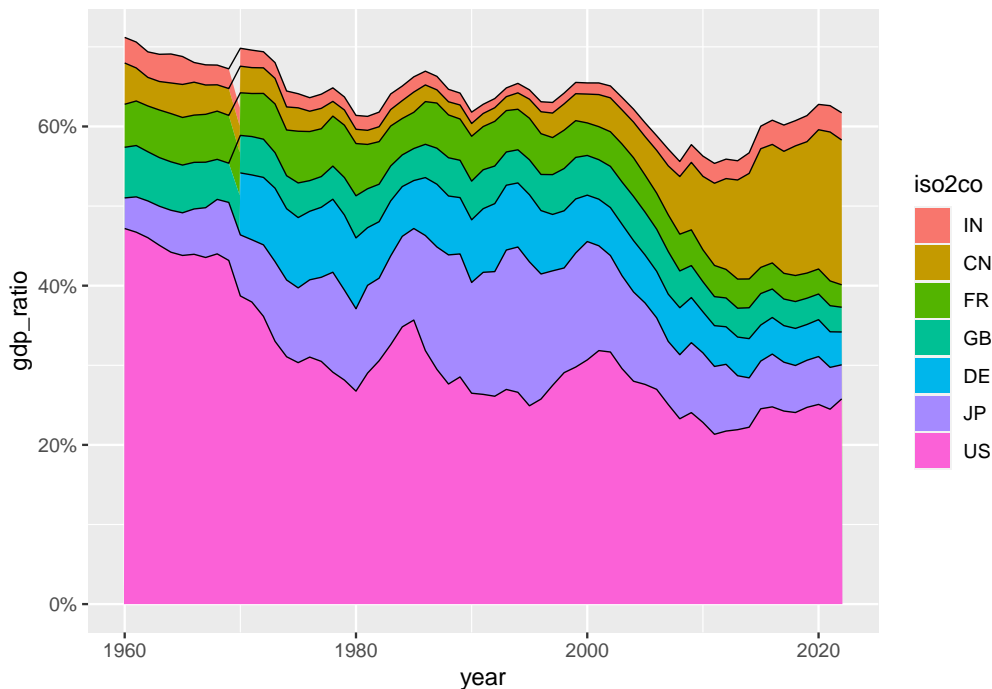
■23.1.2.7.6 Fig 4. 世界の GDP における割合 (2) これは、上から、`iso2c` のアルファベットの順番になっていますが、積み上げの順序を変更することもできます。

```
df_gdp |>
  filter(region != "Aggregates") |> drop_na(gdp) |>
  group_by(year) |> mutate(gdp_ratio = gdp/sum(gdp)) |> ungroup() |>
```

```

filter(iso2c %in% c("US", "CN", "JP", "DE", "IN", "GB", "FR")) |>
mutate(iso2co = factor(iso2c, levels = c("IN", "CN", "FR", "GB", "DE", "JP", "US"))) |>
ggplot(aes(x = year, y = gdp_ratio, fill = iso2co)) + geom_area() +
geom_line(col = "black", position = "stack", linewidth = 0.3) +
scale_y_continuous(labels = scales::percent_format(accuracy = 1))

```



これらは、世界全体の GPT における割合です。

どのようなことがわかりますか。

主要国で、60%~70% を占めていることがわかります。それぞれの国や、幾つかの国の影響力も、ある程度みることができるように見えます。

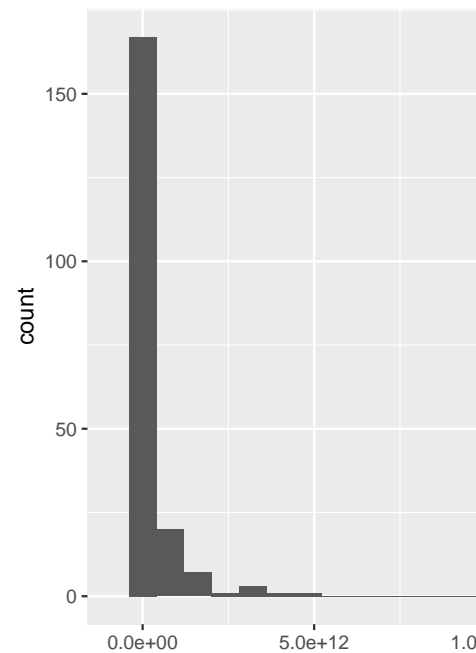
気づいたこと、疑問に思ったことなどを、書き出してみてください。

GDP が大きな国と、小さな国があるのはわかりますが、それは、どのように分布しているのでしょうか。

```

df_gdp |> drop_na(gdp) |>
  filter(year == 2021) |> filter(region != "Aggregates") |>
  ggplot(aes(gdp)) + geom_histogram()
#> `stat_bin()` using `bins = 30`. Pick better value with
#> `binwidth`.

```

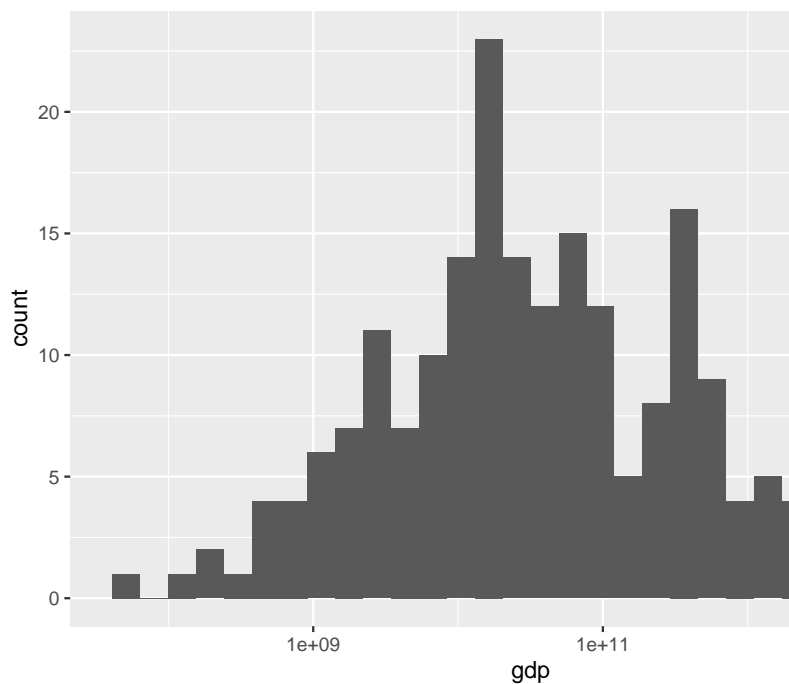


■23.1.2.7.7 Fig 5. 2021 年の世界の国の GDP の分布(1)

小さいところに集中していることがわかりますが、`scale_x_log10()` を加え、対数軸をとってみます。

$\log_{10}(1000) = 3$, $\log_{10}(1000000) = 6$, $\log_{10}(1000000000) = 9$ などになります。

```
df_gdp |> drop_na(gdp) |>
  filter(year == 2021) |> filter(region != "Aggregates") |>
  ggplot(aes(gdp)) + geom_histogram() + scale_x_log10()
#> `stat_bin()` using `bins = 30`. Pick better value with
#> `binwidth`.
```

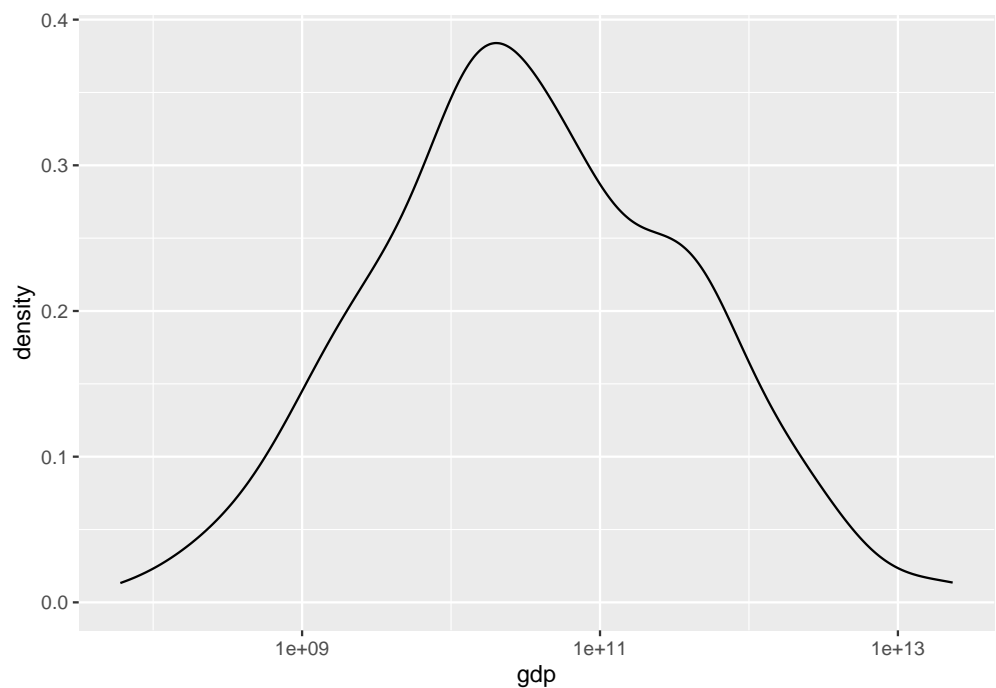


■23.1.2.7.8 Fig 6. 2021年の世界の国のGDPの分布(2)

幅を変更したり、分ける個数を変更するには `binwidth = 0.5` や、`bins = 20` を、`geom_histogram()` のかっこの中に入れます。

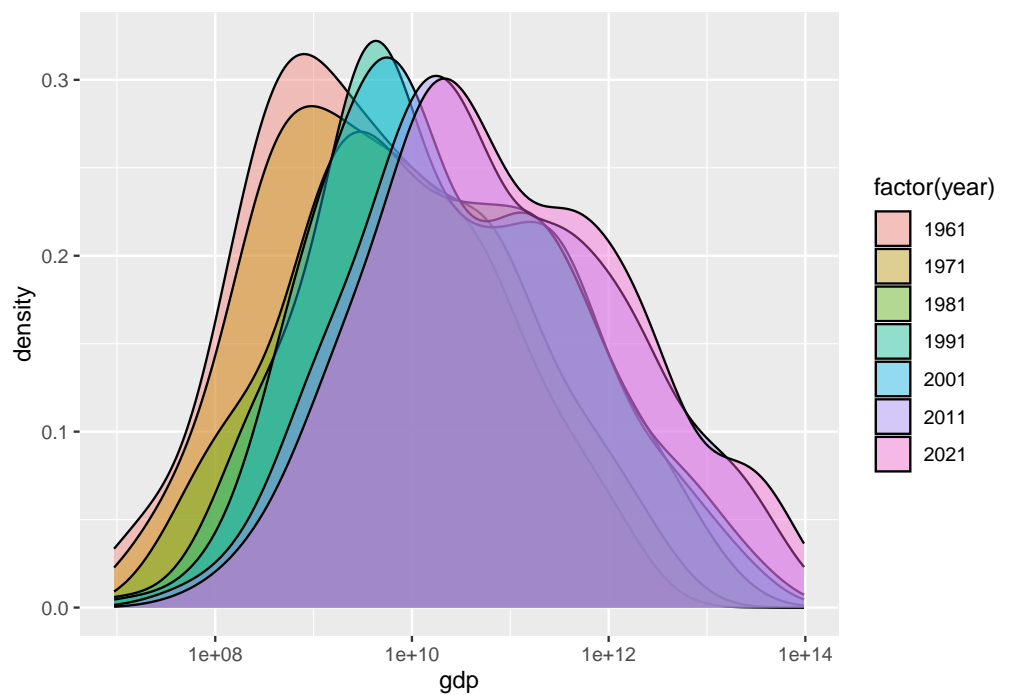
■23.1.2.7.9 Fig 7. 2021年の世界の国のGDPの分布(3) また、密度曲線に変えるには、`geom_density` を使います。

```
df_gdp |> drop_na(gdp) |>
  filter(year == 2021) |> filter(region != "Aggregates") |>
  ggplot(aes(gdp)) + geom_density() + scale_x_log10()
```



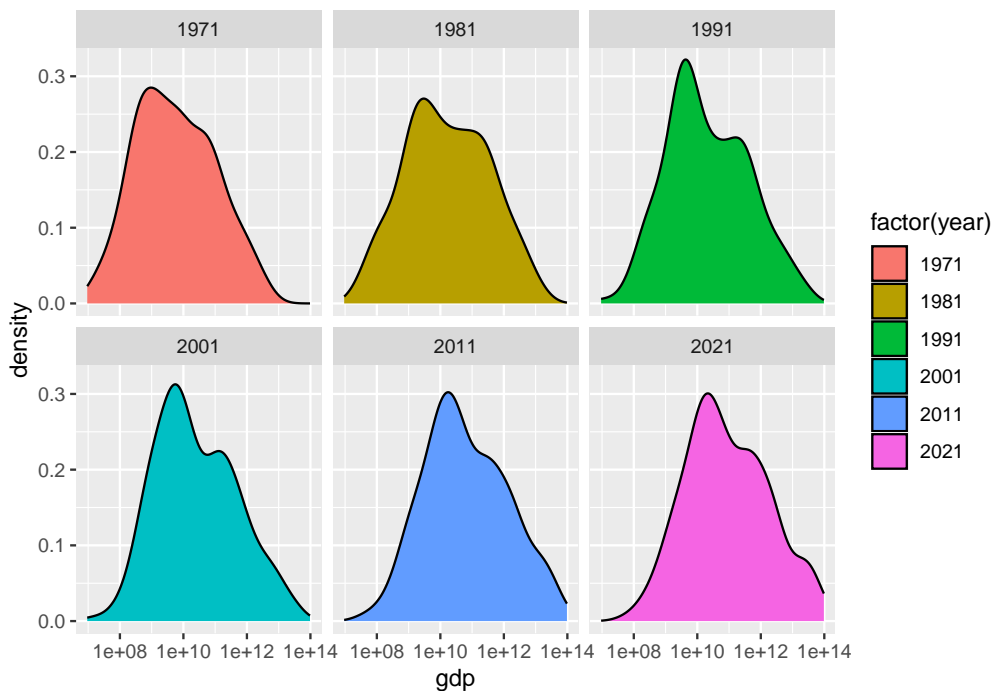
■23.1.2.7.10 Fig 8. 2021 年の世界の国の GDP の分布（4） これは、2021 年のデータですが、density の変化をみます。alpha の値は透明度です。

```
df_gdp |> drop_na(gdp) |>
  filter(year %in% c(1961, 1971, 1981, 1991, 2001, 2011, 2021)) |>
  ggplot(aes(gdp, fill = factor(year))) + geom_density(alpha = 0.4) + scale_x_log
```



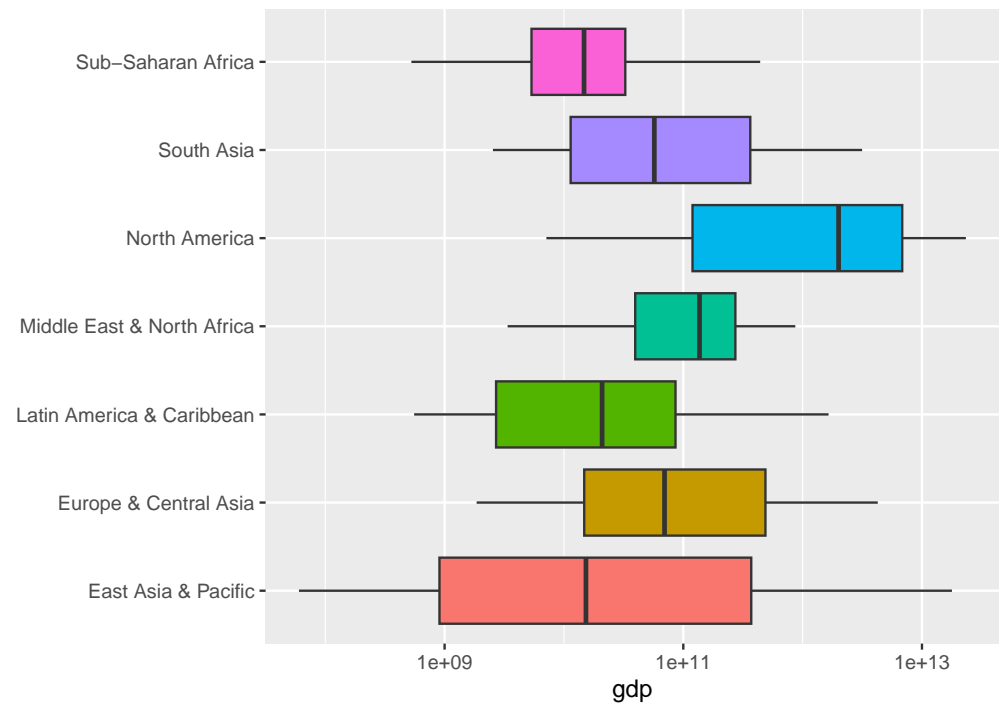
■23.1.2.7.11 Fig 9. 2021 年の世界の国の GDP の分布 (5) 少しみにくいので、分けてみます。

```
df_gdp |> drop_na(gdp) |>
  filter(year %in% c(1971, 1981, 1991, 2001, 2011, 2021)) |>
  ggplot(aes(gdp, fill = factor(year))) +
  geom_density() + scale_x_log10() + facet_wrap(~year)
```

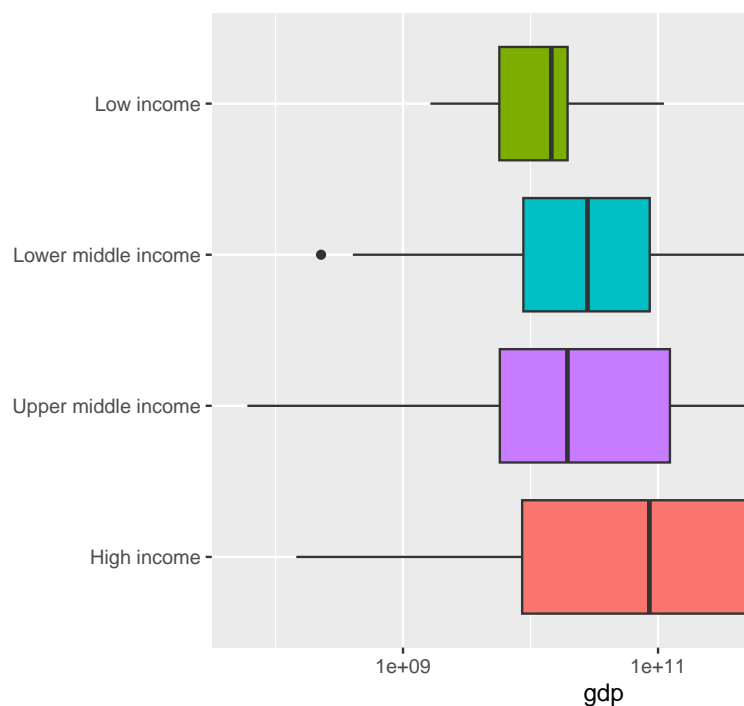


■23.1.2.7.12 Fig 10. 地域ごとの GDP の分布 いくつかのグループごとに分布をみても可能です。それには、Boxplot が有効です。

```
df_gdp |> drop_na(gdp) |> filter(region != "Aggregates") |>
  drop_na(region) |> filter(year %in% c(2021)) |>
  ggplot(aes(gdp, region, fill = region)) +
  geom_boxplot() + scale_x_log10() + labs(y = "") +
  theme(legend.position = "none")
```



```
df_gdp |> drop_na(gdp) |> filter(region != "Aggregates") |>
  drop_na(income) |> filter(year %in% c(2021)) |>
  mutate(level = factor(income, c("High income", "Upper middle income", "Lower mi
ggplot(aes(gdp, level, fill = income)) +
  geom_boxplot() + scale_x_log10() + labs(y = "") +
  theme(legend.position = "none")
```

■23.1.2.7.13 Fig 11. 収入の多寡による分類ごとの GDP 分布

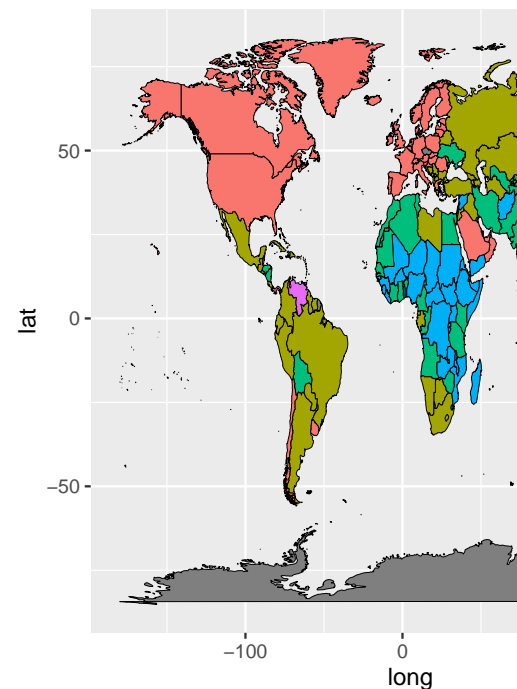
これからも、いろいろなことがわかりますね。

■23.1.2.7.14 世界地図の準備 地図で、国の income level をみてみましょう。

```
library(maps)
#>
#> Attaching package: 'maps'
#> The following object is masked from 'package:purrrr':
#>
#> map
gdp_short <- df_gdp |> filter(year == 2021, region != "Aggregates") |>
  select(iso2c, gdp, income)
map_world <- map_data('world')
map_gdp <- map_world |>
  mutate(iso2c = iso.alpha(region, n=2)) |>
  left_join(gdp_short, by = "iso2c")
head(map_gdp)
#>      long      lat group order region subregion iso2c
#> 1 -69.89912 12.45200     1     1 Aruba      <NA>    AW
#> 2 -69.89571 12.42300     1     2 Aruba      <NA>    AW
#> 3 -69.94219 12.43853     1     3 Aruba      <NA>    AW
#> 4 -70.00415 12.50049     1     4 Aruba      <NA>    AW
#> 5 -70.06612 12.54697     1     5 Aruba      <NA>    AW
```

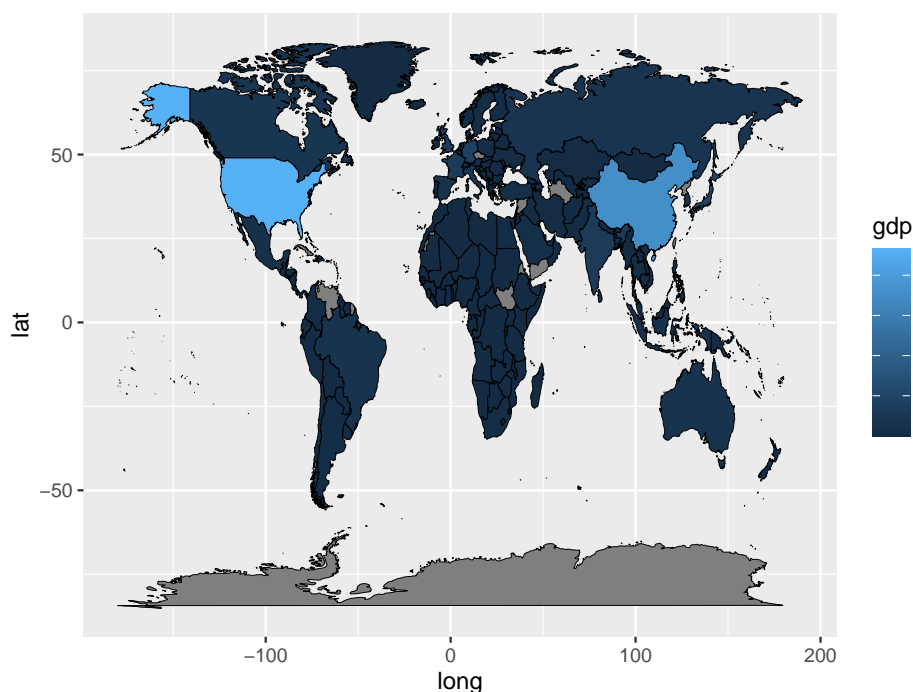
```
#> 6 -70.05088 12.59707 1 6 Aruba <NA> AW
#>      gdp      income
#> 1 3126019385 High income
#> 2 3126019385 High income
#> 3 3126019385 High income
#> 4 3126019385 High income
#> 5 3126019385 High income
#> 6 3126019385 High income
```

```
map_gdp |> mutate(income_level = factor(income, levels = c("High income", "Upper
  ggplot() +
  geom_map(aes(long, lat, map_id = region, fill = income_level), map = map_world,
#> Warning: Using `size` aesthetic for lines was deprecated in ggplot2
#> 3.4.0.
#> i Please use `linewidth` instead.
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where
#> this warning was generated.
#> Warning in geom_map(aes(long, lat, map_id = region, fill =
#> income_level), : Ignoring unknown aesthetics: x and y
```



■23.1.2.7.15 Fig 12. Income Level による色分け地図

```
map_gdp |>
  ggplot() +
  geom_map(aes(x=long, y=lat, map_id = region, fill = gdp), map = map_world, col = "black", size = 0.1)
#> Warning in geom_map(aes(x = long, y = lat, map_id = region,
#> fill = gdp), : Ignoring unknown aesthetics: x and y
```



■23.1.2.7.16 Fig 13. GDP による色分け地図

23.1.3 練習

1. それぞれのグラフから、わかったこと、問いなどを列挙してみましょう。
2. Fig 1 の Japan の部分を他の国や、グループ (World など) に変えてみてください。何がわかりますか。
3. Fig 3 の iso2c で選択する、国を変更してください。何がわかりますか。
4. Fig 4 または Fig 5 の iso2c の部分を他の国に変更してください。何がわかりますか。
5. Fig 5 または Fig 6 の、区間の幅や、数を変更してみてください。何がわかりますか。
6. Fig 7, Fig 8, Fig 9 の年を変更してみてください。何がわかりますか。
7. Fig 12, Fig 13 の年を変更してみてください。何がわかりますか。

23.1.4 プロジェクト

他のデータで、同様のことをしてみましょう。

1. 最初に、`gdp = "NY.GDP.MKTP.CD"` としましたが、GNI per capita, Atlas method

(current US\$): NY.GNP.PCAP.CD に変更してみましょう。

```
df_gnppcap <- WDI(country = "all",  
                  indicator = c(gnppcap = "NY.GNP.PCAP.CD"),  
                  extra = TRUE)
```

2. World Development Indicators のサイトの下にある、Data Themes (テーマ) から自分が調べたいテーマを選び、そのテーマから、データコードを取得して、同様の分析をしてみてください。データがあまりない場合もありますので、ある程度データが多いものを選択することをお勧めします。

23.2 WDI の実際のデータを使って

23.3 Posit Primers - Remaining Tutorials

5. Iterate – r4ds: Program
 - Introduction to Iteration
 - Map
 - Map Shortcut
 - Multiple Vectors
 - List Columns
6. Write Functions – r4ds: Program
 - Function Basics
 - How to Write a Function
 - Argument Matching
 - Environments and Scoping
 - Control Flow
 - Advanced Control Flow
 - Loops in R
8. Build Interactive Web Apps

第III部

第三部オープンデータの活用

第 24 章

オープンデータの分析

第三部では、実際にオープンデータを使って分析を少しずつ紹介していきます。

特に、API（アプリケーション・プログラム・インターフェース）を用いた、R のパッケージがある場合には、積極的に活用していきます。

オープンデータを提供する、それぞれの機関によって、データの取得方法、データ形式なども異なりますので、すべてを説明することはできませんが、有用な情報を提供します。

第 25 章

World Bank

25.1 World Development Indicator (WDI)

パッケージと tidyverse と WDI を使いますから、下のコードによって、ロードします。

```
library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats   1.0.0      v stringr   1.5.0
#> v ggplot2   3.4.3      v tibble    3.2.1
#> v lubridate 1.9.2      v tidyr     1.3.0
#> v purrr     1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(WDI)
```

まず、三つの例を見てみましょう。なにをしているかわかりますか。考えてみてください。

```
WDI(country = "all", indicator = c(gdp = "NY.GDP.MKTP.CD"),
     extra=TRUE) %>% drop_na(gdp) %>%
  filter(year==max(year), income != "Aggregates") %>%
  drop_na(region) %>% arrange(desc(gdp))
```

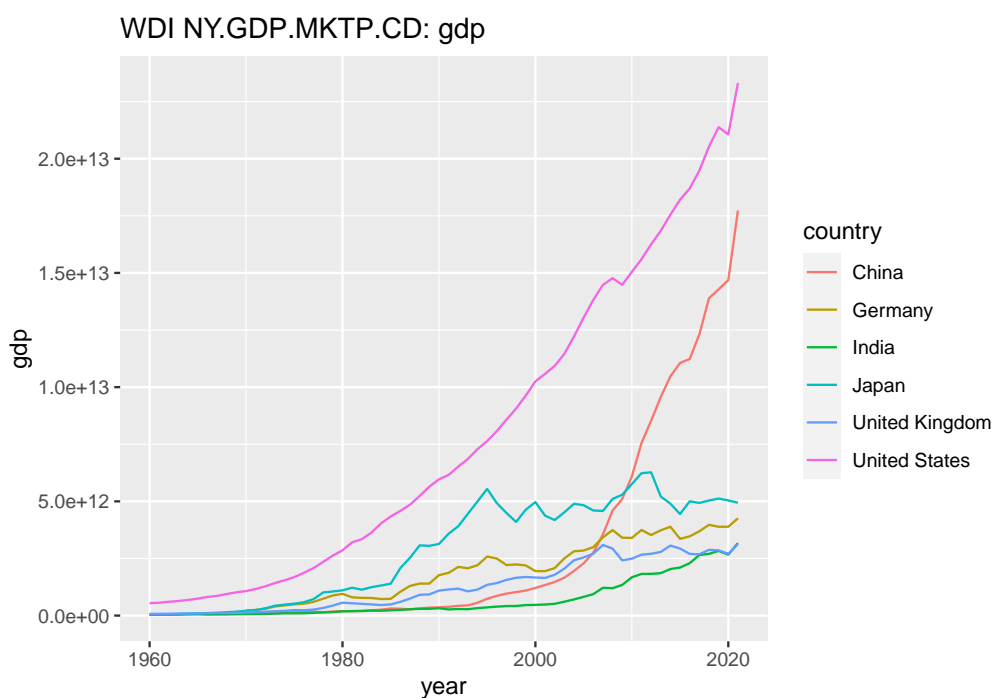
```
#> Rows: 16758 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr  (7): country, iso2c, iso3c, region, capital, income...
#> dbl  (4): year, gdp, longitude, latitude
#> lgl  (1): status
#> date (1): lastupdated
```

```

#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa
#> # A tibble: 184 x 13
#>   country      iso2c iso3c  year    gdp status lastupdated
#>   <chr>         <chr> <chr> <dbl>  <dbl> <lgl> <date>
#> 1 United Stat~ US    USA   2022 2.55e13 NA    2023-07-25
#> 2 China        CN    CHN   2022 1.80e13 NA    2023-07-25
#> 3 Japan        JP    JPN   2022 4.23e12 NA    2023-07-25
#> 4 Germany      DE    DEU   2022 4.07e12 NA    2023-07-25
#> 5 India        IN    IND   2022 3.39e12 NA    2023-07-25
#> 6 United King~ GB    GBR   2022 3.07e12 NA    2023-07-25
#> 7 France       FR    FRA   2022 2.78e12 NA    2023-07-25
#> 8 Russian Fed~ RU    RUS   2022 2.24e12 NA    2023-07-25
#> 9 Canada       CA    CAN   2022 2.14e12 NA    2023-07-25
#> 10 Italy        IT    ITA   2022 2.01e12 NA    2023-07-25
#> # i 174 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> #   longitude <dbl>, latitude <dbl>, income <chr>,
#> #   lending <chr>
WDI(country = c("CN", "GB", "JP", "IN", "US", "DE"), indicator = c(gdp = "NY.GDP.MKTP.
  ggplot(aes(year, gdp, col = country)) + geom_line() +
  labs(title = "WDI NY.GDP.MKTP.CD: gdp")

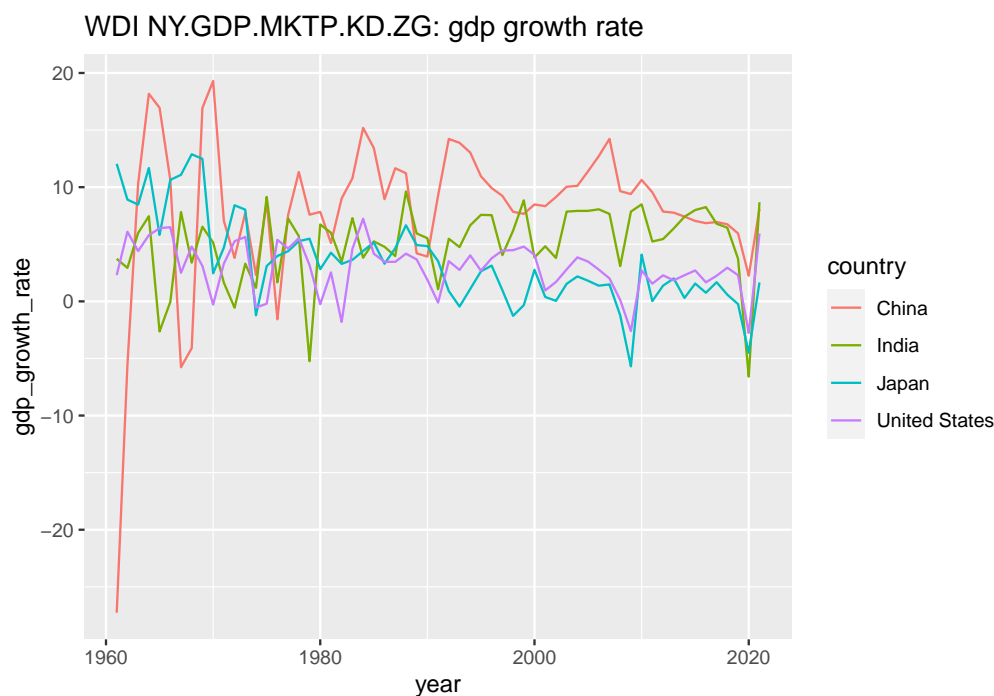
#> Rows: 372 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (4): year, gdp, longitude, latitude
#> lgl (1): status
#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa

```



```
WDI(country = c("CN", "IN", "JP", "US"),
     indicator = c(gdp_growth_rate = "NY.GDP.MKTP.KD.ZG"), extra=TRUE) %>%
drop_na(gdp_growth_rate) %>%
ggplot(aes(year, gdp_growth_rate, col = country)) + geom_line() +
labs(title = paste("WDI NY.GDP.MKTP.KD.ZG: gdp growth rate"))
```

```
#> Rows: 248 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (4): year, gdp_growth_rate, longitude, latitude
#> lgl (1): status
#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```



まず、世界の国々の、GDP (gross domestic product 国内総生産) のデータを、取得して、2021 年の GDP を大きな順に並べています。

値は、たとえば、 $2.331508e + 13$ のように書かれていますが、これは、科学的記法と呼ばれるもので、 2.331508×10^{13} を意味しています。約 23 兆ドルです。

次に、3 兆ドル以上の、6 カ国を選択し、その、iso2c と呼ばれるコードを使って、それらの国のデータをもう一度取得し、年次変化をあらわすグラフを描いています。

さらにその中から、4 カ国を選んで、今度は、GDP の年次変化率を描いています。単位は、パーセントです。

これは、ひとつの例ですが、ここで使われているのが、WDI World Development Indicator というもので、世界銀行が、いくつかの指標を定めて、編纂しているものです。

25.1.1 指標 Indicators (WDI)

上の例では、次の二つの指標のコード Indicator Code (WDI Code) が使われました。

- NY.GDP.MKTP.CD: GDP (current US\$)
- NY.GDP.MKTP.KD.ZG: GDP growth (annual %)

25.1.2 指標 WDI (World Development Indicators)

The World Development Indicators is a compilation of relevant, high-quality, and internationally comparable statistics about global development and the fight against poverty. The database contains 1,400 time series indicators for

217 economies and more than 40 country groups, with data for many indicators going back more than 50 years.

WDI は、世界の開発状況と、貧困との戦いに関する、適切で上質、かつ、国際的に比較可能な時系列の統計データを編纂したものです。このデータベースは、217 の経済と 40 以上の国グループについて 1,400 の時系列指標を含み、指標のデータの多くは 50 年以上前に遡ることができます。

- 世界銀行 (World Bank) : <https://www.worldbank.org>
- World Bank Open Data: <https://data.worldbank.org>
 - Country / Indicator > Featured & All > Details
- World Development Indicators (WDI) :
 - Themes: Poverty and Inequality, People, Environment, Economy, States and Markets, Global Links
 - Open Data & DataBank: Explore data, Query database

25.1.3 指標のコード、WDI code を探してみよう

いくつかの探し方があります。まず、ここでは、World Bank のサイトから探す方法を説明しましょう。

ふた通りあります。

1. World Bank Open Data にいくと、表題の下の検索窓の下に、Country / Indicator とありますから、Indicator を選択します。すると、そこに、項目のリストが、Featured と All という二つのタブに分かれて出ています。かなり膨大です。それを選択すると、その項目のサイトに行きます。それが、指標のサイトです。図などの、右上に、Details とありますから、それを選択すると、その中に、Indicator が書かれています。実は、指標のサイトのアドレス (URL) を見ると、そこにも、この Indicator が書かれていることがわかります。
2. World Development Indicators (WDI) にいくと、下のようなテーマに分かれています。

Themes: Poverty and Inequality, People, Environment, Economy, States and Markets, Global Links

その中から、選択して、スクロールすると、そこに、指標が書かれています。

Indicator, Code, Time coverage, Region coverage, Get data

とあり、Code が、指標のコードです。実は、すべての年や、すべての地域のデータが揃っているわけではないので、この情報を見しておくことはとても重要です。ほとんど、データがない場合もあります。

一番右端の Get data からは、CSV や、データバンク (Data Bank) へのリンクがあり

ます。

それぞれの方法で、上で使った、二つの指標およびそのコードは見つかりましたか。

1 の方法の途中に出てきた、検索窓から検索することも可能です。

25.1.4 指標 WDI の例

このあとの、例で使う指標を書いておきます。

- NY.GDP.MKTP.CD: GDP (current US\$)
- NY.GDP.DEFL.KD.ZG: Inflation, GDP deflator (annual %)
- SL.UEM.TOTL.NE.ZS: Unemployment, total (% of total labor force) (national estimate)
- CPTOTNSXN: CPI Price, nominal
- SL.TLF.CACT.MA.NE.ZS: Labor force participation rate, male (% of male population ages 15+) (national estimate)
- SL.TLF.CACT.FE.NE.ZS: Labor force participation rate, female (% of male population ages 15+) (national estimate)

25.1.5 練習 1. - 調べてみたい WDI 指標とそのコード

いくつか、リストしてみましょう。

25.2 WDI パッケージ

WDI パッケージの使い方を紹介します。

WDI パッケージで、データをダウンロードしたり、探したり、詳細情報を得たりできます。

25.2.1 指標 WDI 検索

まず、検索です。上で、サイトから調べる方法を紹介しましたが、WDI パッケージの、`WDIsearch` でも探すことができます。詳細は、右下の窓枠の `Help` タブの検索窓に、`WDIsearch` といれて調べてみてください。ここでは、二種類の検索方法を紹介します。

25.2.1.1 検索例 1 (WDI 名)

WDI 名に、ある文字列が含まれているものを検索します。検索文字列は、大文字・小文字は関係ありません。

```
WDIsearch(string = "gdp", field = "name", short = TRUE, cache = NULL) %>%
  as_tibble()
#> # A tibble: 540 x 2
```

```
#>   indicator      name
#>   <chr>         <chr>
#> 1 5.51.01.10.gdp  "Per capita GDP growth"
#> 2 6.0.GDP_current "GDP (current $)"
#> 3 6.0.GDP_growth  "GDP growth (annual %)"
#> 4 6.0.GDP_usd     "GDP (constant 2005 $)"
#> 5 6.0.GDPpc_constant "GDP per capita, PPP (constant 2011~
#> 6 BG.GSR.NFSV.GD.ZS "Trade in services (% of GDP)"
#> 7 BG.KAC.FNEI.GD.PP.ZS "Gross private capital flows (% of ~
#> 8 BG.KAC.FNEI.GD.ZS  "Gross private capital flows (% of ~
#> 9 BG.KLT.DINV.GD.PP.ZS "Gross foreign direct investment (%~
#> 10 BG.KLT.DINV.GD.ZS "Gross foreign direct investment (%~
#> # i 530 more rows
```

なんと、500 件以上出てきました。Indicator (指標コード) と、Name (指標名) が列挙されます。すべてに、GDP という文字列が入っていることを確認できると思います。

25.2.1.2 検索例 2 (WDI)

Indicator (指標コード) から、Name (指標名) を検索します。

```
WDIsearch(string = "NY.GDP.MKTP.CD", field = "indicator", short = TRUE, cache = NULL)
#>   indicator
#> 11410 NY.GDP.MKTP.CD
#> 11411 NY.GDP.MKTP.CD.XD
#>
#>   name
#> 11410 GDP (current US$)
#> 11411 GDP deflator, index (2000=100; US$ series)
```

二件出てきました。

25.2.1.3 練習 2. - 検索 (short)

名前で検索 (“ ” の間に、(なるべく簡単な) 検索文字列を入れてください。)

```
WDIsearch(string = "", field = "name", short = TRUE, cache = NULL)
```

Indicator で検索 (“ ” の間に、調べたい indicator を入れてください。)

```
WDIsearch(string = "", field = "indicator", short = TRUE, cache = NULL)
```

25.2.1.4 詳しい情報を得るには

上では、Indicator (指標コード) と、Name (指標名) だけでしたが、Description (説明) なども得ることができます。

それには、`short = FALSE` とします。

一回一回、World Bank にアクセスして調べるのは、時間もかかりますから、Indicator と、名前などの情報をもったファイルを手元に持っておくことにします。それには、次のようにします。

```
wdi_cache <- WDIcache()
```

これは、`series` と、`country` の二つのデータ・フレームからなっているリストです。

右上の窓枠 (pane) から、`wdi_cache` を探して、中身を見てみましょう。三角印や、右から二番目の巻物のようなアイコンをクリックすると中身が見えます。

`series` には、すべての指標がリストされ、その情報が書かれています。

また、`country` には、それぞれについて、さまざまな情報が書かれています。これは、とても、たいせつな情報です。国名と、`iso2c`、`iso3c` のようなコードもありますし、地域 (region) や、その国が、どの `income level` (収入の階級) に入るかも書かれています。また、国だけではなく、地域など、グループの名称も含まれています。

今後、さまざまに利用していきたいと思います。

25.2.1.5 検索例 3 (WDI 名)

`short = FALSE` として、検索してみましょう。文字列が入っている、指標名を検索します。

```
WDIsearch(string = "CPI Price", field = "name", short = FALSE, cache = wdi_cache)
#>      indicator
#> 2586  CPTOTNSXN
#> 2587  CPTOTSAXMZGY
#> 2588  CPTOTSAXN
#> 2589  CPTOTSAXNZGY
#>
#>                                     name
#> 2586                                     CPI Price, nominal
#> 2587 CPI Price, % y-o-y, median weighted, seas. adj.
#> 2588                                     CPI Price, nominal, seas. adj.
#> 2589 CPI Price, % y-o-y, nominal, seas. adj.
#>
#> 2586                                     The consum
#> 2587                                     Median inj
#> 2588                                     The consumer price index re
#> 2589 The consumer price index reflects the change in prices for the average co
#>
#>      sourceDatabase
#> 2586 Global Economic Monitor
#> 2587 Global Economic Monitor
#> 2588 Global Economic Monitor
```



```
#> 2589 Global Economic Monitor
#>
#> 2586 World Bank staff calculations based on Datastream data.
#> 2587 World Bank staff calculations based on Datastream data.
#> 2588 World Bank staff calculations based on Datastream data.
#> 2589 World Bank staff calculations based on Datastream data.
```

- CPTOTNSXN: CPI Price, nominal
 - The consumer price index reflects the change in prices for the average consumer of a constant basket of consumer goods. Data is not seasonally adjusted.

25.2.1.6 検索例 4 (WDI)

指標コードから、詳細情報を得ます。

```
WDIsearch(string = "NY.GDP.MKTP.KD.ZG", field = "indicator", short = FALSE, cache = wdi_cache)
#>
#> 12114 NY.GDP.MKTP.KD.ZG GDP growth (annual %)
#>
#> 12114 Annual percentage growth rate of GDP at market prices based on constant local currency. Aggrego
#>
#> 12114 World Development Indicators
#>
#> 12114 World Bank national accounts data, and OECD National Accounts data files.
```

25.2.1.7 練習 2 - 検索 (long w/ cache)

string と、field を、ふたつとも入れてください。

```
WDIsearch(string = "", field = "", short = FALSE, cache = wdi_cache)
```

25.2.2 指標 WDI データのダウンロード

Indicator が決まったら、ダウンロードします。右下の窓枠の Help タブの検索枠に、WDI と入れて確認しましょう。

```
WDI(
  country = "all",
  indicator = "NY.GDP.PCAP.KD",
  start = 1960,
  end = NULL,
  extra = FALSE,
  cache = NULL,
```

```

    latest = NULL,
    language = "en"
)

```

上が基本的な用法ですが、start 以下は、Default（初期値）が書かれていますから、たいせつなのは、最初の二つ、country と、indicator です。

25.2.2.1 ダウンロード例 1-1

country は、初期値も、“all” となっていますから、最も簡単なのは、indicator に、指標コードを入れることです。引用符を忘れずに。

```
df_gdp1 <- WDI(country = "all", indicator = "NY.GDP.MKTP.CD")
df_gdp1
```

```

#> Rows: 16492 Columns: 5
#> -- Column specification -----
#> Delimiter: ","
#> chr (3): country, iso2c, iso3c
#> dbl (2): year, NY.GDP.MKTP.CD
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa
#> # A tibble: 16,492 x 5
#>   country                iso2c iso3c  year NY.GDP.MKTP.CD
#>   <chr>                  <chr> <chr> <dbl>         <dbl>
#> 1 Africa Eastern and Sout~ ZH    AFE    2021         1.08e12
#> 2 Africa Eastern and Sout~ ZH    AFE    2020         9.27e11
#> 3 Africa Eastern and Sout~ ZH    AFE    2019         1.00e12
#> 4 Africa Eastern and Sout~ ZH    AFE    2018         1.01e12
#> 5 Africa Eastern and Sout~ ZH    AFE    2017         1.02e12
#> 6 Africa Eastern and Sout~ ZH    AFE    2016         8.83e11
#> 7 Africa Eastern and Sout~ ZH    AFE    2015         9.25e11
#> 8 Africa Eastern and Sout~ ZH    AFE    2014         1.00e12
#> 9 Africa Eastern and Sout~ ZH    AFE    2013         9.83e11
#> 10 Africa Eastern and Sout~ ZH    AFE    2012         9.73e11
#> # i 16,482 more rows

```

これでも良いのですが、利用するには、指標コードではわかりにくいので、それを簡単な名前に置き換えて、データを読み込むことができます。

25.2.2.2 ダウンロード例 1-2

```
df_gdp2 <- WDI(country = "all", indicator = c(gdp = "NY.GDP.MKTP.CD"))
df_gdp2
```

```
#> Rows: 16492 Columns: 5
#> -- Column specification -----
#> Delimiter: ","
#> chr (3): country, iso2c, iso3c
#> dbl (2): year, gdp
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
#> # A tibble: 16,492 x 5
#>   country          iso2c iso3c year    gdp
#>   <chr>            <chr> <chr> <dbl> <dbl>
#> 1 Africa Eastern and Southern ZH    AFE    2021 1.08e12
#> 2 Africa Eastern and Southern ZH    AFE    2020 9.27e11
#> 3 Africa Eastern and Southern ZH    AFE    2019 1.00e12
#> 4 Africa Eastern and Southern ZH    AFE    2018 1.01e12
#> 5 Africa Eastern and Southern ZH    AFE    2017 1.02e12
#> 6 Africa Eastern and Southern ZH    AFE    2016 8.83e11
#> 7 Africa Eastern and Southern ZH    AFE    2015 9.25e11
#> 8 Africa Eastern and Southern ZH    AFE    2014 1.00e12
#> 9 Africa Eastern and Southern ZH    AFE    2013 9.83e11
#> 10 Africa Eastern and Southern ZH    AFE    2012 9.73e11
#> # i 16,482 more rows
```

25.2.2.3 ダウンロード例 1-3

今度は、`extra = TRUE` として、読み込みましょう。先ほど、読み込んである、`wdi_cache` を使います。

```
df_gdp3 <- WDI(country = "all", indicator = c(gdp = "NY.GDP.MKTP.CD"),
              extra=TRUE, cache=wdi_cache)
df_gdp3
```

```
#> Rows: 16492 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (4): year, gdp, longitude, latitude
#> lgl (1): status
#> date (1): lastupdated
```

```

#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa
#> # A tibble: 16,492 x 13
#>   country      iso2c iso3c  year      gdp status lastupdated
#>   <chr>         <chr> <chr> <dbl>    <dbl> <lg1> <date>
#> 1 Afghanistan AF    AFG    2021  1.48e10 NA    2022-12-22
#> 2 Afghanistan AF    AFG    2020  2.01e10 NA    2022-12-22
#> 3 Afghanistan AF    AFG    2019  1.89e10 NA    2022-12-22
#> 4 Afghanistan AF    AFG    2018  1.84e10 NA    2022-12-22
#> 5 Afghanistan AF    AFG    2017  1.89e10 NA    2022-12-22
#> 6 Afghanistan AF    AFG    2016  1.80e10 NA    2022-12-22
#> 7 Afghanistan AF    AFG    2015  2.00e10 NA    2022-12-22
#> 8 Afghanistan AF    AFG    2014  2.06e10 NA    2022-12-22
#> 9 Afghanistan AF    AFG    2013  2.06e10 NA    2022-12-22
#> 10 Afghanistan AF    AFG    2012  2.02e10 NA    2022-12-22
#> # i 16,482 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> #   longitude <dbl>, latitude <dbl>, income <chr>,
#> #   lending <chr>

```

右上の三角印を使って、どのような詳細情報が付加されたか見てみましょう。どんなことがわかりますか。

25.2.2.4 ダウンロード例 1-4

国名を指定します。WDI では、iso2c コードを使って、国名を指定します。上で見たように、Environment から、wdi_cache を選択すると、国名と、iso2c コード両方を見ることができました。iso2c や、iso3c は、よく使われるので、web 検索でも簡単にみつけることができます。最初に紹介した例ですから、どの国かはわかりますね、

```

df_gdp4 <- WDI(country = c("CN", "GB", "JP", "IN", "US", "DE"),
               indicator = c(gdp = "NY.GDP.MKTP.CD"), extra=TRUE, cache=wdi_cache)
df_gdp4

```

```

#> Rows: 372 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (4): year, gdp, longitude, latitude
#> lg1 (1): status
#> date (1): lastupdated
#>

```

```

#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
#> # A tibble: 372 x 13
#>   country iso2c iso3c year      gdp status lastupdated
#>   <chr>   <chr> <chr> <dbl>  <dbl> <lgl> <date>
#> 1 China   CN     CHN   2021 1.77e13 NA    2022-12-22
#> 2 China   CN     CHN   2020 1.47e13 NA    2022-12-22
#> 3 China   CN     CHN   2019 1.43e13 NA    2022-12-22
#> 4 China   CN     CHN   2018 1.39e13 NA    2022-12-22
#> 5 China   CN     CHN   2017 1.23e13 NA    2022-12-22
#> 6 China   CN     CHN   2016 1.12e13 NA    2022-12-22
#> 7 China   CN     CHN   2015 1.11e13 NA    2022-12-22
#> 8 China   CN     CHN   2014 1.05e13 NA    2022-12-22
#> 9 China   CN     CHN   2013 9.57e12 NA    2022-12-22
#> 10 China  CN     CHN   2012 8.53e12 NA    2022-12-22
#> # i 362 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> #   longitude <dbl>, latitude <dbl>, income <chr>,
#> #   lending <chr>

```

25.2.2.5 ダウンロード例 2-1

二つの、指標コードを使って、同時に読み込むこともできます。そのときは、`c()` (combine) を使います。

- NY.GDP.DEFL.KD.ZG: Inflation, GDP deflator (annual %)
- CPTOTNSXN: CPI Price, nominal

```

df_gdp21 <- WDI(country = "all",
               indicator = c(gdp_deflator = "NY.GDP.DEFL.KD.ZG",
                           cpi_price = "CPTOTNSXN"),
               extra=TRUE, cache=wdi_cache)

```

```
df_gdp21
```

```

#> Rows: 23972 Columns: 14
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (5): year, gdp_deflator, cpi_price, longitude, lati...
#> lgl (1): status
#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.

```

```

#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa
#> # A tibble: 23,972 x 14
#>   country iso2c iso3c year status lastupdated gdp_deflator
#>   <chr>   <chr> <chr> <dbl> <lgl> <date>           <dbl>
#> 1 Advanc~ AME <NA> 1987 NA 2020-07-27 NA
#> 2 Advanc~ AME <NA> 1988 NA 2020-07-27 NA
#> 3 Advanc~ AME <NA> 1989 NA 2020-07-27 NA
#> 4 Advanc~ AME <NA> 1990 NA 2020-07-27 NA
#> 5 Advanc~ AME <NA> 1991 NA 2020-07-27 NA
#> 6 Advanc~ AME <NA> 1992 NA 2020-07-27 NA
#> 7 Advanc~ AME <NA> 1993 NA 2020-07-27 NA
#> 8 Advanc~ AME <NA> 1994 NA 2020-07-27 NA
#> 9 Advanc~ AME <NA> 1995 NA 2020-07-27 NA
#> 10 Advanc~ AME <NA> 1996 NA 2020-07-27 NA
#> # i 23,962 more rows
#> # i 7 more variables: cpi_price <dbl>, region <chr>,
#> #   capital <chr>, longitude <dbl>, latitude <dbl>,
#> #   income <chr>, lending <chr>

```

NA (not available) つまり、データがないものが多いことがわかります。もう少し、データをよく見て見ましょう。

```

str(df_gdp21)
#> spc_tbl_ [23,972 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
#> $ country      : chr [1:23972] "Advanced Economies" "Advanced Economies" "Advu
#> $ iso2c        : chr [1:23972] "AME" "AME" "AME" "AME" ...
#> $ iso3c        : chr [1:23972] NA NA NA NA ...
#> $ year         : num [1:23972] 1987 1988 1989 1990 1991 ...
#> $ status       : logi [1:23972] NA NA NA NA NA NA ...
#> $ lastupdated  : Date[1:23972], format: "2020-07-27" ...
#> $ gdp_deflator: num [1:23972] NA NA NA NA NA NA NA NA NA NA ...
#> $ cpi_price    : num [1:23972] 58.7 60.5 63 66 69.1 ...
#> $ region       : chr [1:23972] NA NA NA NA ...
#> $ capital      : chr [1:23972] NA NA NA NA ...
#> $ longitude    : num [1:23972] NA NA NA NA NA NA NA NA NA NA ...
#> $ latitude     : num [1:23972] NA NA NA NA NA NA NA NA NA NA ...
#> $ income       : chr [1:23972] NA NA NA NA ...
#> $ lending      : chr [1:23972] NA NA NA NA ...
#> - attr(*, "spec")=
#> .. cols(
#> ..   country = col_character(),
#> ..   iso2c = col_character(),
#> ..   iso3c = col_character(),

```

```

#> .. year = col_double(),
#> .. status = col_logical(),
#> .. lastupdated = col_date(format = ""),
#> .. gdp_deflator = col_double(),
#> .. cpi_price = col_double(),
#> .. region = col_character(),
#> .. capital = col_character(),
#> .. longitude = col_double(),
#> .. latitude = col_double(),
#> .. income = col_character(),
#> .. lending = col_character()
#> .. )
#> - attr(*, "problems")=<externalptr>

```

```
summary(df_gdp21)
```

```

#>   country          iso2c          iso3c
#> Length:23972      Length:23972      Length:23972
#> Class :character  Class :character  Class :character
#> Mode  :character  Mode  :character  Mode  :character
#>
#>
#>
#>   year      status      lastupdated
#> Min.   :1960  Mode:logical  Min.   :2020-07-27
#> 1st Qu.:1982  NA's:23972    1st Qu.:2020-07-27
#> Median :1996                                Median :2022-12-22
#> Mean   :1995                                Mean   :2022-03-23
#> 3rd Qu.:2009                                3rd Qu.:2022-12-22
#> Max.   :2021                                Max.   :2022-12-22
#>
#>   gdp_deflator      cpi_price      region
#> Min.   : -98.704    Min.   : 0.00    Length:23972
#> 1st Qu.:  2.317    1st Qu.: 55.95    Class :character
#> Median :  5.273    Median : 83.28    Mode  :character
#> Mean   : 25.308    Mean   : 84.18
#> 3rd Qu.: 10.411    3rd Qu.:108.75
#> Max.   :26765.858    Max.   :551.25
#> NA's   :11616      NA's   :18410
#>   capital      longitude      latitude
#> Length:23972    Min.   : -175.22  Min.   : -41.286

```

```

#> Class :character 1st Qu.: -15.18 1st Qu.: 4.174
#> Mode :character Median : 19.26 Median : 17.300
#> Mean : 19.14 Mean : 18.889
#> 3rd Qu.: 50.53 3rd Qu.: 40.050
#> Max. : 179.09 Max. : 64.184
#> NA's :10890 NA's :10890
#> income lending
#> Length:23972 Length:23972
#> Class :character Class :character
#> Mode :character Mode :character
#>
#>
#>
#>

```

どんなことが分かりましたか。

右上の窓枠の、Environment でも df_gdp21 を見てみましょう。

25.3 可視化 Visualization

グラフ (Chart) を描いて視覚化します。ここでは、年ごとの変化をみる、折れ線グラフだけを描いて見ます。

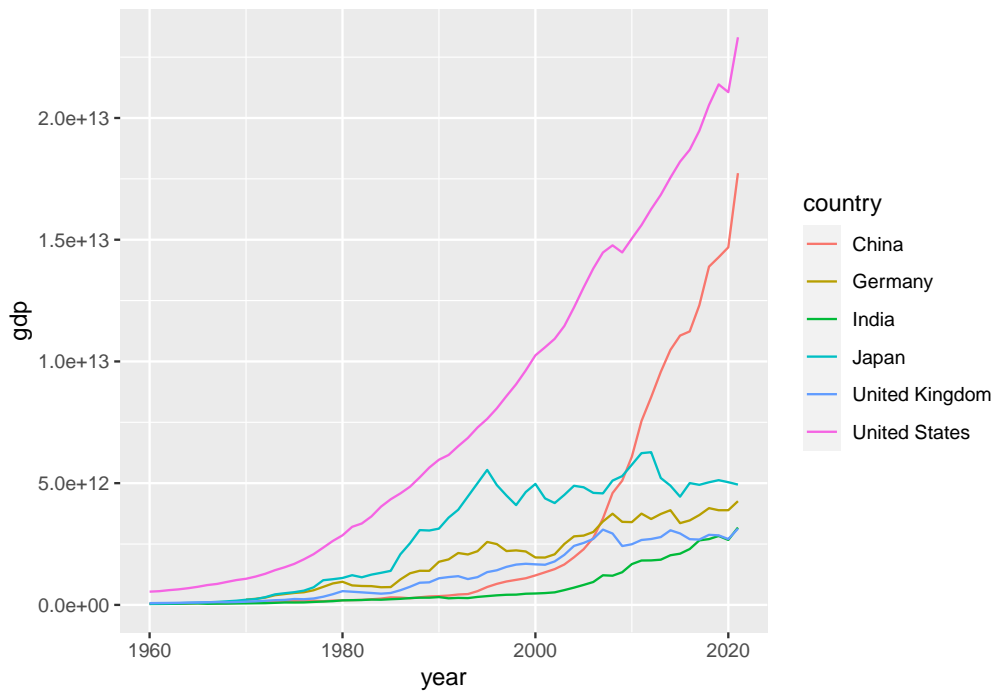
25.3.1 グラフ 1

x = year, y = gdp の、x=, y= は省略してあります。col=country は、国ごとに、グループにして、色分けをします。col は、color としても colour としても、問題ありません。‘

```

df_gdp4 %>% ggplot(aes(year, gdp, col=country)) + geom_line()
#> Warning: Removed 10 rows containing missing values
#> (`geom_line()`).

```

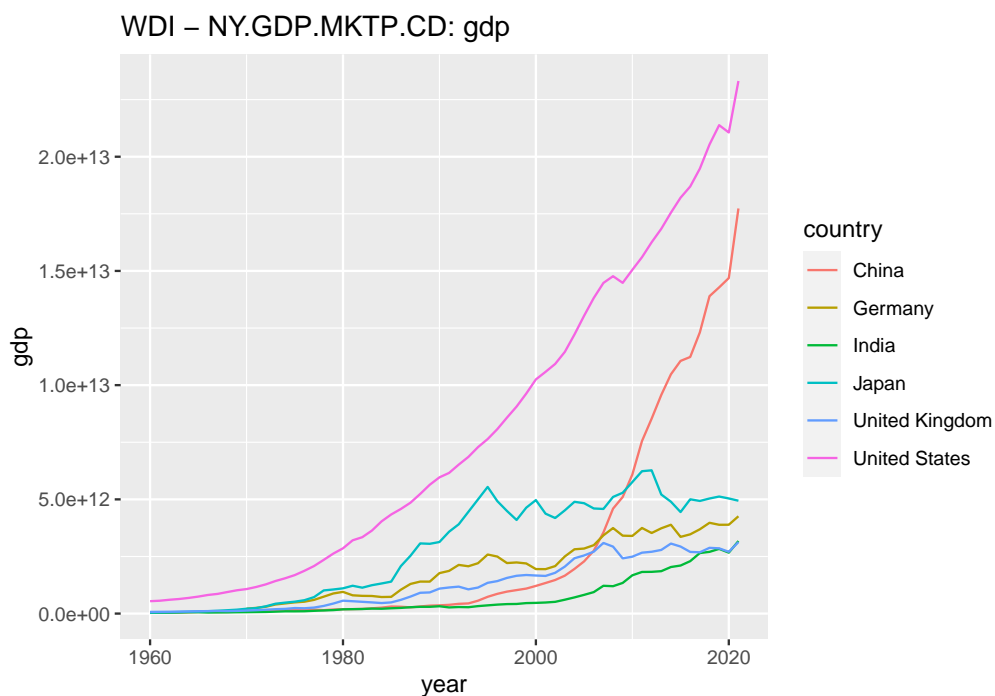



Warning として、missing values があると出ています。どこかは、分かりませんが、図を書くときですから、y に対応する、gdp の値がないものと思われる。

25.3.2 グラフ 2

`drop_na(gdp)` で、gdp の値が、NA であるものを削除します。また、`labs` で、図にタイトルをつけます。

```
df_gdp4 %>% drop_na(gdp) %>%  
  ggplot(aes(year, gdp, col=country)) + geom_line() +  
  labs(title = "WDI - NY.GDP.MKTP.CD: gdp")
```



25.3.3 テンプレート Templates

下に、テンプレートをつけます。コピーして、指標コードや、略称、国などを、それぞれ置き換えて、試して見てください。少し、複雑な変形をしていますが、少しずつ説明します。

25.3.3.1 一つの国についての、一つの指標 (WDI) と、その略称から、折線グラフを作成

Line Plot with one indicator with abbreviation and one country

```
chosen_indicator <- "SL.UEM.TOTL.NE.ZS"
short_name <- "unemployment"
chosen_country <- "United States"
WDI(country = "all", indicator = c(short_name = chosen_indicator), extra=TRUE, ca
  filter(country == chosen_country) %>%
  ggplot(aes(year, short_name)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator, ": ", short_name, " - ", chosen_co
    y = short_name)
```

25.3.3.2 一つの国についての、一つの指標 (WDI) から、折線グラフを作成

Line Plot with one indicator and one country

```
chosen_indicator <- "SL.UEM.TOTL.NE.ZS"
chosen_country <- "United States"
WDI(country = "all", indicator = c(chosen_indicator = chosen_indicator),
  extra=TRUE, cache=wdi_cache) %>%
```

```

filter(country == chosen_country) %>%
ggplot(aes(year, chosen_indicator)) + geom_line() +
labs(title = paste("WDI ", chosen_indicator, " - ", chosen_country),
      y = chosen_indicator)

```

25.3.3.3 いくつかの国についての、一つの指標 (WDI) と、その略称から、折線グラフを作成

Line Plot with one indicator with abbreviation and several countries

```

chosen_indicator <- "SL.UEM.TOTL.NE.ZS"
short_name <- "unemployment"
chosen_countries <- c("United States", "United Kingdom", "Japan")
WDI(country = "all", indicator = c(short_name = chosen_indicator), extra=TRUE, cache=wdi_cache) %>% drop
  filter(country %in% chosen_countries) %>%
  ggplot(aes(year, short_name, col = country)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator, ": ", short_name), y = short_name)

```

25.3.3.4 一つの国についての、二つの指標 (WDI) と、その略称から、折線グラフを作成

Line Plot with two indicators with abbreviation and one country

```

chosen_indicator_1 <- "NY.GDP.DEFL.KD.ZG"
short_name_1 <- "gdp_deflator"
chosen_indicator_2 <- "CPTOTSAXNZGY"
short_name_2 <- "cpi_price"
chosen_country <- "United States"
WDI(country = "all", indicator = c(short_name_1 = chosen_indicator_1, short_name_2 = chosen_indicator_2)
  filter(country == chosen_country) %>%
  pivot_longer(c(short_name_1, short_name_2), names_to = "class", values_to = "value") %>% drop_na(value)
  ggplot(aes(year, value, col = class)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator_1, ": ", short_name_1, "\n", chosen_indicator_2, ": ", short_name_2),
        y = value)
  scale_color_manual(labels = c(short_name_1, short_name_2), values = scales::hue_pal()(2))

```

```

chosen_indicator_1 <- "SL.TLF.CACT.MA.NE.ZS"
short_name_1 <- "male"
chosen_indicator_2 <- "SL.TLF.CACT.FE.NE.ZS"
short_name_2 <- "female"
chosen_country <- "United States"
WDI(country = "all", indicator = c(short_name_1 = chosen_indicator_1, short_name_2 = chosen_indicator_2)
  filter(country == chosen_country) %>%
  pivot_longer(c(short_name_1, short_name_2), names_to = "class", values_to = "value") %>% drop_na(value)
  ggplot(aes(year, value, col = class)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator_1, ": ", short_name_1, "\n", chosen_indicator_2, ": ", short_name_2),
        y = value)

```

```
scale_color_manual(labels = c(short_name_1, short_name_2), values = scales::hue
```

25.3.3.5 いくつかの国についての、二つの指標 (WDI) と、その略称から、折線グラフを作成

Line Plot with two indicators with abbreviation and several countries

```
chosen_indicator_1 <- "NY.GDP.DEFL.KD.ZG"
short_name_1 <- "gdp_deflator"
chosen_indicator_2 <- "CPTOTSAXNZGY"
short_name_2 <- "cpi_price"
chosen_countries <- c("United States", "France", "Japan")
WDI(country = "all", indicator = c(short_name_1 = chosen_indicator_1, short_name_
  filter(country %in% chosen_countries) %>%
  pivot_longer(c(short_name_1, short_name_2), names_to = "class", values_to = "va
  ggplot(aes(year, value, linetype = class, col = country)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator_1, ": ", short_name_1, "\n", chosen
  scale_linetype_manual(labels = c(short_name_1, short_name_2), values = c("solid
```

```
chosen_indicator_1 <- "SL.TLF.CACT.MA.NE.ZS"
short_name_1 <- "male"
chosen_indicator_2 <- "SL.TLF.CACT.FE.NE.ZS"
short_name_2 <- "female"
chosen_countries <- c("United States", "France", "Japan")
WDI(country = "all", indicator = c(short_name_1 = chosen_indicator_1, short_name_
  filter(country %in% chosen_countries) %>%
  pivot_longer(c(short_name_1, short_name_2), names_to = "class", values_to = "va
  ggplot(aes(year, value, linetype = class, col = country)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator_1, ": ", short_name_1, "\n", chosen
  scale_linetype_manual(labels = c(short_name_1, short_name_2), values = c("solid
```

25.4 課題 Assignment

上のテンプレートをコピーして、下に貼り付け、指標 `indicator` と、略称 `short_name` と、いくつかの国名 `chosen_countries` を、入れ替えて、試してみてください。

```
library(tidymodels)
#> -- Attaching packages ----- tidymodels 1.1.1 --
#> v broom          1.0.5      v rsample          1.2.0
#> v dials          1.2.0      v tune             1.1.2
#> v infer          1.0.5      v workflows        1.1.3
#> v modeldata      1.2.0      v workflowsets     1.0.1
#> v parsnip        1.1.1      v yardstick        1.2.0
```

```
#> v recipes      1.0.8
#> -- Conflicts ----- tidymodels_conflicts() --
#> x scales::discard() masks purrr::discard()
#> x dplyr::filter()  masks stats::filter()
#> x recipes::fixed() masks stringr::fixed()
#> x dplyr::lag()     masks stats::lag()
#> x yardstick::spec() masks readr::spec()
#> x recipes::step()  masks stats::step()
#> * Dig deeper into tidy modeling with R at https://www.tmr.org
```

25.5 政府支出（国内総生産比） Government Expenditure, (% of GDP)

ここでは、多変数モデルについては、紹介できませんが、複数の変数を使って、それぞれどのような影響があるかを調べる方法を少しだけ紹介します。

WDI の指標が、出生時平均寿命にどのていど、影響するかを調べてみます。

まずは、指標名の中に、“expenditure” と、“% of GDP” を含むものを探します。下では二つの方法を提示しています。一方は、表の結合の、`inner_join` を使う方法、もう一つは、正規表現の、`grepl` を使う方法です。なにも指定しないと、`grepl` は大文字小文字を区別し、`WDIsearch` は、区別しませんから、ひとつだけ、差が生じています。

```
wdi_cache <- read_rds("./data/wdi_cache.RData")
```

```
WDIsearch("expenditure", "name", cache = wdi_cache) %>%
  inner_join(WDIsearch("% of GDP", "name", cache = wdi_cache))
#> Joining with `by = join_by(indicator, name)`
#>      indicator
#> 1  GB.XPD.DEFN.GDP.ZS
#> 2  GB.XPD.RSDV.GD.ZS
#> 3  GB.XPD.TOTL.GD.ZS
#> 4  GB.XPD.TOTL.GDP.ZS
#> 5  IE.ICT.TOTL.GD.ZS
#> 6  MS.MIL.XPND.GD.ZS
#> 7  NE.CON.GOVT.ZS
#> 8  NE.CON.PETC.ZS
#> 9  NE.CON.PRVT.ZS
#> 10 NE.CON.TETC.ZS
#> 11 NE.CON.TOTL.ZS
#> 12 NE.DAB.TOTL.ZS
#> 13 NY.GEN.AEDU.GD.ZS
```

```

#> 14 SE.XPD.PRIM.PC.ZS
#> 15 SE.XPD.SECO.PC.ZS
#> 16 SE.XPD.TERT.PC.ZS
#> 17 SE.XPD.TOTL.GD.ZS
#> 18 SH.XPD.CHEX.GD.ZS
#> 19 SH.XPD.GHED.GD.ZS
#> 20 SH.XPD.KHEX.GD.ZS
#> 21 SH.XPD.PRIV.ZS
#> 22 SH.XPD.PUBL.ZS
#> 23 SH.XPD.TOTL.ZS
#> 24 UIS.XGDP.0.FSGOV
#> 25 UIS.XGDP.1.FSGOV
#> 26 UIS.XGDP.23.FSGOV
#> 27 UIS.XGDP.2T4.V.FSGOV
#> 28 UIS.XGDP.4.FSGOV
#> 29 UIS.XGDP.56.FSGOV
#>
#> 1
#> 2 Research and devel
#> 3
#> 4
#> 5 Information and communication tech
#> 6 Ma
#> 7 General government final consu
#> 8 Household final consumption
#> 9 Households and NPISHs final consu
#> 10 Final consumption
#> 11 Final consu
#> 12 Gross na
#> 13 Genuine savings: edu
#> 14 Government expenditure per student,
#> 15 Government expenditure per student, se
#> 16 Government expenditure per student, t
#> 17 Government expenditure o
#> 18 Current
#> 19 Domestic general government
#> 20 Capital
#> 21 Health ex
#> 22 Health e
#> 23 Health
#> 24 Government expenditure on pre-prim

```

```
#> 25          Government expenditure on primary education as % of GDP
#> 26          Government expenditure on secondary education as % of GDP
#> 27 Government expenditure on secondary and post-secondary non-tertiary vocational education as % of GDP
#> 28          Government expenditure on post-secondary non-tertiary education as % of GDP
#> 29          Government expenditure on tertiary education as % of GDP
```

```
wdi_cache$series %>% filter(grepl("expenditure", name), grepl("% of GDP", name))
```

```
#>          indicator
#> 1  GB.XPD.DEFN.GDP.ZS
#> 2  GB.XPD.RSDV.GD.ZS
#> 3  GB.XPD.TOTL.GDP.ZS
#> 4  IE.ICT.TOTL.GD.ZS
#> 5  MS.MIL.XPND.GD.ZS
#> 6  NE.CON.GOVT.ZS
#> 7  NE.CON.PETC.ZS
#> 8  NE.CON.PRVT.ZS
#> 9  NE.CON.TETC.ZS
#> 10 NE.CON.TOTL.ZS
#> 11 NE.DAB.TOTL.ZS
#> 12 NY.GEN.AEDU.GD.ZS
#> 13 SE.XPD.PRIM.PC.ZS
#> 14 SE.XPD.SECO.PC.ZS
#> 15 SE.XPD.TERT.PC.ZS
#> 16 SE.XPD.TOTL.GD.ZS
#> 17 SH.XPD.CHEX.GD.ZS
#> 18 SH.XPD.GHED.GD.ZS
#> 19 SH.XPD.KHEX.GD.ZS
#> 20 SH.XPD.PRIV.ZS
#> 21 SH.XPD.PUBL.ZS
#> 22 SH.XPD.TOTL.ZS
#> 23 UIS.XGDP.0.FSGOV
#> 24 UIS.XGDP.1.FSGOV
#> 25 UIS.XGDP.23.FSGOV
#> 26 UIS.XGDP.2T4.V.FSGOV
#> 27 UIS.XGDP.4.FSGOV
#> 28 UIS.XGDP.56.FSGOV
#>
#> 1          Defense expenditure (% of GDP)
#> 2          Research and development expenditure (% of GDP)
#> 3          Total expenditure (% of GDP)
#> 4          Information and communication technology expenditure (% of GDP)
```

```

#> 5
#> 6
#> 7
#> 8
#> 9
#> 10
#> 11
#> 12
#> 13
#> 14
#> 15
#> 16
#> 17
#> 18
#> 19
#> 20
#> 21
#> 22
#> 23
#> 24
#> 25
#> 26
#> 27
#> 28
#>
#> 1
#> 2
#> 3
#> 4
#> 5
#> 6
#> 7
#> 8
#> 9
#> 10
#> 11
#> 12
#> 13
#> 14
#> 15
#> 16

```

Ma
General government final consu
Household final consumption
Households and NPISHs final consu
Final consumption
Final consu
Gross na
Genuine savings: edu
Government expenditure per student,
Government expenditure per student, se
Government expenditure per student, t
Government expenditure o
Current
Domestic general government
Capital
Health ea
Health e
Health
Government expenditure on pre-prim
Government expenditure on prim
Government expenditure on second
Government expenditure on secondary and post-secondary non-tertiary vocation
Government expenditure on post-secondary non-terti
Government expenditure on terta

Military expenditures data from SIPRI are derived from the NATO definition,


```
#> 17
#> 18
#> 19
#> 20
#> 21
#> 22
#> 23
#> 24
#> 25
#> 26
#> 27
#> 28
#>
#> sourceDatabase
#> 1 WDI Database Archives
#> 2 World Development Indicators
#> 3 WDI Database Archives
#> 4 Africa Development Indicators
#> 5 World Development Indicators
#> 6 World Development Indicators
#> 7 WDI Database Archives
#> 8 World Development Indicators
#> 9 WDI Database Archives
#> 10 World Development Indicators
#> 11 World Development Indicators
#> 12 WDI Database Archives
#> 13 World Development Indicators
#> 14 World Development Indicators
#> 15 World Development Indicators
#> 16 World Development Indicators
#> 17 World Development Indicators
#> 18 World Development Indicators
#> 19 Health Nutrition and Population Statistics
#> 20 WDI Database Archives
#> 21 WDI Database Archives
#> 22 WDI Database Archives
#> 23 Education Statistics
#> 24 Education Statistics
#> 25 Education Statistics
#> 26 Education Statistics
#> 27 Education Statistics
#> 28 Education Statistics
```

```

#>
#> 1
#> 2 UNESCO Institute for Statistics (UIS). UIS.Stat Bulk Data Download Service.
#> 3
#> 4 World Information Technology and Services Alliance, Digital
#> 5 Stockholm International Peace Research Institute (SI
#> 6 World E
#> 7 World E
#> 8 World E
#> 9 World E
#> 10 World E
#> 11 World E
#> 12
#> 13 UNESCO Institute
#> 14 UNESCO Institute
#> 15 UNESCO Institute
#> 16 UNESCO Institute for Statistics (UIS). UIS.Stat Bulk Data Download Service.
#> 17 World Health Organization Global Health Expenditure database (http://apps.
#> 18 World Health Organization Global Health Expenditure database (http://apps.
#> 19 World Health Organization Global Health Expenditure database (http://apps.
#> 20 World Health Organization Global Health Expenditure database (
#> 21 World Health Organization Global Health Expenditure database (
#> 22 World Health Organization Global Health Expenditure database (
#> 23
#> 24
#> 25
#> 26
#> 27
#> 28

```

以下の指標のデータを読み込んでみます。

- NY.GDP.PCAP.KD: GDP per capita (constant 2015 US\$) - 国際総生産
- SP.DYN.LE00.IN: Life expectancy at birth, total (years) - 出生時寿命
- SP.POP.TOTL: Population, total - 人口
- GB.XPD.RSDV.GD.ZS: Research and development expenditure (% of GDP) - 2 - 研究開発
- MS.MIL.XPND.GD.ZS: Military expenditure (% of GDP) - 6 - 軍事費
- SE.XPD.TOTL.GD.ZS: Government expenditure on education, total (% of

GDP) - 教育費

```
wdi_world <- WDI(country = "all", indicator = c(gdpPcap = "NY.GDP.PCAP.KD", lifeExp = "SP.DYN.LE00.IN",
```

```
#> Rows: 8778 Columns: 18
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (9): year, gdpPcap, lifeExp, pop, research, militar...
#> lgl (1): status
#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
wdi_world
#> # A tibble: 8,778 x 18
#>   country      iso2c iso3c  year status lastupdated gdpPcap
#>   <chr>         <chr> <chr> <dbl> <lgl> <date>      <dbl>
#> 1 Afghanistan AF    AFG   1990 NA    2023-07-25    NA
#> 2 Afghanistan AF    AFG   1991 NA    2023-07-25    NA
#> 3 Afghanistan AF    AFG   1993 NA    2023-07-25    NA
#> 4 Afghanistan AF    AFG   2015 NA    2023-07-25   592.
#> 5 Afghanistan AF    AFG   1999 NA    2023-07-25    NA
#> 6 Afghanistan AF    AFG   1992 NA    2023-07-25    NA
#> 7 Afghanistan AF    AFG   1996 NA    2023-07-25    NA
#> 8 Afghanistan AF    AFG   1997 NA    2023-07-25    NA
#> 9 Afghanistan AF    AFG   2002 NA    2023-07-25   360.
#> 10 Afghanistan AF    AFG   2013 NA    2023-07-25   608.
#> # i 8,768 more rows
#> # i 11 more variables: lifeExp <dbl>, pop <dbl>,
#> #   research <dbl>, military <dbl>, education <dbl>,
#> #   region <chr>, capital <chr>, longitude <dbl>,
#> #   latitude <dbl>, income <chr>, lending <chr>
```

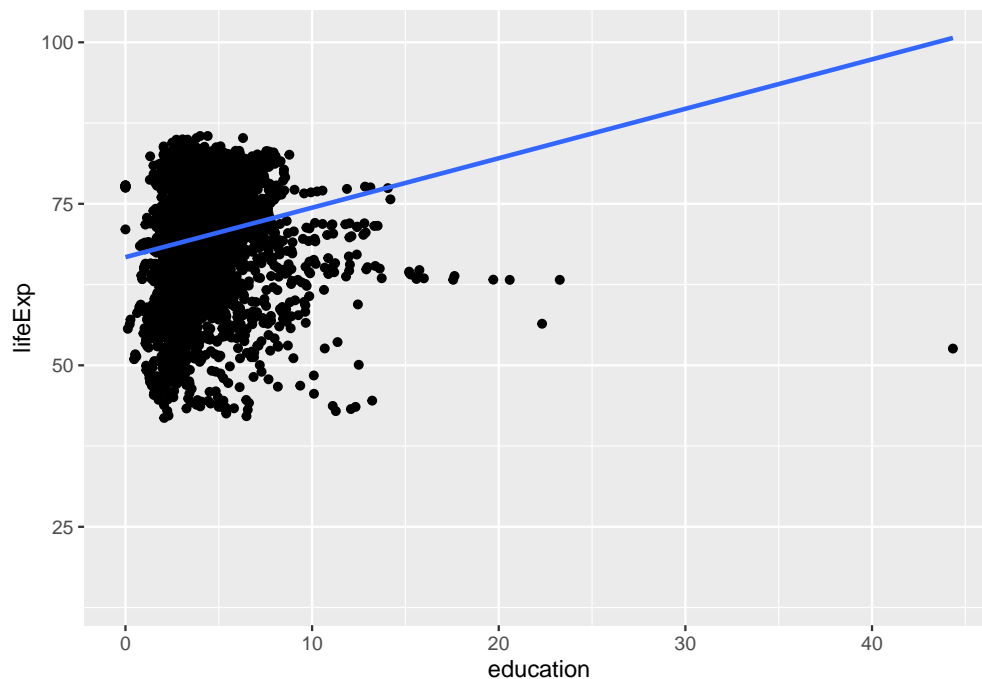
あまりデータがないものもあるようです。少しずつ調べてみましょう。

- SE.XPD.TOTL.GB.ZS: Government expenditure on education, total (% of government expenditure)
- SE.XPD.TOTL.GD.ZS: Government expenditure on education, total (% of GDP)
- SE.XPD.PRIM.PC.ZS: Government expenditure per student, primary (% of GDP per capita)

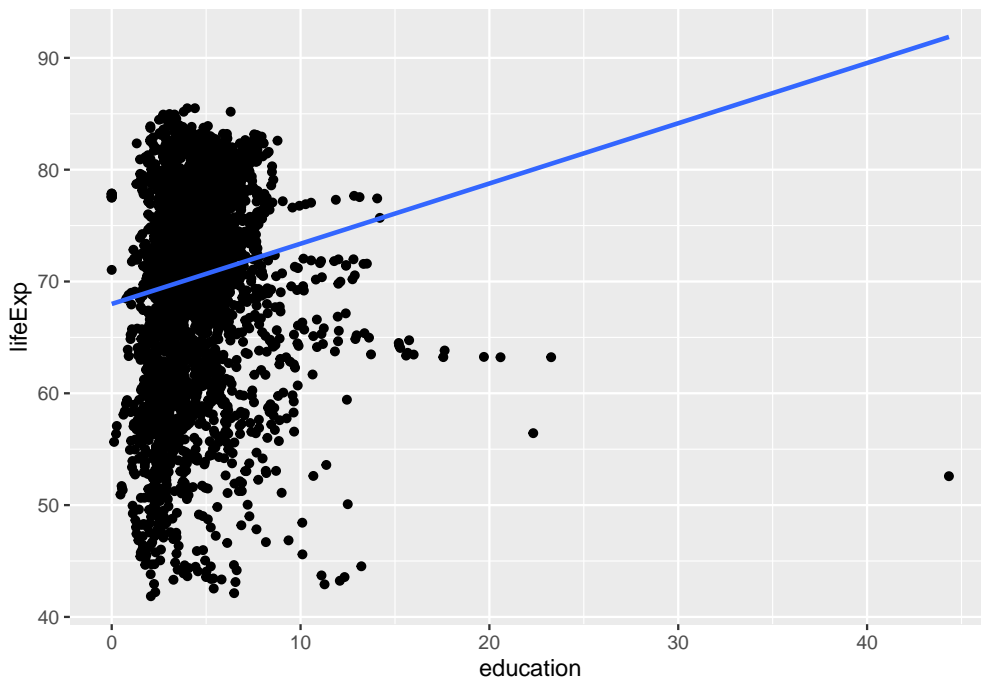
- MS.MIL.XPND.ZS: Military expenditure (% of general government expenditure)
- SE.XPD.TERT.ZS: Expenditure on tertiary education (% of government expenditure on education)

```
mod_e <- lm(lifeExp ~ education, wdi_world); mod_e
#>
#> Call:
#> lm(formula = lifeExp ~ education, data = wdi_world)
#>
#> Coefficients:
#> (Intercept)    education
#>    66.7431      0.7654
```

```
wdi_world |> ggplot(aes(education, lifeExp)) + geom_point() + geom_smooth(formula = lifeExp ~ education)
#> Warning: Removed 4030 rows containing non-finite values
#> (`stat_smooth()`).
#> Warning: Removed 4030 rows containing missing values
#> (`geom_point()`).
```

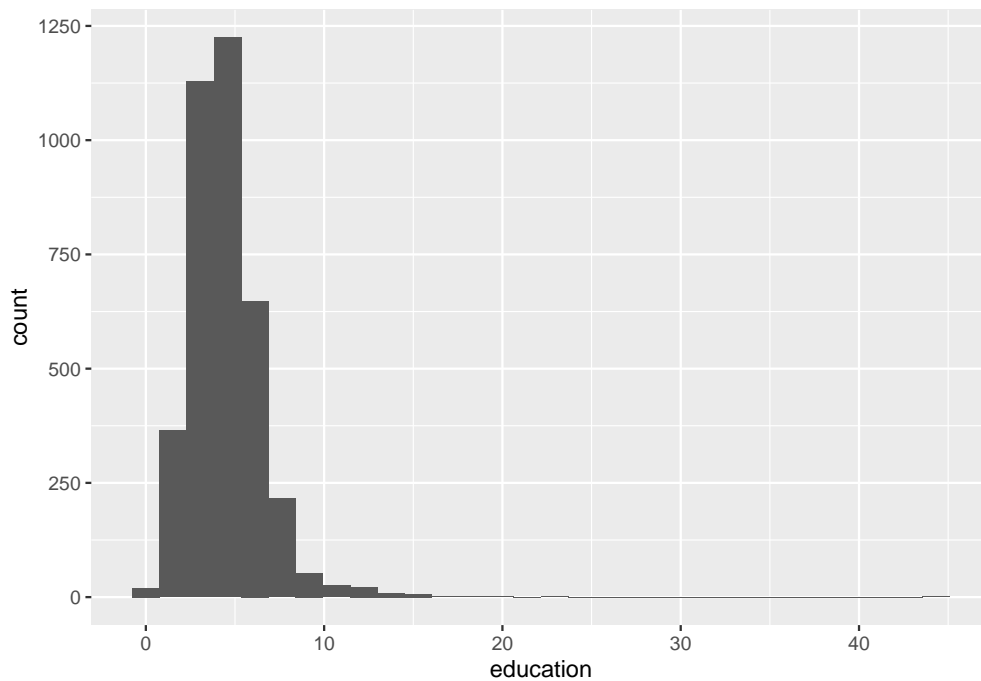


```
wdi_world %>% filter(income != "Aggregates") |> drop_na(education, lifeExp) |> ggplot(aes(education, lifeExp)) + geom_point() + geom_smooth(formula = lifeExp ~ education)
```

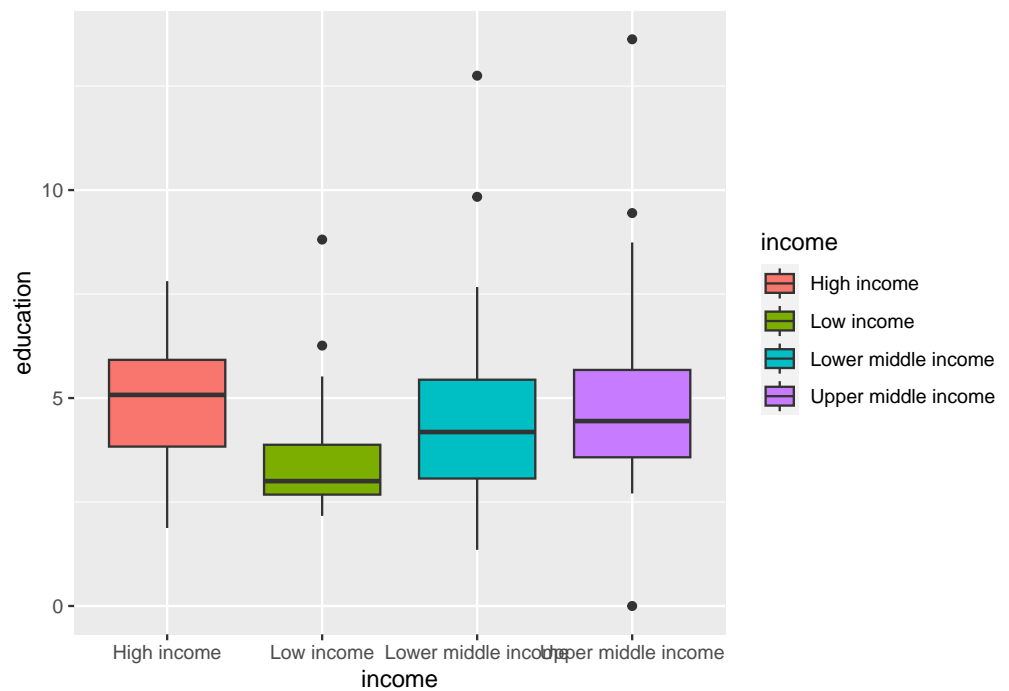


```
wdi_world_el <- wdi_world %>% select(country, year, education, lifeExp, gdpPcap, pop, research, military)
```

```
wdi_world_el |> ggplot(aes(education)) + geom_histogram()
#> `stat_bin()` using `bins = 30`. Pick better value with
#> `binwidth`.
```



```
wdi_world_el |> filter(year==2020) |> ggplot(aes(x = income, y = education, fill = income)) + geom_boxpl
```



```
wdi_world_el |> filter(year==2020) |> arrange(desc(education))
#> # A tibble: 156 x 10
#>   country   year education lifeExp gdpPcap   pop research
#>   <chr>     <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>
#> 1 Marshall~ 2020    13.6    65.0   5041. 4.34e4   NA
#> 2 Solomon ~ 2020    12.8    70.2   2080. 6.91e5   NA
#> 3 Bolivia  2020     9.84    64.5   2920. 1.19e7   NA
#> 4 Namibia  2020     9.45    62.8   4152. 2.49e6   NA
#> 5 Sierra L~ 2020     8.81    59.8    608. 8.23e6   NA
#> 6 Botswana 2020     8.74    65.6   5811. 2.55e6   NA
#> 7 Saudi Ar~ 2020     7.81    76.2  18857. 3.60e7   0.522
#> 8 Iceland  2020     7.72    83.1  52442. 3.66e5   2.47
#> 9 Lesotho  2020     7.67    54.7    969. 2.25e6   NA
#> 10 Cabo Ver~ 2020     7.58    74.8   2950. 5.83e5   NA
#> # i 146 more rows
#> # i 3 more variables: military <dbl>, region <chr>,
#> #   income <chr>
```

```
wdi_world_el |> filter(year==2020) |> arrange(desc(education))
#> # A tibble: 156 x 10
#>   country   year education lifeExp gdpPcap   pop research
#>   <chr>     <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>
#> 1 Marshall~ 2020    13.6    65.0   5041. 4.34e4   NA
#> 2 Solomon ~ 2020    12.8    70.2   2080. 6.91e5   NA
#> 3 Bolivia  2020     9.84    64.5   2920. 1.19e7   NA
```

```
#> 4 Namibia 2020 9.45 62.8 4152. 2.49e6 NA
#> 5 Sierra L~ 2020 8.81 59.8 608. 8.23e6 NA
#> 6 Botswana 2020 8.74 65.6 5811. 2.55e6 NA
#> 7 Saudi Ar~ 2020 7.81 76.2 18857. 3.60e7 0.522
#> 8 Iceland 2020 7.72 83.1 52442. 3.66e5 2.47
#> 9 Lesotho 2020 7.67 54.7 969. 2.25e6 NA
#> 10 Cabo Ver~ 2020 7.58 74.8 2950. 5.83e5 NA
#> # i 146 more rows
#> # i 3 more variables: military <dbl>, region <chr>,
#> # income <chr>
```

```
wdi_world_el |> filter(year==2020) |> lm(gdpPcap ~ education, data = _)
#>
#> Call:
#> lm(formula = gdpPcap ~ education, data = filter(wdi_world_el,
#> year == 2020))
#>
#> Coefficients:
#> (Intercept) education
#> 10568.8 940.2
```

```
wdi_world_el |> filter(year==2020) |> lm(gdpPcap ~ education, data = _) |> glance()
#> # A tibble: 1 x 12
#> r.squared adj.r.squared sigma statistic p.value df
#> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 0.00815 0.00166 20342. 1.26 0.264 1
#> # i 6 more variables: logLik <dbl>, AIC <dbl>, BIC <dbl>,
#> # deviance <dbl>, df.residual <int>, nobs <int>
```

```
wdi_world_el |> lm(lifeExp ~ education + research + military, data = _) |> glance()
#> # A tibble: 1 x 12
#> r.squared adj.r.squared sigma statistic p.value df
#> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 0.346 0.345 5.25 269. 2.26e-140 3
#> # i 6 more variables: logLik <dbl>, AIC <dbl>, BIC <dbl>,
#> # deviance <dbl>, df.residual <int>, nobs <int>
```

```
wdi_world_el |> lm(lifeExp ~ education + research + military, data = _) |> tidy()
#> # A tibble: 4 x 5
#> term estimate std.error statistic p.value
#> <chr> <dbl> <dbl> <dbl> <dbl>
#> 1 (Intercept) 70.2 0.489 143. 0
#> 2 education 0.0767 0.0966 0.794 4.27e- 1
```

```
#> 3 research      3.84      0.146      26.4 7.42e-127
#> 4 military     -0.0683     0.102     -0.669 5.04e- 1
```

$$lifeExp \sim 70.22 + 0.08 \cdot education + 3.84 \cdot research - 0.07 \cdot military$$

```
wdi_world_el |> lm(gdpPcap ~ education + research + military, data = _) |> tidy()
#> # A tibble: 4 x 5
#>   term          estimate std.error statistic  p.value
#>   <chr>          <dbl>    <dbl>    <dbl>    <dbl>
#> 1 (Intercept)    1061.    1310.     0.810 4.18e- 1
#> 2 education     1332.    259.     5.15 3.01e- 7
#> 3 research     12783.    390.    32.8 4.69e-179
#> 4 military     -964.    273.    -3.53 4.35e- 4
```

```
wdi_world_el |> lm(gdpPcap ~ education + research + military, data = _) |> glance()
#> # A tibble: 1 x 12
#>   r.squared adj.r.squared sigma statistic  p.value  df
#>   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <dbl>
#> 1  0.477      0.476 14036.    464. 3.75e-214  3
#> # i 6 more variables: logLik <dbl>, AIC <dbl>, BIC <dbl>,
#> #   deviance <dbl>, df.residual <int>, nobs <int>
```

$$gdpPcap \sim 1077 + 1024 \cdot education + 12792 \cdot research - 967 \cdot military$$

```
mod_r <- lm(lifeExp ~ research, wdi_world); mod_e
#>
#> Call:
#> lm(formula = lifeExp ~ education, data = wdi_world)
#>
#> Coefficients:
#> (Intercept)  education
#>  66.7431      0.7654
```


第 26 章

UN Data (国際連合データ)

第 27 章

Our World in Data

ここでは、R Package `owid` を用いた例をあげる

必要に応じてインストールしてください。

```
install.packages("devtools")  
devtools::install_github("piersyork/owidR")
```


第 28 章

世界不平等データベース

必要に応じてインストールしてください。

```
install.packages("devtools")
devtools::install_github("WIDworld/wid-r-tool")
```

28.1 WIR2022

データの整形について、上で学んだことについて、World Inequality Report 2022 (WIR2022) の概要 (Executive Report) のデータを使って説明します。

- 世界不平等報告 2022 (World Inequality Report) : <https://wir2022.wid.world/>
- 概要 (Executive Summary) : <https://wir2022.wid.world/executive-summary/>
- 分析方法 (Methodology) : <https://wir2022.wid.world/methodology/>
- データ (世界不平等データベース, WID) : <https://wir2022.wid.world/www-site/uploads/2022/03/WIR2022TablesFigures-Summary.xlsx>

```
library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats   1.0.0      v stringr   1.5.0
#> v ggplot2   3.4.3      v tibble    3.2.1
#> v lubridate 1.9.2      v tidyr     1.3.0
#> v purrr     1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(readxl)

url_summary <- "https://wir2022.wid.world/www-site/uploads/2022/03/WIR2022TablesFigures-Summary.xlsx"
download.file(url = url_summary, destfile = "./data/WIR2022s.xlsx", mode = "wb")
```

Excel ファイルは、CSV のようなテキストファイルではなく、デジタルファイル (binary file) ですから、ダウンロードするには `mode = "wb"` を追加する必要があります。

```
excel_sheets("./data/WIR2022s.xlsx")
#> [1] "Index"      "F1"         "F2"         "F3"
#> [5] "F4"         "F5."        "F6"         "F7"
#> [9] "F8"         "F9"         "F10"        "F11"
#> [13] "F12"        "F13"        "F14"        "F15"
#> [17] "T1"         "data-F1"    "data-F2"    "data-F3"
#> [21] "data-F4"    "data-F5"    "data-F6"    "data-F7"
#> [25] "data-F8"    "data-F9"    "data-F10"   "data-F11"
#> [29] "data-F12"   "data-F13." "data-F14." "data-F15"
```

Excel ファイルに限らず、データを読み込んだときに、`...1`, `...2`, `...3`, などが、列名に現れることがあります。これは、元々のデータには、列名がないために、R が自動的につけた名前です。

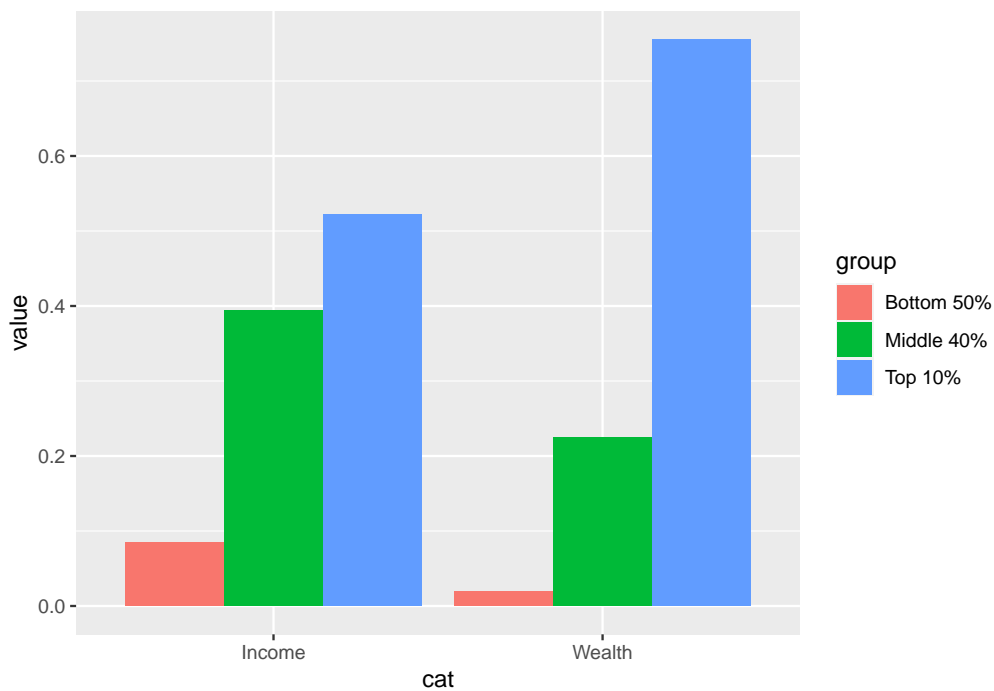
28.1.1 F1: Global income and wealth inequality, 2021

```
df_f1 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F1")
df_f1
#> # A tibble: 2 x 5
#>   ...1 `Bottom 50%` `Middle 40%` `Top 10%` `Top 1%`
#>   <chr>      <dbl>      <dbl>      <dbl>      <dbl>
#> 1 Income     0.084      0.394      0.522      0.192
#> 2 Wealth     0.0199     0.224      0.756      0.378
```

The table above is nothing terrible; however, if we have it in the following format, we can construct a chart applying the color aesthetic mapping to the group.

```
#> # A tibble: 6 x 3
#>   cat      group      value
#>   <chr>   <chr>      <dbl>
#> 1 Income Bottom 50% 0.084
#> 2 Income Middle 40% 0.394
#> 3 Income Top 10% 0.522
#> 4 Wealth Bottom 50% 0.0199
#> 5 Wealth Middle 40% 0.224
#> 6 Wealth Top 10% 0.756
```

```
df_f1_rev %>%
  ggplot(aes(x = cat, y = value, fill = group)) +
  geom_col(position = "dodge")
```



We apply the `pivot_longer` function of the `tidyr` package, to transform the first table into the second.

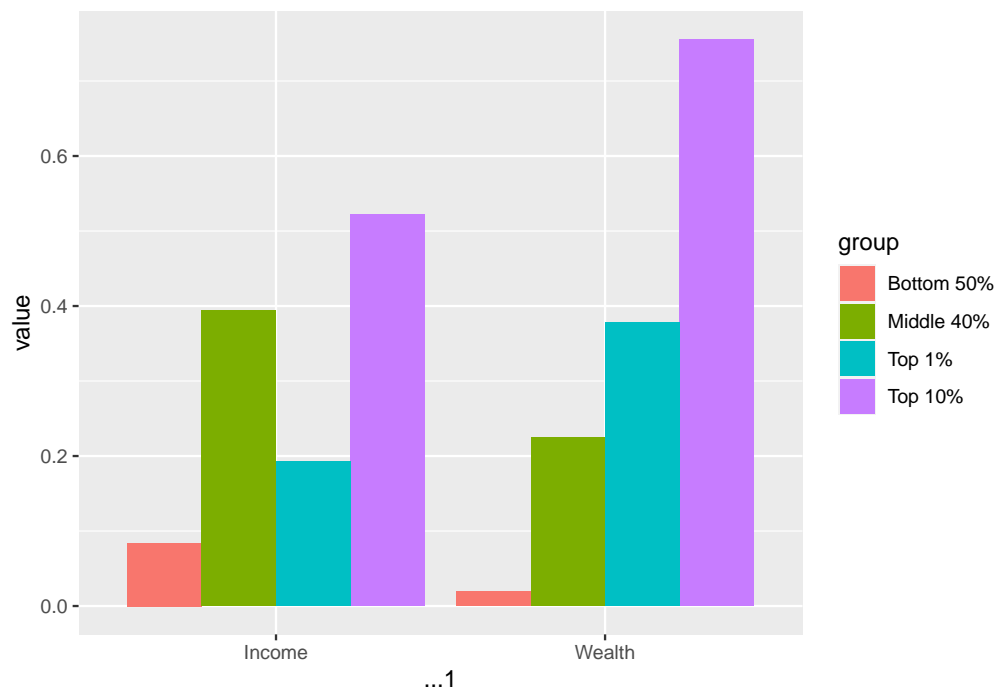
```
df_f1
#> # A tibble: 2 x 5
#>   ...1 `Bottom 50%` `Middle 40%` `Top 10%` `Top 1%`
#>   <chr>      <dbl>      <dbl>      <dbl>      <dbl>
#> 1 Income      0.084      0.394      0.522      0.192
#> 2 Wealth      0.0199     0.224      0.756      0.378

(df_f1_rev <- df_f1 %>% pivot_longer(-1, names_to = "group", values_to = "value"))
#> # A tibble: 8 x 3
#>   ...1 group      value
#>   <chr> <chr>      <dbl>
#> 1 Income Bottom 50% 0.084
#> 2 Income Middle 40% 0.394
#> 3 Income Top 10%  0.522
#> 4 Income Top 1%   0.192
#> 5 Wealth Bottom 50% 0.0199
#> 6 Wealth Middle 40% 0.224
#> 7 Wealth Top 10%  0.756
#> 8 Wealth Top 1%   0.378
```

In the example above, `-1`, i.e., `cols = -1` stands for all columns except the first.

Now, we can use the `fill` aesthetic in addition to `position = "dodge"`. The default position is "stack".

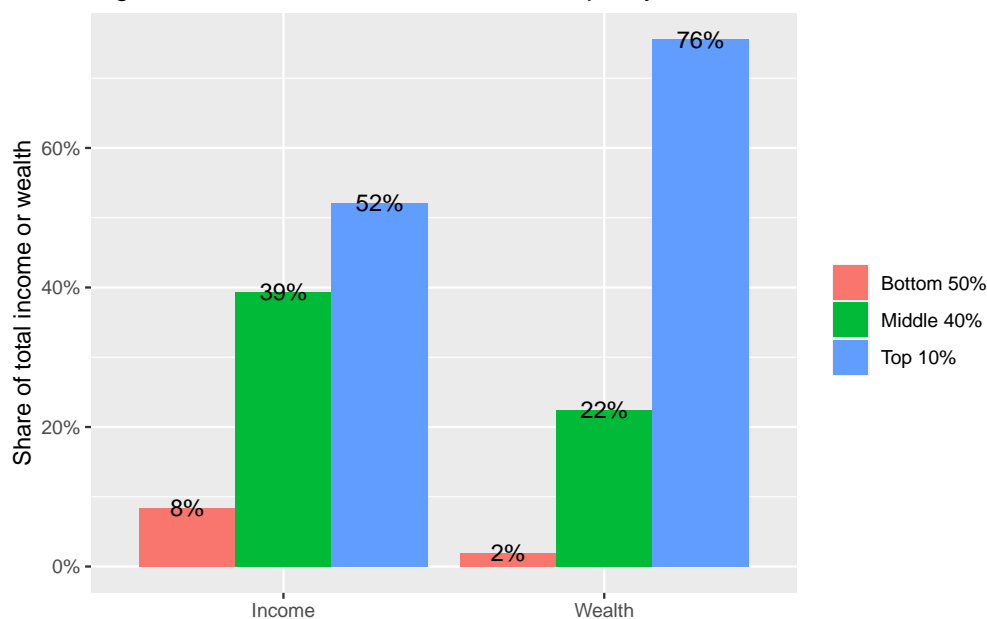
```
df_f1_rev %>%
  ggplot(aes(x = ...1, y = value, fill = group)) +
  geom_col(position = "dodge")
```



Let us add the value as a label, change the y-axis to percent, and add the title. The interpretation and source are from the original

```
df_f1_rev %>% filter(group != "Top 1%") %>%
  ggplot() +
  geom_col(aes(x = ...1, y = value, fill = group), position = "dodge") +
  geom_text(aes(x = ...1, y = value, group = group,
               label = scales::label_percent(accuracy=1)(value)),
            position = position_dodge(width = 0.9)) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  labs(title = "Figure 1. Global income and wealth inequality, 2021",
       x = "", y = "Share of total income or wealth", fill = "")
```


Figure 1. Global income and wealth inequality, 2021



Interpretation: The global bottom 50% captures 8.5% of total income measured at Purchasing Power Parity (PPP). The global bottom 50% owns 2% of wealth (at Purchasing Power Parity). The global top 10% owns 76% of total Household wealth and captures 52% of total income in 2021. Note that top wealth holders are not necessarily top income holders. Incomes are measured after the operation of pension and unemployment systems and before taxes and transfers.

Sources and series: wir2022.wid.world/methodology.

The next F2 is similar to F1.

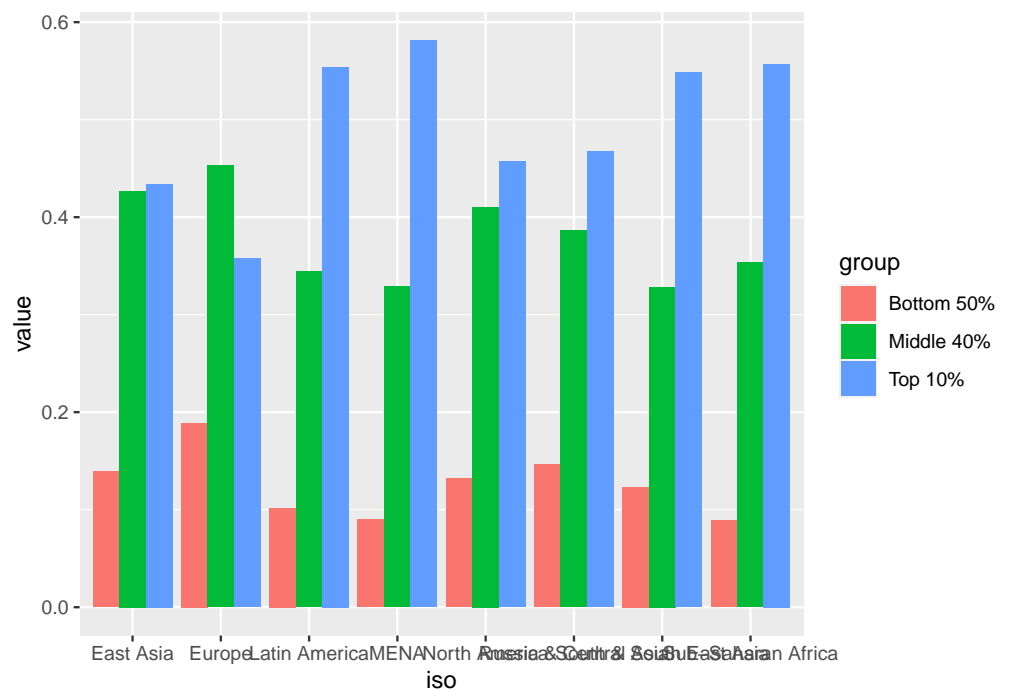
28.2 F2: The poorest half lags behind: Bottom 50%, middle 40% and top 10% income shares across the world in 2021

```
df_f2 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F2")
df_f2
#> # A tibble: 8 x 5
#>   year iso          `Bottom 50%` `Middle 40%` `Top 10%`
#>   <dbl> <chr>          <dbl>      <dbl>      <dbl>
#> 1  2021 Europe          0.189      0.453      0.358
#> 2  2021 East Asia      0.139      0.427      0.434
#> 3  2021 North America  0.132      0.411      0.457
#> 4  2021 Russia & Centra~ 0.147      0.386      0.467
#> 5  2021 South & South E~ 0.123      0.328      0.548
#> 6  2021 Latin America  0.102      0.344      0.554
```

```
#> 7 2021 Sub-Saharan Afr~ 0.0892 0.354 0.557
#> 8 2021 MENA 0.09 0.329 0.581
```

```
df_f2 %>% pivot_longer(cols = 3:5, names_to = "group", values_to = "value")
#> # A tibble: 24 x 4
#>   year iso group value
#>   <dbl> <chr> <chr> <dbl>
#> 1 2021 Europe Bottom 50% 0.189
#> 2 2021 Europe Middle 40% 0.453
#> 3 2021 Europe Top 10% 0.358
#> 4 2021 East Asia Bottom 50% 0.139
#> 5 2021 East Asia Middle 40% 0.427
#> 6 2021 East Asia Top 10% 0.434
#> 7 2021 North America Bottom 50% 0.132
#> 8 2021 North America Middle 40% 0.411
#> 9 2021 North America Top 10% 0.457
#> 10 2021 Russia & Central Asia Bottom 50% 0.147
#> # i 14 more rows
```

```
df_f2 %>% pivot_longer(cols = 3:5, names_to = "group", values_to = "value") %>%
  ggplot(aes(x = iso, y = value, fill = group)) +
  geom_col(position = "dodge")
```



28.3 Pivot data from long to wide:

`pivot_wider()` In Console: vignette("pivot")

```
pivot_wider(data,
  names_from = <name of the column (or columns) to get the name of the output column>,
  values_from = <name of the column to get the value of the output>)
```

```
#> # A tibble: 24 x 4
#>   year iso          group      value
#>   <dbl> <chr>          <chr>    <dbl>
#> 1  2021 Europe      Bottom 50% 0.189
#> 2  2021 Europe      Middle 40% 0.453
#> 3  2021 Europe      Top 10%    0.358
#> 4  2021 East Asia   Bottom 50% 0.139
#> 5  2021 East Asia   Middle 40% 0.427
#> 6  2021 East Asia   Top 10%    0.434
#> 7  2021 North America Bottom 50% 0.132
#> 8  2021 North America Middle 40% 0.411
#> 9  2021 North America Top 10%    0.457
#> 10 2021 Russia & Central Asia Bottom 50% 0.147
#> # i 14 more rows
```

```
pivot_wider(data, names_from = group, values_from = value)
```

28.4 Practice: F4 and F13

F4 and F13 are similar. Please use `pivot_longer` to tidy the data and create charts.

- **References:** <https://ds-sl.github.io/data-analysis/wir2022.nb.html>

28.4.1 Done Last Week

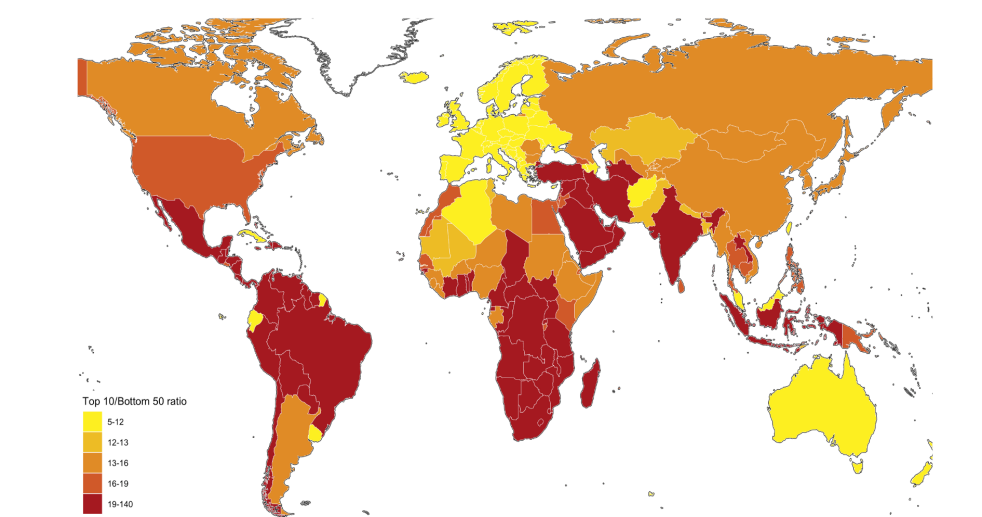
- F12: Female share in global labor incomes, 1990-2020
- F14: Global carbon inequality, 2019. Group contribution to world emissions (%)

The next is an example of the world map.

28.5 F3: Top 10/Bottom 50 income gaps across the world, 2021

```
df_f3 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F3")
df_f3
#> # A tibble: 177 x 3
#>   year Country      T10B50
#>   <dbl> <chr>      <dbl>
#> 1  2021 United Arab Emirates  19.2
#> 2  2021 Afghanistan          11.7
#> 3  2021 Albania              8.99
#> 4  2021 Armenia              11.0
#> 5  2021 Angola               32.1
#> 6  2021 Argentina           13.2
#> 7  2021 Austria              7.68
#> 8  2021 Australia            10.4
#> 9  2021 Azerbaijan           9.63
#> 10 2021 Bosnia and Herzegovina  9.32
#> # i 167 more rows
```

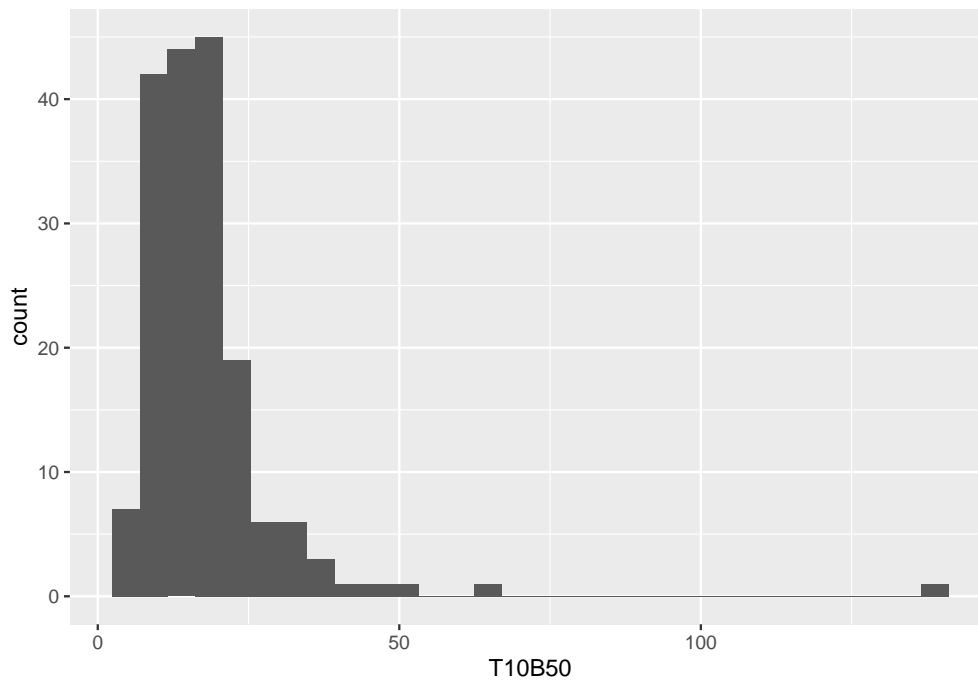
28.6 F3: Top 10/Bottom 50 income gaps across the world, 2021 - Original



- To 10 / Bottom 50 ratio has 5 classes: 5-12, 12-13, 13-16, 16-19, 19-140
- Let us look at the range and distribution of the values in T10B50.

```
df_f3$T10B50 %>% summary()
#>   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>  5.394 10.958 15.676 17.635 19.838 139.591
```

```
df_f3 %>% ggplot() + geom_histogram(aes(T10B50))
#> `stat_bin()` using `bins = 30`. Pick better value with
#> `binwidth`.
```



```
df_f3 %>% arrange(desc(T10B50))
#> # A tibble: 177 x 3
#>   year Country      T10B50
#>   <dbl> <chr>         <dbl>
#> 1  2021 Oman          140.
#> 2  2021 South Africa  63.1
#> 3  2021 Namibia       49.0
#> 4  2021 Zambia        44.4
#> 5  2021 Central African Republic 42.5
#> 6  2021 Mozambique    38.9
#> 7  2021 Swaziland     38.1
#> 8  2021 Botswana      36.5
#> 9  2021 Angola        32.1
#> 10 2021 Yemen         32.0
#> # i 167 more rows
```

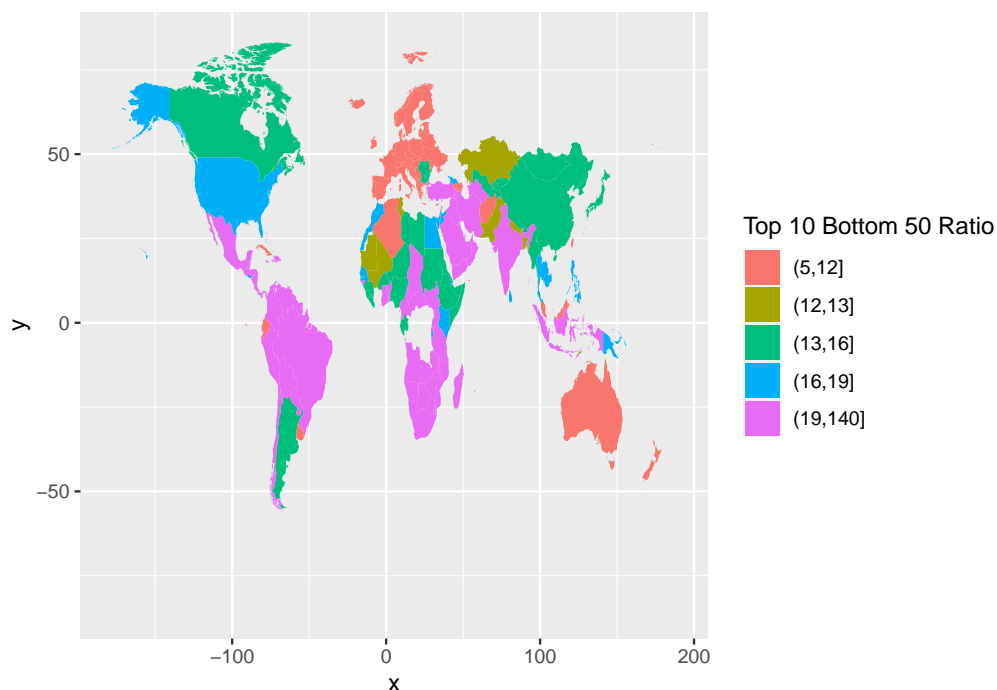
Using the information above, we set breakpoints and use R Base's `cut` command to divide into five classes, and add it as a new column using `mutate`

```
df_f3 %>%
  mutate(`Top 10 Bottom 50 Ratio` = cut(T10B50,breaks = c(5, 12, 13, 16, 19,140),
                                         include.lowest = FALSE))

#> # A tibble: 177 x 4
#>   year Country          T10B50 Top 10 Bottom 50 Rat~1
#>   <dbl> <chr>          <dbl> <fct>
#> 1  2021 United Arab Emirates  19.2 (19,140]
#> 2  2021 Afghanistan          11.7 (5,12]
#> 3  2021 Albania              8.99 (5,12]
#> 4  2021 Armenia              11.0 (5,12]
#> 5  2021 Angola              32.1 (19,140]
#> 6  2021 Argentina           13.2 (13,16]
#> 7  2021 Austria              7.68 (5,12]
#> 8  2021 Australia           10.4 (5,12]
#> 9  2021 Azerbaijan          9.63 (5,12]
#> 10 2021 Bosnia and Herzegovi~ 9.32 (5,12]
#> # i 167 more rows
#> # i abbreviated name: 1: `Top 10 Bottom 50 Ratio`
```

```
world_map <- map_data("world")
df_f3 %>% mutate(`Top 10 Bottom 50 Ratio` = cut(T10B50,breaks = c(5, 12, 13, 16,
                                                                include.lowest = FALSE)) %>%

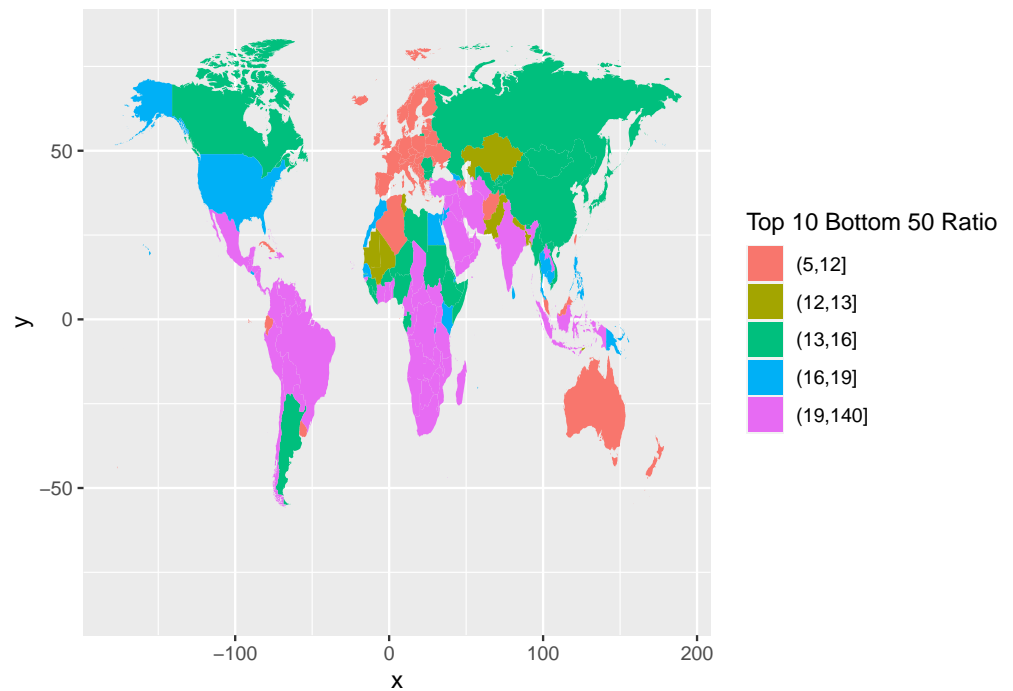
ggplot(aes(map_id = Country)) +
  geom_map(aes(fill = `Top 10 Bottom 50 Ratio`), map = world_map) +
  expand_limits(x = world_map$long, y = world_map$lat)
```



We observe that we have missing data from several countries. One common problem is the description of the country names varies in different data; in this case, the country names of `map_data()` and those of `wir2022`. There are several ways to edit country names. Here is one of them.

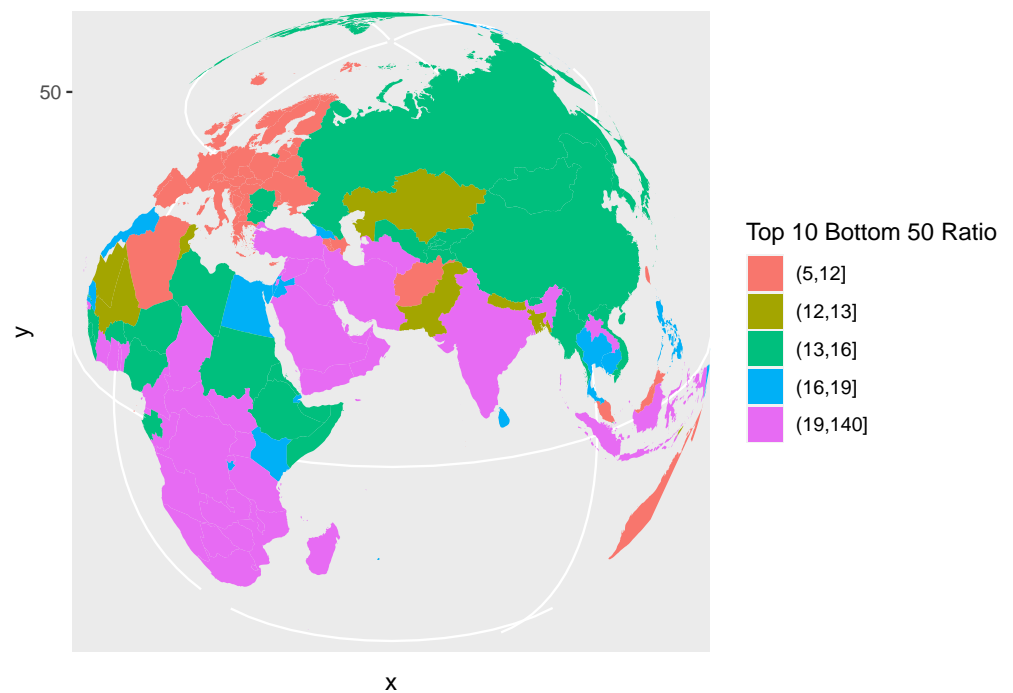
```
world_map_wir <- world_map
world_map_wir$region[
  world_map_wir$region=="Democratic Republic of the Congo"]<-"DR Congo"
world_map_wir$region[world_map_wir$region=="Republic of Congo"]<-"Congo"
world_map_wir$region[world_map_wir$region=="Ivory Coast"]<-"Cote d'Ivoire"
world_map_wir$region[world_map_wir$region=="Vietnam"]<-"Viet Nam"
world_map_wir$region[world_map_wir$region=="Russia"]<-"Russian Federation"
world_map_wir$region[world_map_wir$region=="South Korea"]<-"Korea"
world_map_wir$region[world_map_wir$region=="UK"]<-"United Kingdom"
world_map_wir$region[world_map_wir$region=="Brunei"]<-"Brunei Darussalam"
world_map_wir$region[world_map_wir$region=="Laos"]<-"Lao PDR"
world_map_wir$region[world_map_wir$region=="Cote d'Ivoire"]<-"Cote d'Ivoire"
world_map_wir$region[world_map_wir$region=="Cape Verde"]<- "Cabo Verde"
world_map_wir$region[world_map_wir$region=="Syria"]<- "Syrian Arab Republic"
world_map_wir$region[world_map_wir$region=="Trinidad"]<- "Trinidad and Tobago"
world_map_wir$region[world_map_wir$region=="Tobago"]<- "Trinidad and Tobago"

df_f3 %>% mutate(`Top 10 Bottom 50 Ratio` =
  cut(T10B50, breaks = c(5, 12, 13, 16, 19,140), include.lowest = FALSE)) %>%
  ggplot(aes(map_id = Country)) +
  geom_map(aes(fill = `Top 10 Bottom 50 Ratio`),
    map = world_map_wir) +
  expand_limits(x = world_map_wir$long, y = world_map_wir$lat)
```

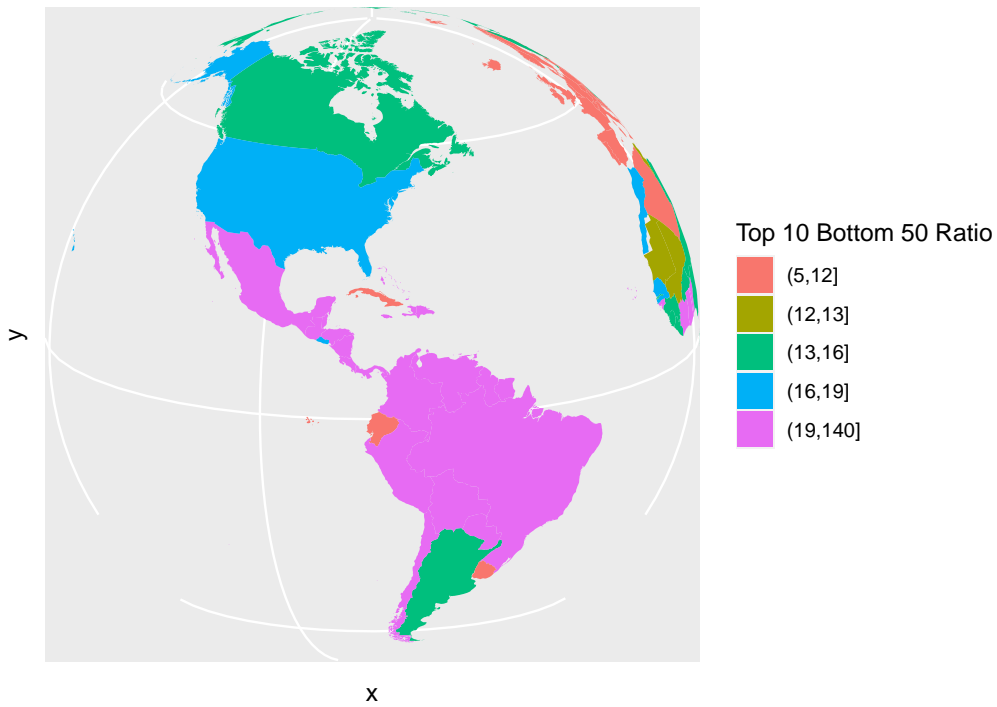


Now it is much better.

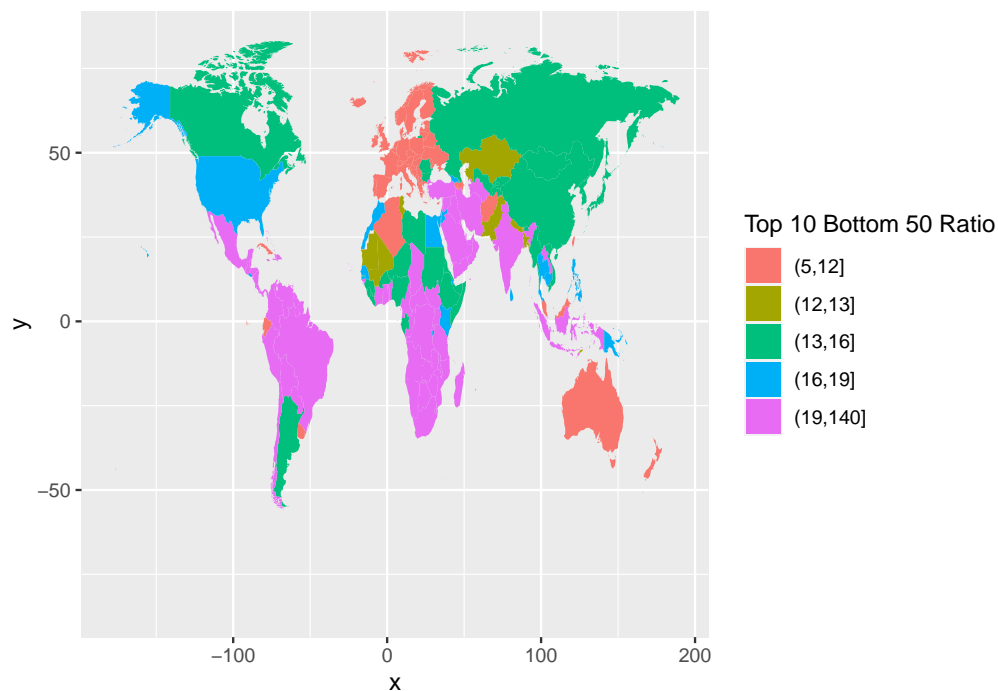
```
df_f3 %>% mutate(`Top 10 Bottom 50 Ratio` =
  cut(T10B50,breaks = c(5, 12, 13, 16, 19,140), include.lowest = FALSE)) %>%
  ggplot(aes(map_id = Country)) + geom_map(aes(fill = `Top 10 Bottom 50 Ratio`),
    map = world_map_wir) + expand_limits(x = world_map_wir$long, y = world_map_wir$lat) +
  coord_map("orthographic", orientation = c(25, 60, 0))
```




```
df_f3 %>% mutate(`Top 10 Bottom 50 Ratio` =
  cut(T10B50,breaks = c(5, 12, 13, 16, 19,140), include.lowest = FALSE)) %>%
  ggplot(aes(map_id = Country)) + geom_map(aes(fill = `Top 10 Bottom 50 Ratio`),
    map = world_map_wir) + expand_limits(x = world_map_wir$long, y = world_map_wir$lat) +
  coord_map("orthographic", orientation = c(15, -80, 0))
```



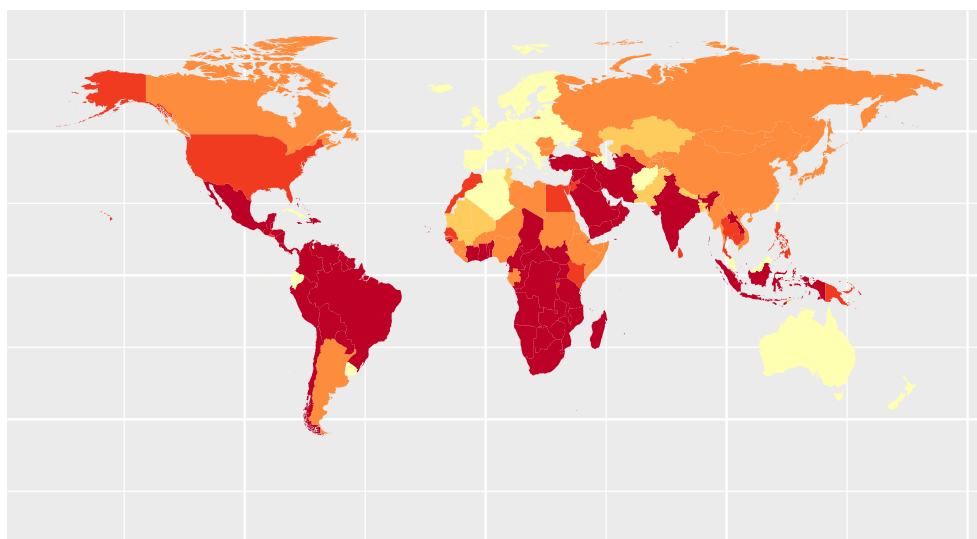
```
df_f3 %>% mutate(`Top 10 Bottom 50 Ratio` =
  cut(T10B50,breaks = c(5, 12, 13, 16, 19,140), include.lowest = FALSE)) %>%
  ggplot(aes(map_id = Country)) + geom_map(aes(fill = `Top 10 Bottom 50 Ratio`),
    map = world_map_wir) +
  expand_limits(x = world_map_wir$long, y = world_map_wir$lat)
```



Finally, change colors and change labels.

```
df_f3 %>%
  mutate(`Top 10 Bottom 50 Ratio` =
    cut(T10B50,breaks = c(5, 12, 13, 16, 19,140), include.lowest = FALSE)) %>%
  ggplot(aes(map_id = Country)) +
  geom_map(aes(fill = `Top 10 Bottom 50 Ratio`), map = world_map_wir) +
  expand_limits(x = world_map_wir$long, y = world_map_wir$lat) +
  labs(title = "Figure 3. Top 10/Bottom 50 income gaps across the world, 2021",
    x = "", y = "", fill = "Top 10/Bottom 50 ratio") +
  theme(legend.position="bottom",
    axis.text.x=element_blank(), axis.ticks.x=element_blank(),
    axis.text.y=element_blank(), axis.ticks.y=element_blank()) +
  scale_fill_brewer(palette='YlOrRd')
```

Figure 3. Top 10/Bottom 50 income gaps across the world, 2021



Top 10/Bottom 50 ratio (5,12] (12,13] (13,16] (16,19] (19,140]

We could not treat the data of three. We can check by using `anti_join`.

```
df_f3 %>% anti_join(world_map_wir, by = c("Country" = "region"))
#> # A tibble: 3 x 3
#>   year Country   T10B50
#>   <dbl> <chr>     <dbl>
#> 1  2021 Hong Kong  17.7
#> 2  2021 Macao    14.5
#> 3  2021 Zanzibar  19.8
```

Filtering joins

- `anti_join(x,y, ...)`: return all rows from x without a match in y.
- `semi_join(x,y, ...)`: return all rows from x with a match in y.

Check `dplyr` cheat sheet, and Posit Primers Tidy Data.

28.7 Remaining Charts

- F5: Global income inequality: T10/B50 ratio, 1820-2020 - fit curve
- F9: Average annual wealth growth rate, 1995-2021 - fit curve + alpha
- F7: Global income inequality, 1820-2020 - pivot + fit curve
- F10: The share of wealth owned by the global 0.1% and billionaires, 2021 - pivot + fit curve

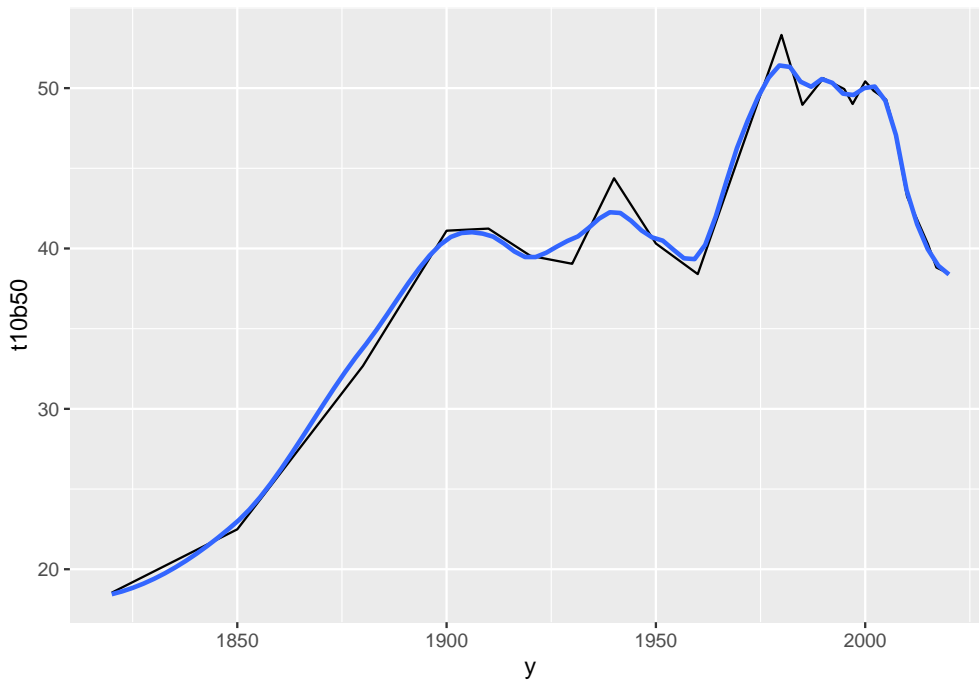
- F6: Global income inequality: Between vs. Within country inequality (Theil index), 1820-2020 - pivot + area
- F11: Top 1% vs bottom 50% wealth shares in Western Europe and the US, 1910-2020 - pivot name_sep + fit curve
- F8: The rise of private versus the decline of public wealth in rich countries, 1970-2020 - rename + pivot + pivot + fit curve
- F15: Per capita emissions across the world, 2019 - add row names + dodge

We will discuss `geom_smooth` and `stat_smooth` in Chapter ?? applied to F5, F9, F7, F10.

28.8 F5: Global income inequality: T10/B50 ratio, 1820-2020

```
(df_f5 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F5"))
#> # A tibble: 24 x 2
#>       y t10b50
#>   <dbl> <dbl>
#> 1  1820  18.5
#> 2  1850  22.5
#> 3  1880  32.7
#> 4  1900  41.1
#> 5  1910  41.2
#> 6  1920  39.5
#> 7  1930  39.0
#> 8  1940  44.4
#> 9  1950  40.3
#> 10 1960  38.4
#> # i 14 more rows
```

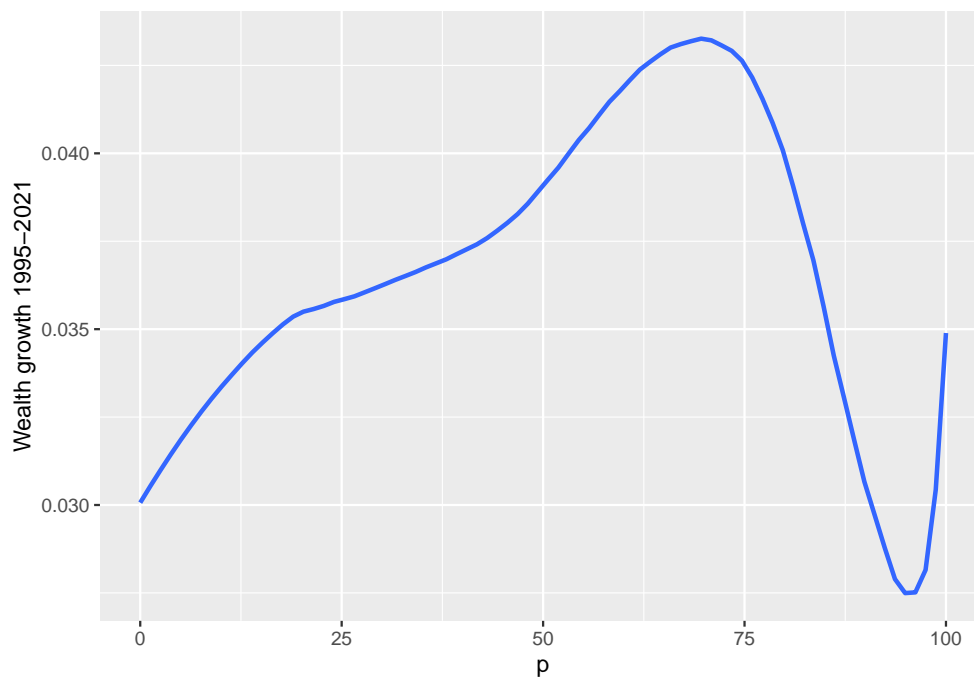
```
df_f5 %>% ggplot(aes(x = y, y = t10b50)) + geom_line() + geom_smooth(span=0.25, s
#> `geom_smooth()` using method = 'loess' and formula = 'y ~
#> x'
```



28.9 F9: Average annual wealth growth rate, 1995-2021 - fit curve + alpha

```
df_f9 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F9"); df_f9
#> # A tibble: 127 x 2
#>       p `Wealth growth 1995-2021`
#>   <dbl>           <dbl>
#> 1     0           0.0310
#> 2     1           0.0310
#> 3     2           0.0310
#> 4     3           0.0310
#> 5     4           0.0310
#> 6     5           0.0310
#> 7     6           0.0312
#> 8     7           0.0317
#> 9     8           0.0322
#> 10    9           0.0328
#> # i 117 more rows
```

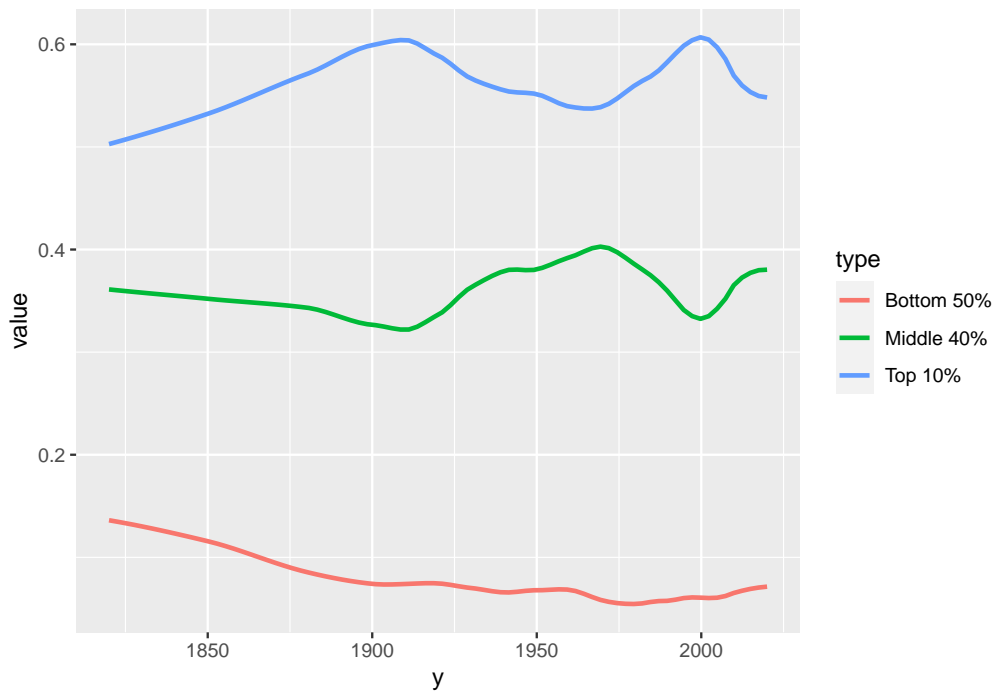
```
df_f9 %>%
  ggplot(aes(x = p, y = `Wealth growth 1995-2021`)) + geom_smooth(span = 0.30, se = FALSE)
#> `geom_smooth()` using method = 'loess' and formula = 'y ~
#> x'
```



28.10 F7: Global income inequality, 1820-2020 - pivot + fit curve

```
df_f7 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F7"); df_f7
#> # A tibble: 24 x 4
#>       y `Bottom 50%` `Middle 40%` `Top 10%`
#>   <dbl>     <dbl>     <dbl> <dbl>
#> 1  1820     0.136     0.361  0.503
#> 2  1850     0.118     0.350  0.532
#> 3  1880     0.0870    0.345  0.568
#> 4  1900     0.0724    0.332  0.595
#> 5  1910     0.0729    0.326  0.601
#> 6  1920     0.0755    0.328  0.597
#> 7  1930     0.0714    0.371  0.558
#> 8  1940     0.0629    0.379  0.558
#> 9  1950     0.0687    0.377  0.554
#> 10 1960     0.0701    0.392  0.538
#> # i 14 more rows
```

```
df_f7 %>%
  pivot_longer(cols = 2:4, names_to = "type", values_to = "value") %>%
  ggplot(aes(x = y, y = value, color = type)) +
  stat_smooth(formula = y~x, method = "loess", span = 0.25, se = FALSE)
```

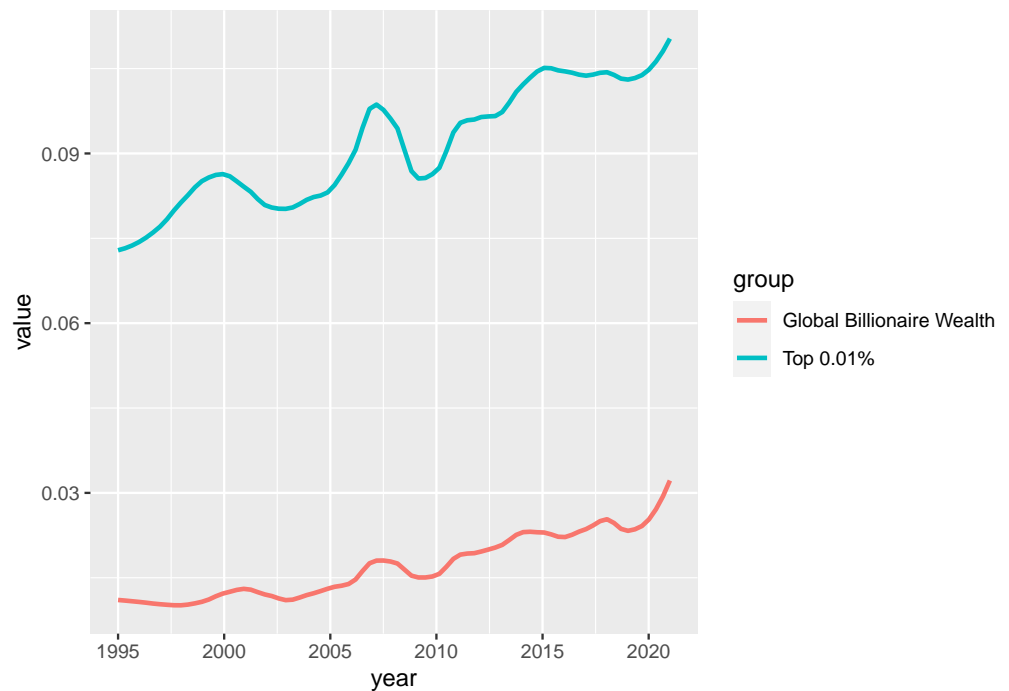


28.11 F10: The share of wealth owned by the global 0.1% and billionaires, 2021 - pivot + fit curve

```
df_f10 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F10"); df_f10
#> New names:
#> * `` -> `...4`
#> * `` -> `...5`
#> * `` -> `...6`
#> # A tibble: 27 x 6
#>   year bn_hhweal top0.1_hhweal ...4 ...5 ...6
#>   <dbl>   <dbl>         <dbl> <lgl> <dbl> <dbl>
#> 1  1995  0.0108         0.0729 NA     NA     NA
#> 2  1996  0.0114         0.0744 NA     NA     NA
#> 3  1997  0.0101         0.0768 NA     NA     NA
#> 4  1998  0.00995        0.0815 NA     NA     NA
#> 5  1999  0.0112         0.0855 NA     NA     NA
#> 6  2000  0.0115         0.0862 NA     NA     NA
#> 7  2001  0.0139         0.0846 NA     NA     NA
#> 8  2002  0.0119         0.08   NA     NA     NA
#> 9  2003  0.0103         0.08   NA     NA     NA
#> 10 2004  0.0127         0.0828 NA     NA     NA
#> # i 17 more rows
```

```
df_f10 %>%
  select(year, "Global Billionaire Wealth" = bn_hhweal, "Top 0.01%" = top0.1_hhweal)
  pivot_longer(!year, names_to = "group", values_to = "value")
#> # A tibble: 54 x 3
#>   year group          value
#>   <dbl> <chr>          <dbl>
#> 1 1995 Global Billionaire Wealth 0.0108
#> 2 1995 Top 0.01%          0.0729
#> 3 1996 Global Billionaire Wealth 0.0114
#> 4 1996 Top 0.01%          0.0744
#> 5 1997 Global Billionaire Wealth 0.0101
#> 6 1997 Top 0.01%          0.0768
#> 7 1998 Global Billionaire Wealth 0.00995
#> 8 1998 Top 0.01%          0.0815
#> 9 1999 Global Billionaire Wealth 0.0112
#> 10 1999 Top 0.01%          0.0855
#> # i 44 more rows
```

```
df_f10 %>%
  select(year, "Global Billionaire Wealth" = bn_hhweal, "Top 0.01%" = top0.1_hhweal)
  pivot_longer(!year, names_to = "group", values_to = "value") %>%
  ggplot() +
  stat_smooth(aes(x = year, y = value, color = group), formula = y~x, method = "lm")
```



28.12 F6: Global income inequality: Between vs. Within country inequality (Theil index), 1820-2020 - pivot + area

```
df_f6 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F6"); df_f6
#> New names:
#> * ` ` -> `...1`
#> # A tibble: 9 x 3
#>   ...1 `Between-country inequality` `Within-country inequa-1
#>   <dbl>                <dbl>                <dbl>
#> 1  1820                0.120                0.880
#> 2  1850                0.166                0.834
#> 3  1880                0.241                0.759
#> 4  1900                0.257                0.743
#> 5  1920                0.320                0.680
#> 6  1950                0.439                0.561
#> 7  1980                0.569                0.431
#> 8  2000                0.473                0.527
#> 9  2020                0.320                0.680
#> # i abbreviated name: 1: `Within-country inequality`
```

```
df_f6 %>% select(year = "...1", 2:3) %>%
  pivot_longer(cols = 2:3, names_to = "type", values_to = "value") %>%
  mutate(types = factor(type,
    levels = c("Within-country inequality", "Between-country inequality"))) %>%
  ggplot(aes(x = year, y = value, fill = types)) +
  geom_area() +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  scale_x_continuous(breaks = round(seq(1820, 2020, by = 20), 1)) +
  scale_fill_manual(values=rev(scales::hue_pal()(2)),
    labels = function(x) str_wrap(x, width = 15)) +
  labs(title = "Figure 6. Global income inequality:
  \nBetween vs. within country inequality (Theil index), 1820-2020",
    x = "", y = "Share of global inequality (% of total Theil index)", fill = "") +
  annotate("text", x = 1850, y = 0.28,
    label = stringr::str_wrap("1820: Between country inequality represents 11%
    of global inequality", width = 20), size = 3) +
  annotate("text", x = 1980, y = 0.70,
    label = stringr::str_wrap("1980: Between country inequality represents 57%
    of global inequality", width = 20), size = 3) +
```

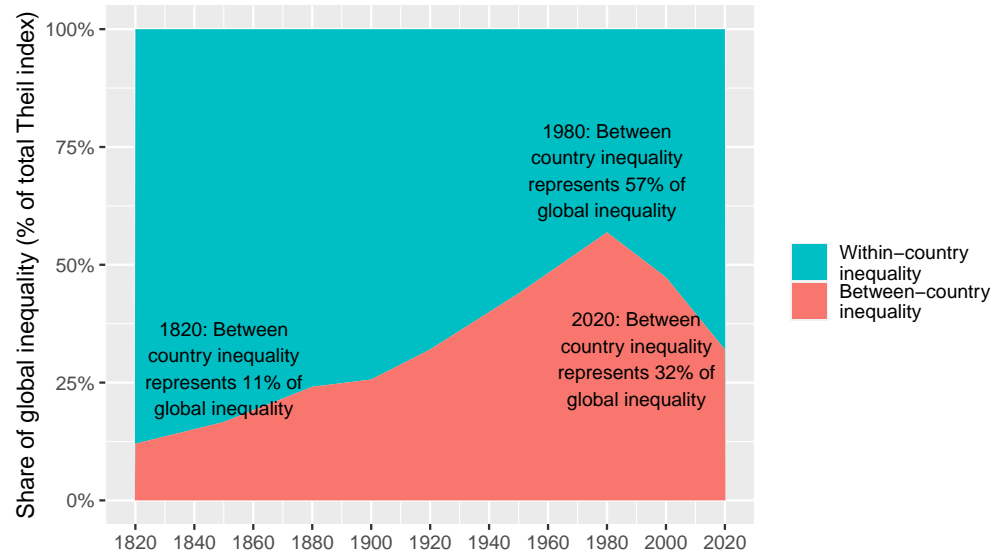
```

annotate("text", x = 1990, y = 0.30,
  label = stringr::str_wrap("2020: Between country inequality represents 32%
    of global inequality", width = 20), size = 3)

```

Figure 6. Global income inequality:

Between vs. within country inequality (Theil index), 1820–2020



28.13 F11: Top 1% vs bottom 50% wealth shares in Western Europe and the US, 1910-2020 - pivot name_sep + fit curve

```

df_f11 <- read_excel("../data/WIR2022s.xlsx", sheet = "data-F11"); df_f11
#> # A tibble: 12 x 5
#>   year USbot50 UStop1 EUbot50 EUtop1
#>   <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 1910 0.00700 0.425 0.0128 0.554
#> 2 1920 0.0102 0.410 0.0139 0.496
#> 3 1930 0.00737 0.409 0.0175 0.464
#> 4 1940 0.0112 0.319 0.0282 0.412
#> 5 1950 0.0270 0.276 0.0272 0.339
#> 6 1960 0.0232 0.279 0.0503 0.304
#> 7 1970 0.0221 0.241 0.0649 0.235
#> 8 1980 0.0260 0.251 0.0658 0.200
#> 9 1990 0.0211 0.294 0.0535 0.200
#> 10 2000 0.0162 0.323 0.0543 0.214
#> 11 2010 0.0111 0.357 0.0500 0.219
#> 12 2020 0.0149 0.354 0.0576 0.219

```

We want to separate 'US', 'EU' and 'bot50', 'top10', and 'top1'. To apply names_sep = "_", we first changed the name of columns.

```
df_f11 %>%
  rename(!year, US_bot50 = USbot50, US_top1 = UStop1,
         EU_bot50 = EUbot50, EU_top1 = EUtop1) %>%
  pivot_longer(!year, names_to = c("group", ".value"), names_sep = "_") %>%
  pivot_longer(3:4, names_to = "type", values_to = "value") %>%
  ggplot() +
  stat_smooth(aes(x = year, y = value, color = group, linetype = type),
             span = 0.25, se = FALSE) +
  scale_x_continuous(breaks = round(seq(1910, 2020, by = 10), 1)) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  labs(title = "Figure 11. Top 1% vs bottom 50% wealth shares
         \n in Western Europe and the US, 1910-2020",
       x = "", y = "Share of total personal wealth (%)", color = "", linetype = "") +
  scale_linetype_manual(values = c("dotted", "solid")) +
  annotate("text", x = 2000, y = 0.50,
         label = stringr::str_wrap("Wealth inequality has been rising at
         different speeds after a historical decline. The bottom 50% has always been
         extremely low.", width = 30), size = 3)
```

28.13.1 Step 1.

```
df_f11 %>% rename(!year, US_bot50 = USbot50, US_top1 = UStop1,
                 EU_bot50 = EUbot50, EU_top1 = EUtop1)
#> # A tibble: 12 x 5
#>   year US_bot50 US_top1 EU_bot50 EU_top1
#>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
#> 1  1910  0.00700  0.425  0.0128  0.554
#> 2  1920  0.0102  0.410  0.0139  0.496
#> 3  1930  0.00737  0.409  0.0175  0.464
#> 4  1940  0.0112  0.319  0.0282  0.412
#> 5  1950  0.0270  0.276  0.0272  0.339
#> 6  1960  0.0232  0.279  0.0503  0.304
#> 7  1970  0.0221  0.241  0.0649  0.235
#> 8  1980  0.0260  0.251  0.0658  0.200
#> 9  1990  0.0211  0.294  0.0535  0.200
#> 10 2000  0.0162  0.323  0.0543  0.214
#> 11 2010  0.0111  0.357  0.0500  0.219
#> 12 2020  0.0149  0.354  0.0576  0.219
```

28.13.2 Step 2.

```
df_f11 %>%
  rename(!year, US_bot50 = USbot50, US_top1 = UStop1,
         EU_bot50 = EUbot50, EU_top1 = EUtop1) %>%
  pivot_longer(!year, names_to = c("group", ".value"), names_sep = "_")
```

28.13.3 Step 2.

```
#> # A tibble: 24 x 4
#>   year group  bot50 top1
#>   <dbl> <chr>  <dbl> <dbl>
#> 1  1910 US     0.00700 0.425
#> 2  1910 EU     0.0128  0.554
#> 3  1920 US     0.0102  0.410
#> 4  1920 EU     0.0139  0.496
#> 5  1930 US     0.00737 0.409
#> 6  1930 EU     0.0175  0.464
#> 7  1940 US     0.0112  0.319
#> 8  1940 EU     0.0282  0.412
#> 9  1950 US     0.0270  0.276
#> 10 1950 EU     0.0272  0.339
#> # i 14 more rows
```

28.13.4 Step 3.

```
df_f11 %>%
  rename(!year, US_bot50 = USbot50, US_top1 = UStop1,
         EU_bot50 = EUbot50, EU_top1 = EUtop1) %>%
  pivot_longer(!year, names_to = c("group", ".value"),
              names_sep = "_") %>%
  pivot_longer(3:4, names_to = "type", values_to = "value")
```

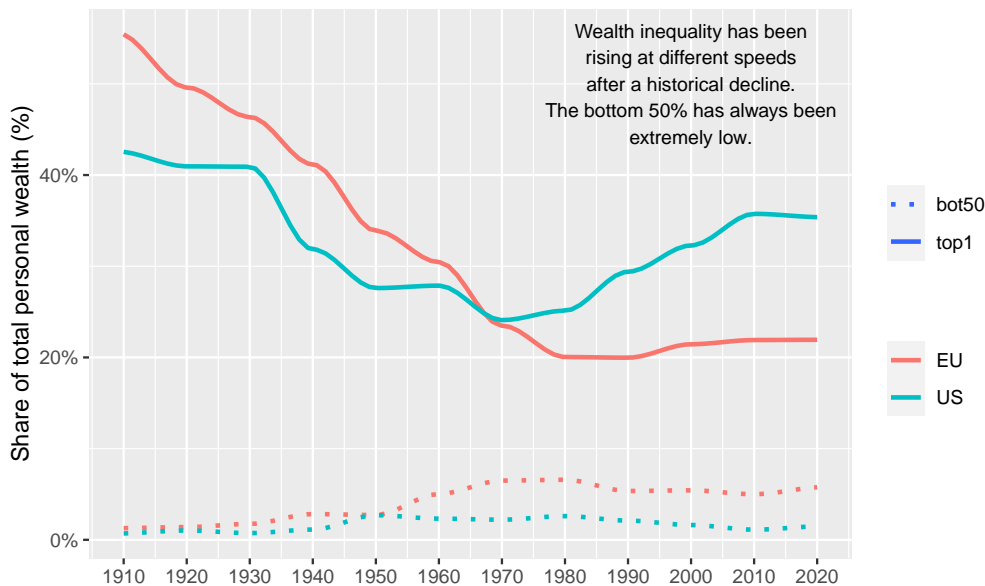
28.13.5 Step 3.

```
#> # A tibble: 48 x 4
#>   year group type  value
#>   <dbl> <chr> <chr>  <dbl>
#> 1  1910 US  bot50 0.00700
```

28.14 F8: The rise of private versus the decline of public wealth in rich countries, 1970-2020 - rename + pivot + pivot + fit curve

```
#> 2 1910 US top1 0.425
#> 3 1910 EU bot50 0.0128
#> 4 1910 EU top1 0.554
#> 5 1920 US bot50 0.0102
#> 6 1920 US top1 0.410
#> 7 1920 EU bot50 0.0139
#> 8 1920 EU top1 0.496
#> 9 1930 US bot50 0.00737
#> 10 1930 US top1 0.409
#> # i 38 more rows
```

Figure 11. Top 1% vs bottom 50% wealth shares in Western Europe and the US, 1910–2020



The following is similar to the previous example.

28.14 F8: The rise of private versus the decline of public wealth in rich countries, 1970-2020 - rename + pivot + pivot + fit curve

```
df_f8 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F8"); df_f8
#> # A tibble: 51 x 17
#>   year Germany `Germany (private)` Spain `Spain (private)`
#>   <dbl> <dbl> <dbl> <dbl> <dbl>
#> 1 1970 1.11 2.30 0.604 4.06
#> 2 1971 1.12 2.25 0.657 4.53
#> 3 1972 1.11 2.27 0.624 4.36
#> 4 1973 1.11 2.23 0.596 4.46
```

```

#> 5 1974 1.13 2.25 0.586 4.64
#> 6 1975 1.12 2.35 0.602 4.83
#> 7 1976 1.03 2.34 0.581 4.46
#> 8 1977 1.01 2.42 0.586 4.10
#> 9 1978 0.990 2.52 0.604 4.10
#> 10 1979 0.989 2.55 0.621 4.20
#> # i 41 more rows
#> # i 12 more variables: France <dbl>,
#> # `France (private)` <dbl>, UK <dbl>,
#> # `UK (private)` <dbl>, Japan <dbl>,
#> # `Japan (private)` <dbl>, Norway <dbl>,
#> # `Norway (private)` <dbl>, USA <dbl>,
#> # `USA (private)` <dbl>, gwealAVGRICH <dbl>, ...

```

```

df_f8 %>%
  select(year, Germany_public = Germany, Germany_private = 'Germany (private)',
         Spain_public = Spain, Spain_private = 'Spain (private)',
         France_public = France, France_private = 'France (private)',
         UK_public = UK, UK_private = 'UK (private)',
         Japan_public = Japan, Japan_private = 'Japan (private)',
         Norway_public = Norway, Norway_private = 'Norway (private)',
         USA_public = USA, USA_private = 'USA (private)') %>%
  pivot_longer(!year, names_to = c("country", ".value"), names_sep = "_") %>%
  pivot_longer(3:4, names_to = "type", values_to = "value") %>%
  ggplot() +
  stat_smooth(aes(x = year, y = value, color = country, linetype = type),
             span = 0.25, se = FALSE, size=0.75) +
  scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
  labs(title = "Figure 8. The rise of private versus the decline of public
         wealth in rich countries, 1970-2020",
       x = "", y = "wealth as as % of national income", color = "", type = "")

```

28.14.1 Step 1

```

df_f8 %>%
  select(year, Germany_public = Germany, Germany_private = 'Germany (private)',
         Spain_public = Spain, Spain_private = 'Spain (private)',
         France_public = France, France_private = 'France (private)',
         UK_public = UK, UK_private = 'UK (private)',
         Japan_public = Japan, Japan_private = 'Japan (private)',
         Norway_public = Norway, Norway_private = 'Norway (private)',

```

```
USA_public = USA, USA_private = 'USA (private)')
```

```
#> # A tibble: 51 x 15
#>   year Germany_public Germany_private Spain_public
#>   <dbl>         <dbl>         <dbl>         <dbl>
#> 1  1970           1.11           2.30           0.604
#> 2  1971           1.12           2.25           0.657
#> 3  1972           1.11           2.27           0.624
#> 4  1973           1.11           2.23           0.596
#> 5  1974           1.13           2.25           0.586
#> 6  1975           1.12           2.35           0.602
#> 7  1976           1.03           2.34           0.581
#> 8  1977           1.01           2.42           0.586
#> 9  1978           0.990          2.52           0.604
#> 10 1979           0.989          2.55           0.621
#> # i 41 more rows
#> # i 11 more variables: Spain_private <dbl>,
#> #   France_public <dbl>, France_private <dbl>,
#> #   UK_public <dbl>, UK_private <dbl>, Japan_public <dbl>,
#> #   Japan_private <dbl>, Norway_public <dbl>,
#> #   Norway_private <dbl>, USA_public <dbl>,
#> #   USA_private <dbl>
```

28.14.2 Step 2.

```
df_f8 %>%
  select(year, Germany_public = Germany, Germany_private = 'Germany (private)',
         Spain_public = Spain, Spain_private = 'Spain (private)',
         France_public = France, France_private = 'France (private)',
         UK_public = UK, UK_private = 'UK (private)',
         Japan_public = Japan, Japan_private = 'Japan (private)',
         Norway_public = Norway, Norway_private = 'Norway (private)',
         USA_public = USA, USA_private = 'USA (private)') %>%
  pivot_longer(!year, names_to = c("country", ".value"), names_sep = "_")
```

```
#> # A tibble: 357 x 4
#>   year country public private
#>   <dbl> <chr>    <dbl>  <dbl>
#> 1  1970 Germany  1.11    2.30
#> 2  1970 Spain    0.604   4.06
#> 3  1970 France  0.422   3.12
```

```
#> 4 1970 UK      0.601  2.85
#> 5 1970 Japan  0.719  3.09
#> 6 1970 Norway NA      NA
#> 7 1970 USA    0.364  3.26
#> 8 1971 Germany 1.12   2.25
#> 9 1971 Spain  0.657  4.53
#> 10 1971 France 0.443  3.06
#> # i 347 more rows
```

28.14.3 Step 3.

```
df_f8 %>%
  select(year, Germany_public = Germany, Germany_private = 'Germany (private)',
         Spain_public = Spain, Spain_private = 'Spain (private)',
         France_public = France, France_private = 'France (private)',
         UK_public = UK, UK_private = 'UK (private)',
         Japan_public = Japan, Japan_private = 'Japan (private)',
         Norway_public = Norway, Norway_private = 'Norway (private)',
         USA_public = USA, USA_private = 'USA (private)') %>%
  pivot_longer(!year, names_to = c("country", ".value"), names_sep = "_") %>%
  pivot_longer(3:4, names_to = "type", values_to = "value")
```

```
#> # A tibble: 714 x 4
#>   year country type    value
#>   <dbl> <chr>  <chr>  <dbl>
#> 1 1970 Germany public  1.11
#> 2 1970 Germany private 2.30
#> 3 1970 Spain  public  0.604
#> 4 1970 Spain  private 4.06
#> 5 1970 France public  0.422
#> 6 1970 France private 3.12
#> 7 1970 UK      public  0.601
#> 8 1970 UK      private 2.85
#> 9 1970 Japan  public  0.719
#> 10 1970 Japan private 3.09
#> # i 704 more rows
```

28.14.4 Step 3. Final Step

```
df_f8 %>%
  select(year, Germany_public = Germany, Germany_private = 'Germany (private)',
```



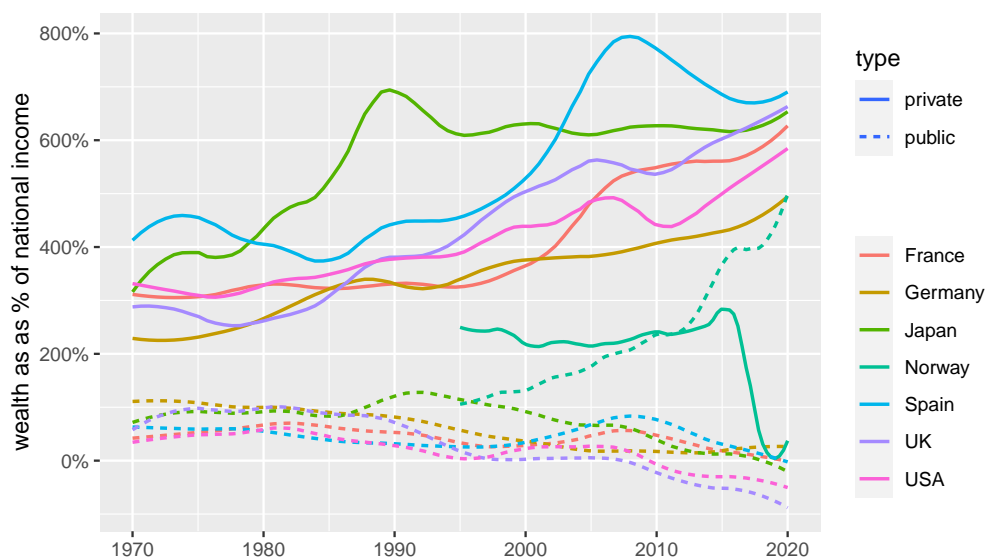
```

Spain_public = Spain, Spain_private = 'Spain (private)',
France_public = France, France_private = 'France (private)',
UK_public = UK, UK_private = 'UK (private)',
Japan_public = Japan, Japan_private = 'Japan (private)',
Norway_public = Norway, Norway_private = 'Norway (private)',
USA_public = USA, USA_private = 'USA (private)' %>%
pivot_longer(!year, names_to = c("country", ".value"), names_sep = "_") %>%
pivot_longer(3:4, names_to = "type", values_to = "value") %>%
ggplot() +
stat_smooth(aes(x = year, y = value, color = country, linetype = type),
            formula = y~x, method = "loess", span = 0.25, se = FALSE, size=0.75) +
scale_y_continuous(labels = scales::percent_format(accuracy = 1)) +
labs(title = "Figure 8. The rise of private versus the decline of public wealth
       \nin rich countries, 1970-2020",
      x = "", y = "wealth as as % of national income", color = "", type = "")

```

Figure 8. The rise of private versus the decline of public wealth

in rich countries, 1970–2020



28.15 F15: Per capita emissions across the world, 2019 - add row names + dodge

```

df_f15 <- read_excel("./data/WIR2022s.xlsx", sheet = "data-F15"); df_f15
#> # A tibble: 24 x 4
#>   regionWID      group      tcap mark
#>   <chr>          <chr>    <dbl> <dbl>
#> 1 East Asia     Bottom 50% 3.12    1

```

```

#> 2 <NA> Middle 40% 7.91 1
#> 3 <NA> Top 10% 38.9 1
#> 4 Europe Bottom 50% 5.09 2
#> 5 <NA> Middle 40% 10.6 2
#> 6 <NA> Top 10% 29.2 2
#> 7 North America Bottom 50% 9.67 3
#> 8 <NA> Middle 40% 21.7 3
#> 9 <NA> Top 10% 73.0 3
#> 10 South & South-East Asia Bottom 50% 1.04 4
#> # i 14 more rows

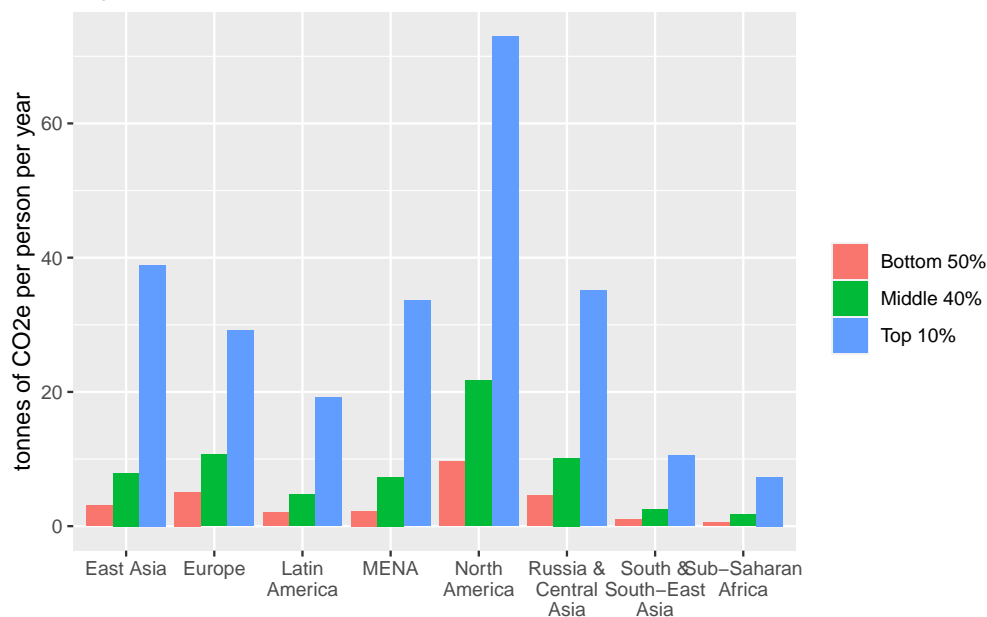
```

```

df_f15 %>% mutate(region = rep(regionWID[!is.na(regionWID)], each = 3)) %>%
  select(region, group, tcap) %>%
  ggplot(aes(x = region, y = tcap, fill = group)) +
  geom_col(position = "dodge") +
  scale_x_discrete(labels = function(x) stringr::str_wrap(x, width = 10)) +
  labs(title = "Figure 15 Per capita emissions across the world, 2019",
       x = "", y = "tonnes of CO2e per person per year", fill = "")

```

Figure 15 Per capita emissions across the world, 2019



Review one by one, referring to the following.

References: <https://ds-sl.github.io/data-analysis/wir2022.nb.html>

第 29 章

e-Stat

29.1 API の使い方

29.1.1 はじめての e-Stat with API

はじめて、R の `estatapi` を使って、e-Stat のデータを取得する方法を簡単に説明します。

初期設定

パッケージの、`estatapi` が、e-Stat のデータを取得するためのものです。`tidyverse` は基本的な解析をおこなうため、`showtext` と `showtext_auto()` は、グラフなどに、日本語が含まれる場合の文字化けを防ぐためのものです。

インストールされていないパッケージがありましたら、上のメニューの Tools > Install Packages から、インストールしてください。

```
library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats   1.0.0      v stringr   1.5.0
#> v ggplot2   3.4.3      v tibble    3.2.1
#> v lubridate 1.9.2      v tidyr     1.3.0
#> v purrr     1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(showtext)
#> Loading required package: sysfonts
#> Loading required package: showtextdb
library(estatapi)
#> このサービスは、政府統計総合窓口 (e-Stat) の API 機能を使用していますが、サービスの内容は国によって保証されたもので
```

```
showtext_auto()
```

29.1.2 e-Stat 登録と、アプリケーション ID の取得

e-Stat API を利用するには、appId が必要です。

1. e-stat: <https://www.e-stat.go.jp/> のページへ行く。
2. 右上の新規登録から、アカウント登録をします。
 - メールアドレスを入力し、そこに届いてリンクを開き、パスワードを設定。
 - パスワード設定は、Google などのアカウント連携ですることも可能です。
3. アカウントにログイン。上の帯に、メールアドレスが表示されます。
4. その右のマイページに行き、API 機能（アプリケーション ID 発行）へ。まず、必要事項の入力が必要ですが、説明をみてください。
 - 名称：API 機能を利用するアプリケーション名（システム名等） - R または、RStudio
 - URL：アプリケーションの URL（トップページ等の URL）
- If you do not use it on the public site, please enter “http: // test.localhost /” etc. - localhost でよいでしょう。
 - 概要：アプリケーション（システム）の概要 - estatapi とでも書いておけば良いでしょう。
 - 右の発行を押すと、appId が表示されます。これを記録。

29.2 estatapi 利用概要

CRAN estatapi URL: <https://CRAN.R-project.org/package=estatapi>

29.2.1 アプリケーション ID の設定

```
appId <- " " # 私のものは、英数40文字
```

29.2.2 統計表情報取得 (estat_getStatsList())

提供されている統計表を検索します。この関数は、結果を tbl_df (dplyr の data.frame。data.frame とほぼ同じように扱える) として返します。

STAT_NAME や GOV_ORG は人間が読みやすい形式のラベルになっていますが、プログラム中で扱う場合はコードのままの方が都合がいいこともあります。そのときは、use_label = FALSE を指定してください。

```
estat_getStatsList(appId = appId, searchWord = "")
```

```

estat_getStatsList(appId = appId, searchWord = "人口推計")
#> # A tibble: 478 x 21
#>   `@id`      STAT_NAME GOV_ORG STATISTICS_NAME TITLE CYCLE
#>   <chr>      <chr>      <chr>      <chr>      <chr> <chr>
#> 1 0000150001 人口推計 総務省 人口推計 平成 3~ 年齢 ~ -
#> 2 0000150002 人口推計 総務省 人口推計 平成 3~ 年齢 ~ -
#> 3 0000150003 人口推計 総務省 人口推計 平成 3~ 人口 ~ -
#> 4 0000150004 人口推計 総務省 人口推計 平成 3~ 年齢 ~ -
#> 5 0000150005 人口推計 総務省 人口推計 平成 3~ 年齢 ~ -
#> 6 0000150006 人口推計 総務省 人口推計 平成 3~ 年齢 ~ -
#> 7 0000150007 人口推計 総務省 人口推計 平成 3~ 男女 ~ -
#> 8 0000150008 人口推計 総務省 人口推計 平成 3~ 男女 ~ -
#> 9 0000150009 人口推計 総務省 人口推計 平成 3~ 年 (~ -
#> 10 0000150010 人口推計 総務省 人口推計 平成 3~ 男女 ~ -
#> # i 468 more rows
#> # i 15 more variables: SURVEY_DATE <chr>, OPEN_DATE <chr>,
#> #   SMALL_AREA <chr>, COLLECT_AREA <chr>,
#> #   MAIN_CATEGORY <chr>, SUB_CATEGORY <chr>,
#> #   OVERALL_TOTAL_NUMBER <chr>, UPDATED_DATE <chr>,
#> #   TABULATION_CATEGORY <chr>,
#> #   TABULATION_SUB_CATEGORY1 <chr>, DESCRIPTION <chr>, ...

```

```

estat_getStatsList(appId = appId, searchWord = "外国人人口")
#> # A tibble: 38 x 22
#>   `@id`      STAT_NAME GOV_ORG STATISTICS_NAME TITLE CYCLE
#>   <chr>      <chr>      <chr>      <chr>      <chr> <chr>
#> 1 0000150003 人口推計 総務省 人口推計 平成 3~ 人口 ~ -
#> 2 0000150022 人口推計 総務省 人口推計 平成 4~ 人口 ~ -
#> 3 0000150041 人口推計 総務省 人口推計 平成 5~ 人口 ~ -
#> 4 0000150062 人口推計 総務省 人口推計 平成 6~ 人口 ~ -
#> 5 0000150083 人口推計 総務省 人口推計 平成 8~ 人口 ~ -
#> 6 0000150115 人口推計 総務省 人口推計 平成 9~ [参 ~ -
#> 7 0000150137 人口推計 総務省 人口推計 平成 10~ [参 ~ -
#> 8 0000150159 人口推計 総務省 人口推計 平成 11~ [参 ~ -
#> 9 0000150181 人口推計 総務省 人口推計 平成 13~ [参 ~ -
#> 10 0000150203 人口推計 総務省 人口推計 平成 14~ [参 ~ -
#> # i 28 more rows
#> # i 16 more variables: SURVEY_DATE <chr>, OPEN_DATE <chr>,
#> #   SMALL_AREA <chr>, COLLECT_AREA <chr>,
#> #   MAIN_CATEGORY <chr>, SUB_CATEGORY <chr>,
#> #   OVERALL_TOTAL_NUMBER <chr>, UPDATED_DATE <chr>,

```

```
#> # TABULATION_CATEGORY <chr>,
#> # TABULATION_SUB_CATEGORY1 <chr>, DESCRIPTION <chr>, ...
```

```
statsDataAPI=0003448219 statsDataAPI=0004001640
```

```
estat_getStatsList(appId = appId, searchWord = "国勢調査 時系列データ 男女, 年齢, 配
```

```
#> # A tibble: 16 x 19
#>   `@id`      STAT_NAME GOV_ORG STATISTICS_NAME TITLE CYCLE
#>   <chr>      <chr>      <chr>      <chr>      <chr> <chr>
#> 1 0003410379 国勢調査 総務省 時系列データ 男 ~ 男女 ~ -
#> 2 0003410380 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 3 0003410381 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 4 0003410382 国勢調査 総務省 時系列データ 男 ~ 配偶 ~ -
#> 5 0003410383 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 6 0003412413 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 7 0003412414 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 8 0003412415 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 9 0003412416 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 10 0003412417 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 11 0003412418 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 12 0003412419 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 13 0003412420 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 14 0003448300 国勢調査 総務省 時系列データ 男 ~ 【不 ~ -
#> 15 0003448299 国勢調査 総務省 時系列データ 男 ~ 年齢 ~ -
#> 16 0003448301 国勢調査 総務省 時系列データ 男 ~ 【不 ~ -
#> # i 13 more variables: SURVEY_DATE <chr>, OPEN_DATE <chr>,
#> # SMALL_AREA <chr>, COLLECT_AREA <chr>,
#> # MAIN_CATEGORY <chr>, SUB_CATEGORY <chr>,
#> # OVERALL_TOTAL_NUMBER <chr>, UPDATED_DATE <chr>,
#> # TABULATION_CATEGORY <chr>,
#> # TABULATION_SUB_CATEGORY1 <chr>, DESCRIPTION <chr>,
#> # TABLE_NAME <chr>, TABLE_EXPLANATION <chr>
```

```
statsDataID=0003410379
```

29.2.3 メタ情報取得 (estat_getMetaInfo())

統計データのメタ情報を取得します。この関数は、結果を list として返します。list の各要素が、それぞれのデータ項目についてのメタ情報を含んだ tbl_df になっています。

```
meta_info <- estat_getMetaInfo(appId = appId, statsDataId = "")
```

```

meta_info <- estat_getMetaInfo(appId = appId, statsDataId = "0003410379")
glimpse(meta_info)
#> List of 5
#> $ tab : tibble [2 x 4] (S3: tbl_df/tbl/data.frame)
#> ..$ @code : chr [1:2] "020" "1120"
#> ..$ @name : chr [1:2] "人口" "人口性比"
#> ..$ @level: chr [1:2] "" ""
#> ..$ @unit : chr [1:2] "人" NA
#> $ cat01 : tibble [3 x 3] (S3: tbl_df/tbl/data.frame)
#> ..$ @code : chr [1:3] "100" "110" "120"
#> ..$ @name : chr [1:3] "総数" "男" "女"
#> ..$ @level: chr [1:3] "1" "1" "1"
#> $ area : tibble [50 x 4] (S3: tbl_df/tbl/data.frame)
#> ..$ @code : chr [1:50] "00000" "00100" "00200" "01000" ...
#> ..$ @name : chr [1:50] "全国" "人口集中地区" "人口集中地区以外の地区" "北海道" ...
#> ..$ @level : chr [1:50] "1" "2" "2" "2" ...
#> ..$ @parentCode: chr [1:50] NA "00000" "00000" "00000" ...
#> $ time : tibble [21 x 3] (S3: tbl_df/tbl/data.frame)
#> ..$ @code : chr [1:21] "1920000000" "1925000000" "1930000000" "1935000000" ...
#> ..$ @name : chr [1:21] "1920年" "1925年" "1930年" "1935年" ...
#> ..$ @level: chr [1:21] "1" "1" "1" "1" ...
#> $ .names: tibble [4 x 2] (S3: tbl_df/tbl/data.frame)
#> ..$ id : chr [1:4] "tab" "cat01" "area" "time"
#> ..$ name: chr [1:4] "表章項目" "男女_時系列" "地域_時系列" "時間軸 (調査年)"

```

```

meta_info$cat01
#> # A tibble: 3 x 3
#>   `@code` `@name` `@level`
#>   <chr>   <chr>   <chr>
#> 1 100     総数     1
#> 2 110     男       1
#> 3 120     女       1

```

29.2.4 統計データ取得 (estat_getStatsData())

統計データを取得します。この関数は、結果をメタ情報と紐づけて `tbl_df` として返します。

必ず指定しなくてはいけないのは `appId` と `statsDataId` だけですが、それだけだとデータがかなり大きくなって取得に時間がかかります。 `cdCat01` (分類事項 01) などを指定して必要な項目だけに絞ることをおすすめします。他に絞り込みに指定できるパラメータについては公式ドキュメントを参照してください。

```

estat_getStatsData(
  appId = appId,
  statsDataId = "",
  cdCat01 = c("", ""))

```

```

estat_pop <- estat_getStatsData(appId = appId, statsDataId = "0003410379")
#> Fetching record 1-4200... (total: 4200 records)

```

```

estat_pop
#> # A tibble: 4,200 x 11
#>   tab_code 表章項目 cat01_code 男女_時系列 area_code
#>   <chr>    <chr>    <chr>    <chr>    <chr>
#> 1 020      人口      100      総数      00000
#> 2 020      人口      100      総数      00000
#> 3 020      人口      100      総数      00000
#> 4 020      人口      100      総数      00000
#> 5 020      人口      100      総数      00000
#> 6 020      人口      100      総数      00000
#> 7 020      人口      100      総数      00000
#> 8 020      人口      100      総数      00000
#> 9 020      人口      100      総数      00000
#> 10 020     人口      100      総数      00000
#> # i 4,190 more rows
#> # i 6 more variables: 地域_時系列 <chr>, time_code <chr>,
#> #   `時間軸 (調査年)` <chr>, unit <chr>, value <dbl>,
#> #   annotation <chr>

```

```

estat_pop_alien <- estat_getStatsData(appId = appId, statsDataId = "0003448219")
#> Fetching record 1-87... (total: 87 records)

```

```

estat_pop_alien
#> # A tibble: 87 x 15
#>   tab_code 表章項目 cat01_code 人口及び人口増減 cat02_code
#>   <chr>    <chr>    <chr>    <chr>    <chr>
#> 1 008      人口      204      人口      000
#> 2 008      人口      204      人口      000
#> 3 008      人口      204      人口      000
#> 4 008      人口      204      人口      000
#> 5 008      人口      204      人口      000
#> 6 008      人口      204      人口      000
#> 7 008      人口      204      人口      001
#> 8 008      人口      204      人口      001
#> 9 008      人口      204      人口      001
#> 10 008     人口      204      人口      001

```



```

#> # i 77 more rows
#> # i 10 more variables: 男女別 <chr>, cat03_code <chr>,
#> #   人口 <chr>, area_code <chr>, 全国 <chr>,
#> #   time_code <chr>, `時間軸 (年間)` <chr>, unit <chr>,
#> #   value <dbl>, annotation <chr>

estat_pop_alien2 <- estat_getStatsData(appId = appId, statsDataId = "0004001640")
#> Fetching record 1-87... (total: 87 records)
estat_pop_alien2
#> # A tibble: 87 x 15
#>   tab_code 表章項目 cat01_code 人口及び人口増減 cat02_code
#>   <chr>    <chr>    <chr>    <chr>                <chr>
#> 1 008      人口      204      人口                000
#> 2 008      人口      204      人口                000
#> 3 008      人口      204      人口                000
#> 4 008      人口      204      人口                000
#> 5 008      人口      204      人口                000
#> 6 008      人口      204      人口                000
#> 7 008      人口      204      人口                001
#> 8 008      人口      204      人口                001
#> 9 008      人口      204      人口                001
#> 10 008     人口      204      人口                001
#> # i 77 more rows
#> # i 10 more variables: 男女別 <chr>, cat03_code <chr>,
#> #   人口 <chr>, area_code <chr>, 全国 <chr>,
#> #   time_code <chr>, `時間軸 (年間)` <chr>, unit <chr>,
#> #   value <dbl>, annotation <chr>

```

0003445244

```

estat_pop_alien0 <- estat_getStatsData(appId = appId, lang = "E", statsDataId = "0003445244")
#> Fetching record 1-100000... (total: 100215 records)
#> Fetching record 100001-100215... (total: 100215 records)
estat_pop_alien0
#> # A tibble: 100,215 x 13
#>   tab_code tab_name cat01_code Sex   cat02_code Nationality
#>   <chr>    <chr>    <chr>    <chr> <chr>    <chr>
#> 1 2020_01 Populat~ 0      Total 0      Total
#> 2 2020_01 Populat~ 0      Total 0      Total
#> 3 2020_01 Populat~ 0      Total 0      Total
#> 4 2020_01 Populat~ 0      Total 0      Total
#> 5 2020_01 Populat~ 0      Total 0      Total
#> 6 2020_01 Populat~ 0      Total 0      Total

```

```

#> 7 2020_01 Populat~ 0          Total 0          Total
#> 8 2020_01 Populat~ 0          Total 0          Total
#> 9 2020_01 Populat~ 0          Total 0          Total
#> 10 2020_01 Populat~ 0         Total 0          Total
#> # i 100,205 more rows
#> # i 7 more variables: area_code <chr>,
#> #   `Japan, Prefectures, Municipalities` <chr>,
#> #   time_code <chr>, Time <chr>, unit <chr>, value <dbl>,
#> #   annotation <chr>

```

```

estat_pop_alien0 |> distinct(Nationality) |> pull()
#> [1] "Total"
#> [2] "Foreigner"
#> [3] "Republic of Korea"
#> [4] "People's Republic of China"
#> [5] "Republic of the Philippines"
#> [6] "Kingdom of Thailand"
#> [7] "Republic of Indonesia"
#> [8] "Socialist Republic of Viet Nam"
#> [9] "India"
#> [10] "Federal Democratic Republic of Nepal"
#> [11] "United Kingdom of Great Britain and Northern Ireland"
#> [12] "United States of America"
#> [13] "Federative Republic of Brazil"
#> [14] "Republic of Peru"
#> [15] "Others"
#> [16] "Japanese"
#> [17] "Nationality not reported"

```

```

estat_pop_alien2020 <- estat_getStatsData(appId = appId, statsDataId = "000344524")
#> Fetching record 1-100000... (total: 100215 records)
#> Fetching record 100001-100215... (total: 100215 records)
estat_pop_alien2020
#> # A tibble: 100,215 x 13
#>   tab_code 表章事項 cat01_code 男女  cat02_code 国籍
#>   <chr>     <chr>     <chr>     <chr> <chr>     <chr>
#> 1 2020_01 人口      0          総数 0          総数
#> 2 2020_01 人口      0          総数 0          総数
#> 3 2020_01 人口      0          総数 0          総数
#> 4 2020_01 人口      0          総数 0          総数
#> 5 2020_01 人口      0          総数 0          総数
#> 6 2020_01 人口      0          総数 0          総数

```

```
#> 7 2020_01 人口 0 総数 0 総数
#> 8 2020_01 人口 0 総数 0 総数
#> 9 2020_01 人口 0 総数 0 総数
#> 10 2020_01 人口 0 総数 0 総数
#> # i 100,205 more rows
#> # i 7 more variables: area_code <chr>,
#> # `全国, 都道府県, 市区町村` <chr>, time_code <chr>,
#> # `時間軸 (年次)` <chr>, unit <chr>, value <dbl>,
#> # annotation <chr>
```

```
estat_pop_alien2020 |> distinct(国籍) |> pull()
```

```
#> [1] "総数"
#> [2] "外国人"
#> [3] "韓国, 朝鮮"
#> [4] "中国"
#> [5] "フィリピン"
#> [6] "タイ"
#> [7] "インドネシア"
#> [8] "ベトナム"
#> [9] "インド"
#> [10] "ネパール"
#> [11] "イギリス"
#> [12] "アメリカ"
#> [13] "ブラジル"
#> [14] "ペルー"
#> [15] "その他"
#> [16] "日本人"
#> [17] "日本人・外国人の別「不詳」"
```

```
estat_pop_alien2020 |> distinct(男女) |> pull()
```

```
#> [1] "総数" "男" "女"
```

```
colnames(estat_pop_alien2020)
```

```
#> [1] "tab_code" "表章事項"
#> [3] "cat01_code" "男女"
#> [5] "cat02_code" "国籍"
#> [7] "area_code" "全国, 都道府県, 市区町村"
#> [9] "time_code" "時間軸 (年次)"
#> [11] "unit" "value"
#> [13] "annotation"
```

```
estat_pop_alien2020 |>
```

```
filter(`全国, 都道府県, 市区町村` == "全国") |>
```

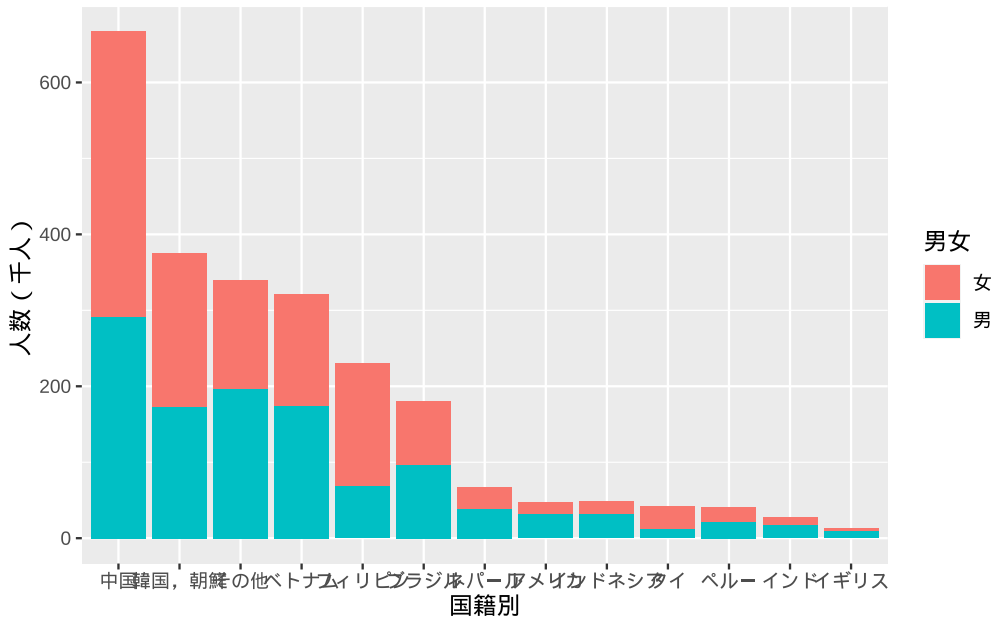
```

filter(男女 == "総数") |>
filter(国籍 != "総数") |>
select(国籍, value) |>
arrange(desc(value))
#> # A tibble: 16 x 2
#>   国籍                value
#>   <chr>                <dbl>
#> 1 日本人              121541155
#> 2 外国人              2402460
#> 3 日本人・外国人の別「不詳」 2202484
#> 4 中国                667475
#> 5 韓国, 朝鮮          374593
#> 6 その他              339634
#> 7 ベトナム           320805
#> 8 フィリピン         230351
#> 9 ブラジル           180014
#> 10 ネパール           67325
#> 11 インドネシア      49147
#> 12 アメリカ          47875
#> 13 タイ              42702
#> 14 ペルー            41034
#> 15 インド            27915
#> 16 イギリス          13590

estat_pop_alien2020 |>
  filter(`全国, 都道府県, 市区町村` == "全国") |>
  filter(男女 != "総数") |>
  filter(!(国籍 %in% c("総数", "日本人", "外国人", "日本人・外国人の別「不詳」"))) |>
  select(国籍, 男女, value) |>
  arrange(desc(value)) |>
  mutate(国籍別 = factor(fct_inorder(国籍))) |>
  mutate(`人数 (千人)` = value/1000) |>
  ggplot(aes(国籍別, `人数 (千人)`, fill = 男女)) + geom_col() +
  labs(title = "国籍別日本在住外国人人数", caption = "令和 2 (2020) 年 国勢調査 統計表表

```

国籍別日本在住外国人人数



令和2(2020)年国勢調査 統計表表示ID=0003445244

第 30 章

その他のデータベース

30.1 国際機関

30.1.1 国連児童基金 (UNICEF : United Nations Children's Fund)

<https://data.unicef.org>

30.2 世界の国の政府機関

30.2.1 米国 : DATA.GOV

The Home of the U.S. Government's Open Data: <https://data.gov/>

30.2.2 米国情勢調査 : United States Census Bureau

Explore Census Data: <https://data.census.gov>

30.2.3 中国 : National Bureaus of Statistics of China

<http://www.stats.gov.cn/english/>

30.2.5 英国 : Find Open Data: data.gov.uk

<https://www.data.gov.uk>

30.2.6 英国データサービス : UK Data Service

<https://ukdataservice.ac.uk>

30.2.7 欧州連合 (EU) : The official portal for European data

<https://data.europa.eu/en>

30.2.8 韓国：Statistics Korea

<https://kostat.go.kr/anse/>

30.3 ClinicalTrials.gov

ClinicalTrials.gov is a place to learn about clinical studies from around the world.

<https://clinicaltrials.gov/>

30.4 Harvard Dataverse

<https://dataverse.harvard.edu/>

30.5 Kagle

- Kagle Home: <https://www.kaggle.com>
- Kagle Datasets: <https://www.kaggle.com/datasets>

30.6 Gapminder

30.6.1 パッケージ Gapminder を使って

- gapminder: Data from Gapminder: <https://cran.r-project.org/web/packages/gapminder/index.html>
- Gapminder Data: <https://www.gapminder.org/data/>

```
library(tidyverse)
library(gapminder)
```

すでに、`dplyr` をつかった変形で確認しましたが、簡単に、データを見ておきましょう。

```
df_gm <- gapminder
df_gm %>% slice(1:10)
#> # A tibble: 10 x 6
#>   country      continent  year lifeExp      pop gdpPercap
#>   <fct>         <fct>    <int> <dbl>    <int>    <dbl>
#> 1 Afghanistan Asia      1952  28.8  8425333  779.
#> 2 Afghanistan Asia      1957  30.3  9240934  821.
#> 3 Afghanistan Asia      1962  32.0 10267083  853.
#> 4 Afghanistan Asia      1967  34.0 11537966  836.
```



```
#> 5 Afghanistan Asia      1972   36.1 13079460    740.
#> 6 Afghanistan Asia      1977   38.4 14880372    786.
#> 7 Afghanistan Asia      1982   39.9 12881816    978.
#> 8 Afghanistan Asia      1987   40.8 13867957    852.
#> 9 Afghanistan Asia      1992   41.7 16317921    649.
#> 10 Afghanistan Asia     1997   41.8 22227415    635.
```

```
glimpse(df_gm)
```

```
#> Rows: 1,704
#> Columns: 6
#> $ country <fct> "Afghanistan", "Afghanistan", "Afghanist~
#> $ continent <fct> Asia, Asia, Asia, Asia, Asia, Asia, Asia~
#> $ year <int> 1952, 1957, 1962, 1967, 1972, 1977, 1982~
#> $ lifeExp <dbl> 28.801, 30.332, 31.997, 34.020, 36.088, ~
#> $ pop <int> 8425333, 9240934, 10267083, 11537966, 13~
#> $ gdpPercap <dbl> 779.4453, 820.8530, 853.1007, 836.1971, ~
```

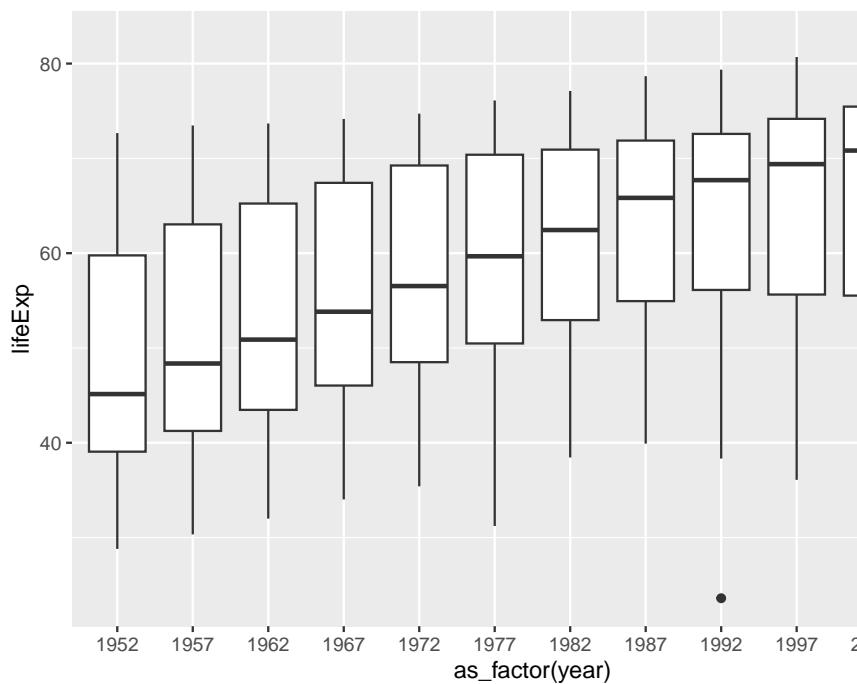
```
summary(df_gm)
```

```
#>      country      continent      year
#> Afghanistan: 12 Africa :624 Min. :1952
#> Albania : 12 Americas:300 1st Qu.:1966
#> Algeria : 12 Asia :396 Median :1980
#> Angola : 12 Europe :360 Mean :1980
#> Argentina : 12 Oceania : 24 3rd Qu.:1993
#> Australia : 12 Max. :2007
#> (Other) :1632
#>      lifeExp      pop      gdpPercap
#> Min. :23.60 Min. :6.001e+04 Min. : 241.2
#> 1st Qu.:48.20 1st Qu.:2.794e+06 1st Qu.: 1202.1
#> Median :60.71 Median :7.024e+06 Median : 3531.8
#> Mean :59.47 Mean :2.960e+07 Mean : 7215.3
#> 3rd Qu.:70.85 3rd Qu.:1.959e+07 3rd Qu.: 9325.5
#> Max. :82.60 Max. :1.319e+09 Max. :113523.1
#>
```

```
unique(df_gm$year)
```

```
#> [1] 1952 1957 1962 1967 1972 1977 1982 1987 1992 1997 2002
#> [12] 2007
```

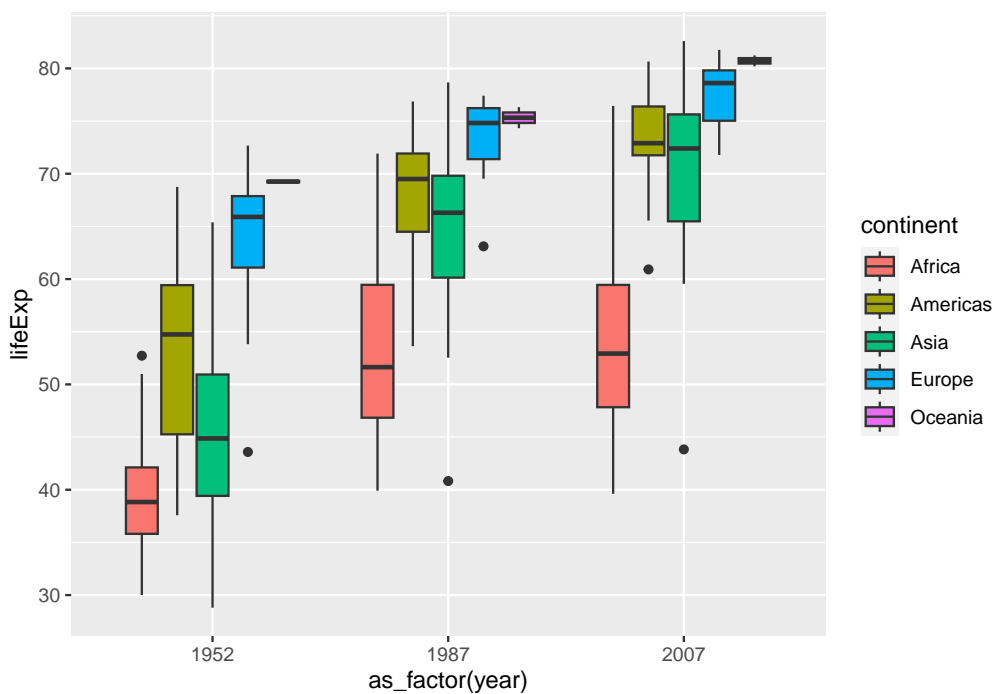
```
ggplot(df_gm, aes(x = as_factor(year), y = lifeExp)) + geom_boxplot()
```



■30.6.1.0.1 Box Plot

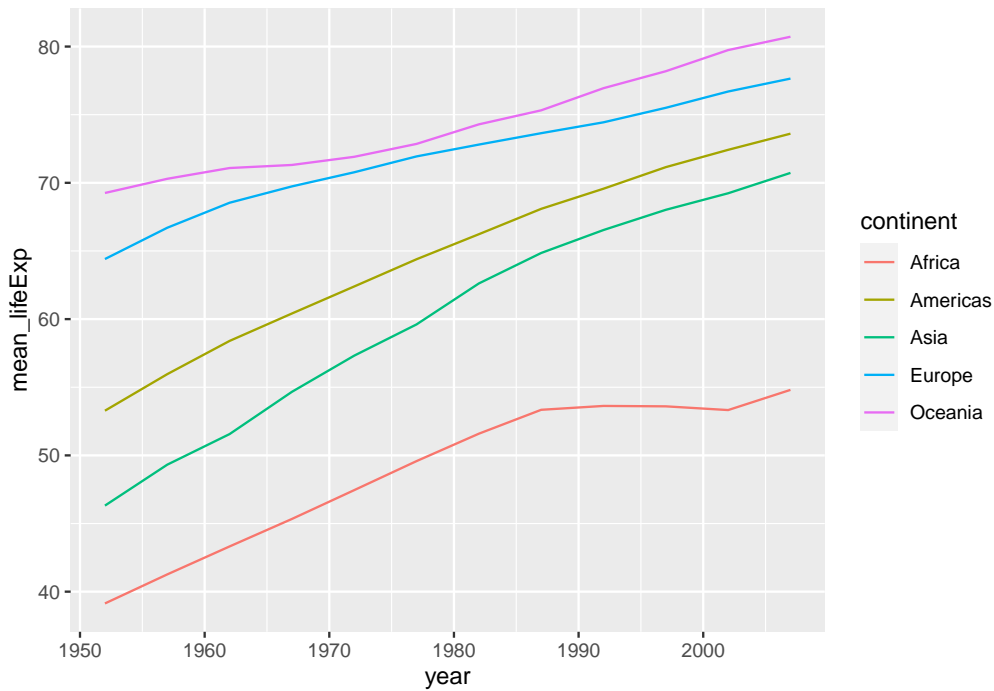
fill や color を追加してみましょう。

```
df_gm %>% filter(year %in% c(1952, 1987, 2007)) %>%
  ggplot(aes(x=as_factor(year), y = lifeExp, fill = continent)) +
  geom_boxplot()
```

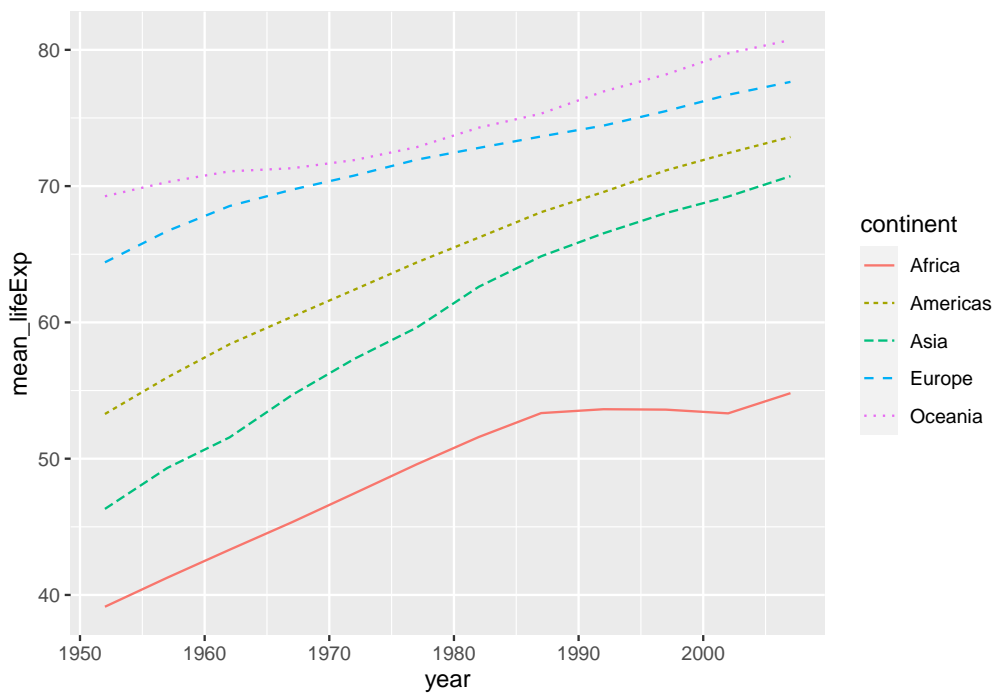


折れ線グラフの例です。

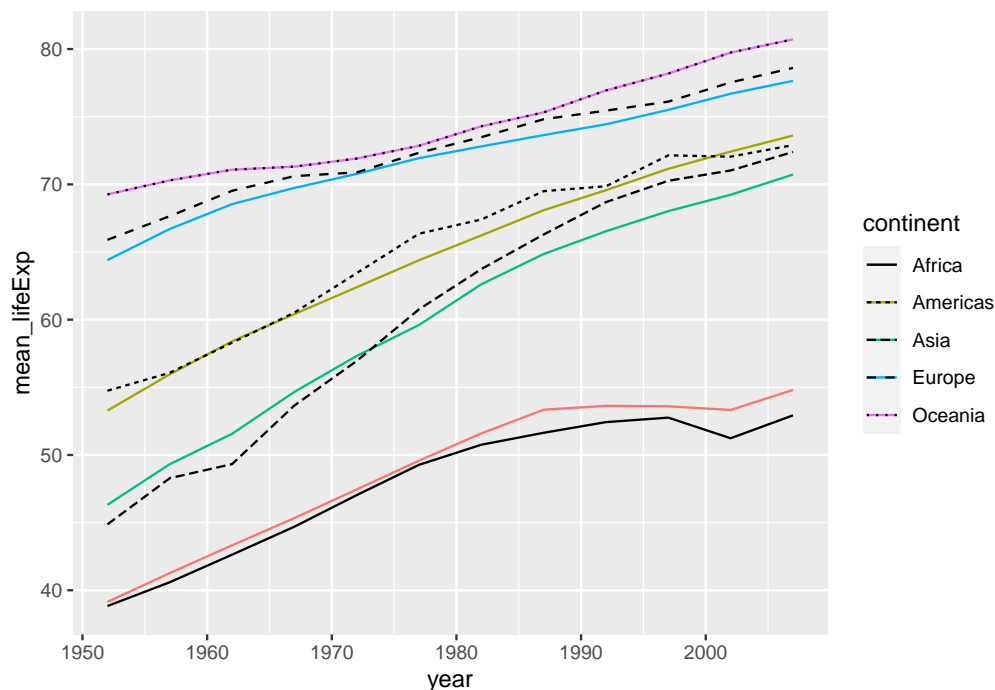
```
df_lifeExp <- df_gm %>%
  group_by(continent, year) %>%
  summarize(mean_lifeExp = mean(lifeExp), median_lifeExp = median(lifeExp), max_lifeExp = max(lifeExp),
df_lifeExp %>% ggplot(aes(x = year, y = mean_lifeExp, color = continent)) +
  geom_line()
```



```
df_lifeExp %>% ggplot(aes(x = year, y = mean_lifeExp, color = continent, linetype = continent)) +
  geom_line()
```



```
df_lifeExp %>% ggplot() +
  geom_line(aes(x = year, y = mean_lifeExp, color = continent)) +
  geom_line(aes(x = year, y = median_lifeExp, linetype = continent))
```



30.6.2 Original Data? WDI?

```
df_gm %>% slice(1:10)
#> # A tibble: 10 x 6
#>   country    continent  year lifeExp    pop  gdpPercap
#>   <fct>      <fct>    <int> <dbl>    <int>    <dbl>
#> 1 Afghanistan Asia      1952  28.8  8425333  779.
#> 2 Afghanistan Asia      1957  30.3  9240934  821.
#> 3 Afghanistan Asia      1962  32.0 10267083  853.
#> 4 Afghanistan Asia      1967  34.0 11537966  836.
#> 5 Afghanistan Asia      1972  36.1 13079460  740.
#> 6 Afghanistan Asia      1977  38.4 14880372  786.
#> 7 Afghanistan Asia      1982  39.9 12881816  978.
#> 8 Afghanistan Asia      1987  40.8 13867957  852.
#> 9 Afghanistan Asia      1992  41.7 16317921  649.
#> 10 Afghanistan Asia      1997  41.8 22227415  635.
```

さて、このデータは、どのようにして集めているのでしょうか。WDI のデータと同じでしょうか。

第 IV 部

PART IV EDA

第 31 章

探索的データ解析

31.1 探索的データ解析 (EDA) とは

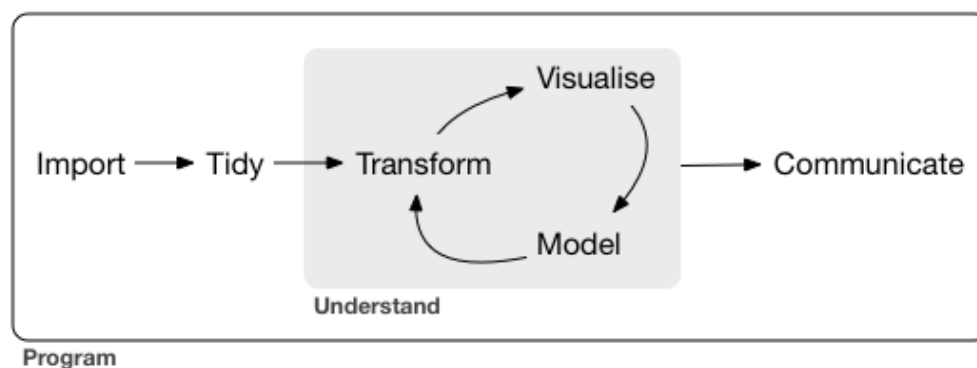


図31.1 image from r4ds

以下は、Posit Primers: Visualise Data から

探索的データ解析 (EDA) は、データを理解するための反復的なサイクルです。EDA では、以下のことを行います。

1. データに関する問いを作成する
2. データの可視化、変形・整形、モデリングによって、問いの答えを探索する。
3. 学んだことを使って、問いをより洗練されたものとする。

EDA は、あらゆるデータ分析において重要な役割を担います。EDA によって、課題解決のいとぐちを発見することもありますし、他の課題との関係性を発見する場合があります。EDA を使用してデータの問題や品質を確認したり、データが信頼できるものであるかを見極める問いを作成できる場合があります。

31.2 探索的データ解析 (EDA) の一例

WDI の一つの指標を使って、流れを見てみましょう。

31.2.1 データの取得と読み込み - Data Import

NY.GDP.PCAP.CD: GDP per capita (current US\$)

```
df_wdi_gdppcap <- WDI(country = "all", indicator = c(gdp_pcap = "NY.GDP.PCAP.CD"))
write_csv(df_wdi_gdppcap, "./data/df_wdi_gdppcap.csv")
```

```
df_wdi_gdppcap
#> # A tibble: 16,492 x 5
#>   country                iso2c iso3c year gdp_pcap
#>   <chr>                  <chr> <chr> <dbl> <dbl>
#> 1 Africa Eastern and Southern ZH   AFE   2021  1550.
#> 2 Africa Eastern and Southern ZH   AFE   2020  1364.
#> 3 Africa Eastern and Southern ZH   AFE   2019  1512.
#> 4 Africa Eastern and Southern ZH   AFE   2018  1565.
#> 5 Africa Eastern and Southern ZH   AFE   2017  1629.
#> 6 Africa Eastern and Southern ZH   AFE   2016  1444.
#> 7 Africa Eastern and Southern ZH   AFE   2015  1539.
#> 8 Africa Eastern and Southern ZH   AFE   2014  1719.
#> 9 Africa Eastern and Southern ZH   AFE   2013  1730.
#> 10 Africa Eastern and Southern ZH  AFE   2012  1759.
#> # i 16,482 more rows
```

31.2.2 データ変形・整形 - Data Transformation

31.2.2.1 列を select

どの変数について分析するかを選ぶ。

```
df_wdi_gdppcap_small <- df_wdi_gdppcap %>%
  select(country, year, gdp_pcap)
df_wdi_gdppcap_small
#> # A tibble: 16,492 x 3
#>   country                year gdp_pcap
#>   <chr>                  <dbl> <dbl>
#> 1 Africa Eastern and Southern 2021  1550.
#> 2 Africa Eastern and Southern 2020  1364.
#> 3 Africa Eastern and Southern 2019  1512.
#> 4 Africa Eastern and Southern 2018  1565.
#> 5 Africa Eastern and Southern 2017  1629.
#> 6 Africa Eastern and Southern 2016  1444.
#> 7 Africa Eastern and Southern 2015  1539.
#> 8 Africa Eastern and Southern 2014  1719.
```



```
#> 9 Africa Eastern and Southern 2013 1730.
#> 10 Africa Eastern and Southern 2012 1759.
#> # i 16,482 more rows
```

31.2.2.2 行を filter

いくつかの国に、フォーカスして調べる。

```
df_wdi_gdppcap_short <- df_wdi_gdppcap %>%
  filter(country %in% c("Japan", "Germany", "United States"))
df_wdi_gdppcap_short
#> # A tibble: 186 x 5
#>   country iso2c iso3c year gdp_pcap
#>   <chr>   <chr> <chr> <dbl>   <dbl>
#> 1 Germany DE    DEU    2021  51204.
#> 2 Germany DE    DEU    2020  46773.
#> 3 Germany DE    DEU    2019  46794.
#> 4 Germany DE    DEU    2018  47939.
#> 5 Germany DE    DEU    2017  44653.
#> 6 Germany DE    DEU    2016  42136.
#> 7 Germany DE    DEU    2015  41103.
#> 8 Germany DE    DEU    2014  48024.
#> 9 Germany DE    DEU    2013  46299.
#> 10 Germany DE    DEU    2012  43856.
#> # i 176 more rows
```

列 (変数) と、行 (国) の選択を続けて、実行すると次のようになる。一つ一つ変形したデータ (オブジェクト) に名前をつけて、保存する必要がないので、パイプ (%>%) の活用は有用である。

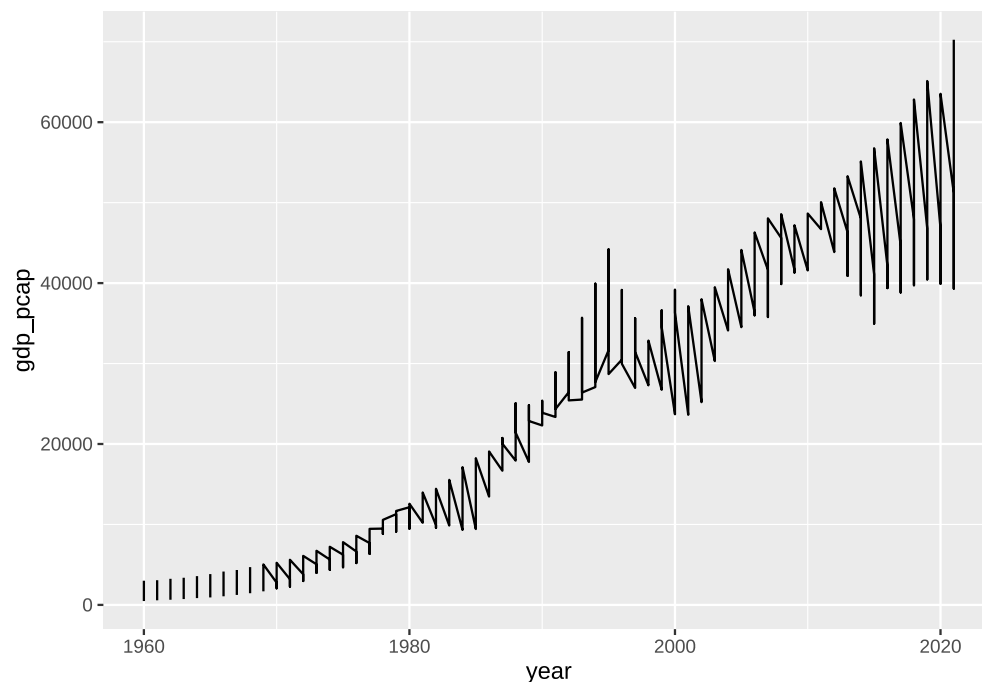
```
df_wdi_gdppcap_small_short <- df_wdi_gdppcap %>% select(country, year, gdp_pcap) %>%
  filter(country %in% c("Japan", "Germany", "United States"))
df_wdi_gdppcap_small_short
#> # A tibble: 186 x 3
#>   country year gdp_pcap
#>   <chr>   <dbl>   <dbl>
#> 1 Germany 2021  51204.
#> 2 Germany 2020  46773.
#> 3 Germany 2019  46794.
#> 4 Germany 2018  47939.
#> 5 Germany 2017  44653.
#> 6 Germany 2016  42136.
#> 7 Germany 2015  41103.
```

```
#> 8 Germany 2014 48024.
#> 9 Germany 2013 46299.
#> 10 Germany 2012 43856.
#> # i 176 more rows
```

31.2.3 可視化 Data Visualization

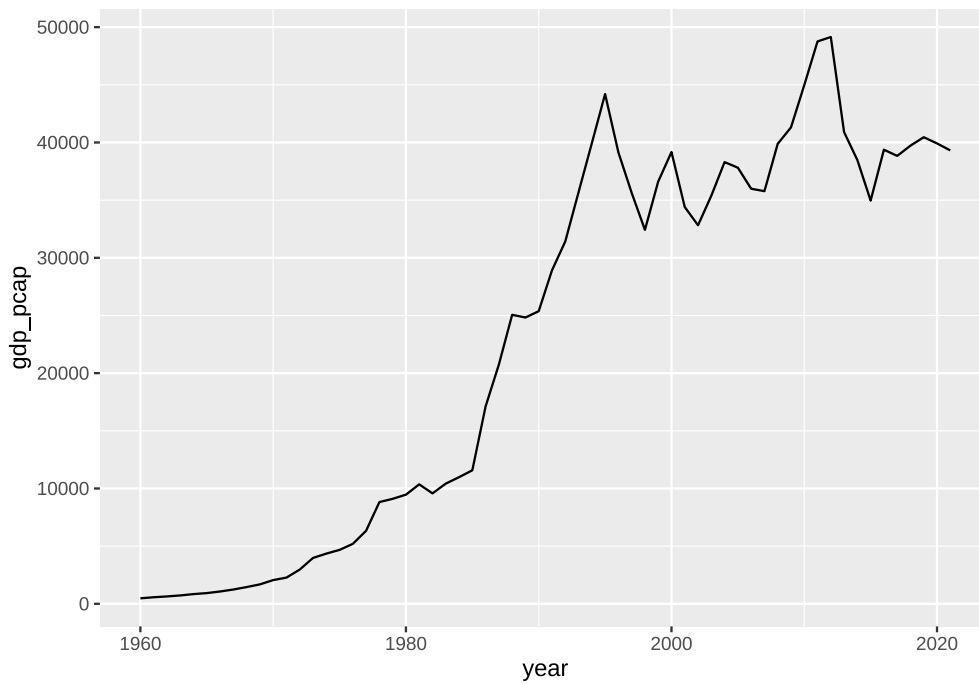
次は、よく生じる、誤りの例で、ノコギリの歯 (sawtoothed) のようなギザギザ・グラフと呼ばれます。なぜこのようなことが起きているかわかりますか。

```
df_wdi_gdppcap_small_short %>%
  ggplot(aes(x = year, y = gdp_pcap)) + geom_line()
#> Warning: Removed 1 row containing missing values
#> (`geom_line()`).
```



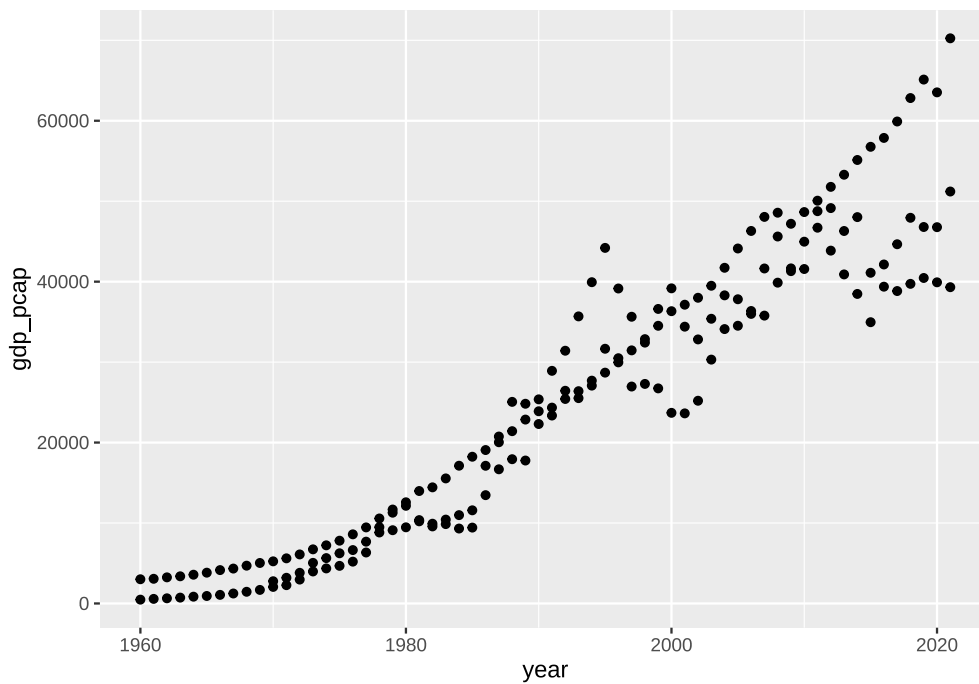
同じ年に、多くのデータがあるので、折れ線グラフを適切に書くことができませんでした。

```
df_wdi_gdppcap_small_short %>% filter(country %in% c("Japan")) %>%
  ggplot(aes(x = year, y = gdp_pcap)) + geom_line()
```



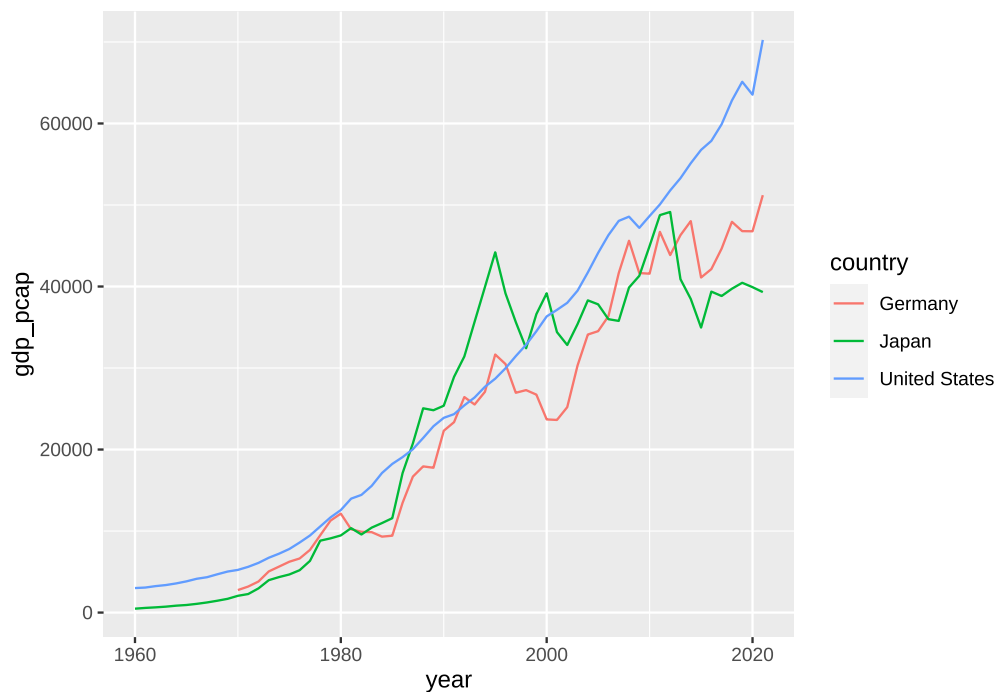
一般的には、散布図をまず、書いてみるのも一つです。

```
df_wdi_gdppcap_small_short %>%  
  ggplot(aes(x = year, y = gdp_pcap)) + geom_point()  
#> Warning: Removed 10 rows containing missing values  
#> (`geom_point()`).
```



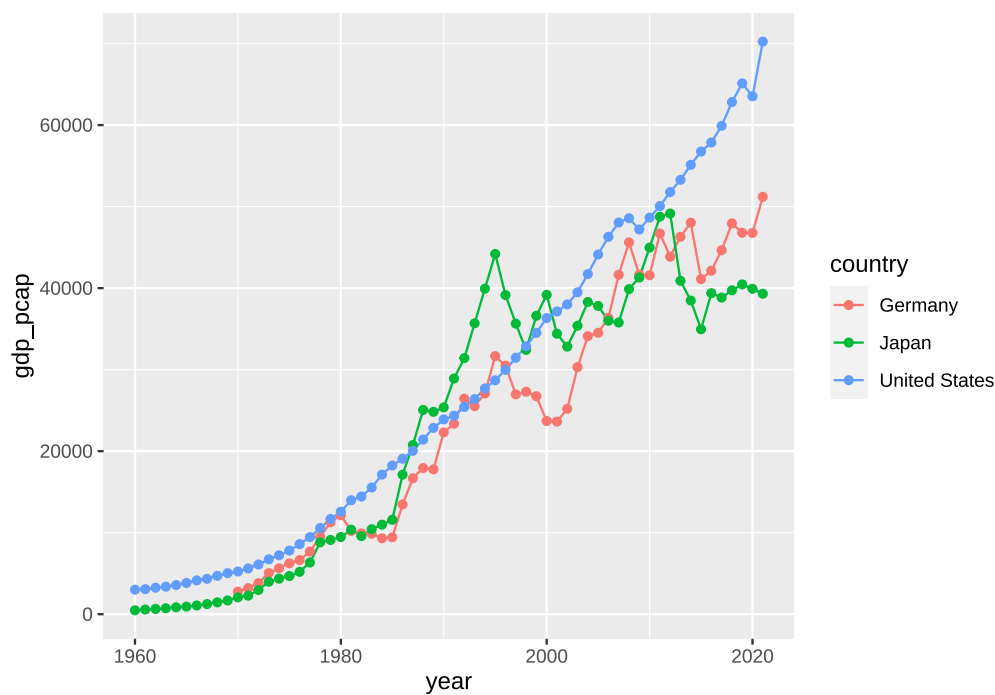
国別に、異なる色を使うことで、折れ線グラフを書くことも可能です。

```
df_wdi_gdppcap_small_short %>% drop_na(gdp_pcap) %>%
  ggplot(aes(x = year, y = gdp_pcap, col = country)) + geom_line()
```



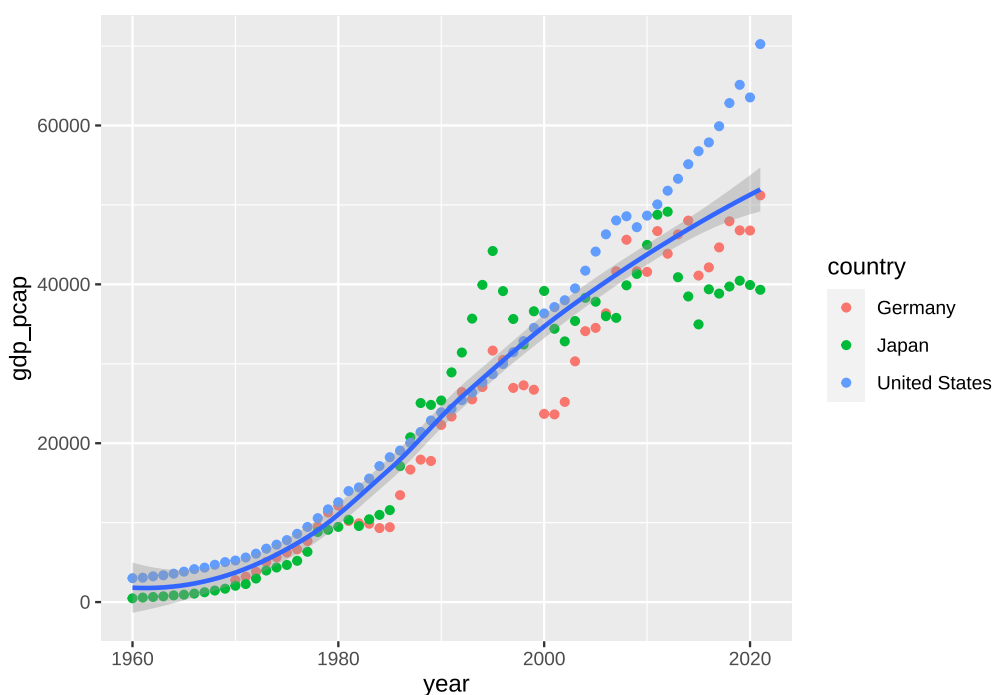
折線グラフと、散布図を同時に描くこともかこのうです。

```
df_wdi_gdppcap_small_short %>% drop_na(gdp_pcap) %>%
  ggplot(aes(x = year, y = gdp_pcap, col = country)) + geom_line() +
  geom_point()
```



点を、曲線で近似する方法はいくつも知られているが、ある幅で、近似していく、LOESS が初期値となっている。method='loess' を省略しても、同じ近似がなされる。span という値を調節することで、ことなる幅での近似曲線を書くことも可能である。初期値は、0.75。

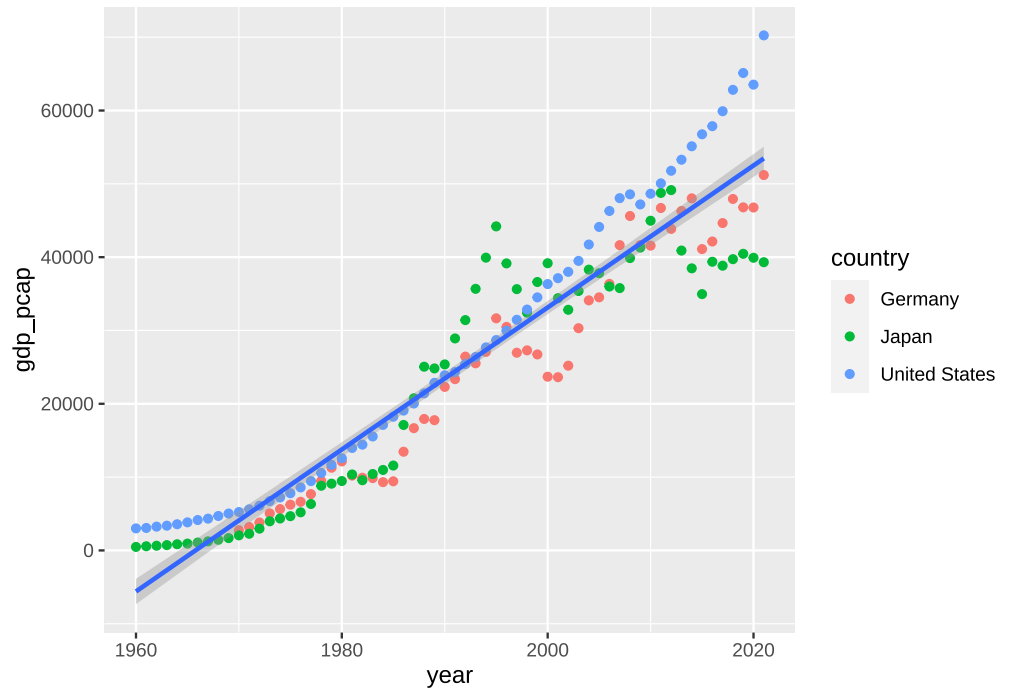
```
df_wdi_gdppcap_small_short %>% drop_na(gdp_pcap) %>%  
  ggplot(aes(x = year, y = gdp_pcap)) +  
  geom_point(aes(color = country)) +  
  geom_smooth(method = 'loess', formula = 'y ~ x')
```



31.2.4 データモデリング Data Modeling

上の例では、曲線ではなく、直線で近似することも考えられる。

```
df_wdi_gdppcap_small_short %>% drop_na(gdp_pcap) %>%  
  ggplot(aes(x = year, y = gdp_pcap)) +  
  geom_point(aes(color = country)) +  
  geom_smooth(method = 'lm', formula = 'y ~ x')
```



簡単な線形回帰モデルでの、回帰直線の y-切片や、傾きは、次のコードで与えられ、p-value や、R squared の値も求められる。

この例では、年とともに、増加の傾向があること。そして、線形モデルが \$\$、90% 程度説明していると表現される。すなわち、

は、良い、近似であることがわかる。

```
df_wdi_gdppcap_small_short %>% lm(gdp_pcap ~ year, .) %>% summary()
#>
#> Call:
#> lm(formula = gdp_pcap ~ year, data = .)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -14156.8  -3200.5  -507.4   3237.7  16779.2
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -1903497.5   48007.9  -39.65  <2e-16 ***
#> year          968.3       24.1   40.18  <2e-16 ***
#> ---
#> Signif. codes:
#> 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 5514 on 174 degrees of freedom
```

```
#> (10 observations deleted due to missingness)
#> Multiple R-squared:  0.9027, Adjusted R-squared:  0.9021
#> F-statistic: 1614 on 1 and 174 DF, p-value: < 2.2e-16
```


第 32 章

地図 Choropleth Maps

この章では、`ggplot2` を使って、情報付きの地図を生成する方法を学ぶ。

最も基本的な `data.frame` を用い `geom_map` または `geom_polygon` で地図を書くことも可能だが、最近では、`simple feature (sf)` 形式のデータを使って `ggplot2` の `geom_sf` を用いて地図を描くことで、複雑なものを扱えるようになってきている。

本書のテーマからも、ある情報を加えた地図は有効だと思われるので、紹介を試みる。

32.1 準備

```
library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.3      v readr      2.1.4
#> v forcats   1.0.0      v stringr   1.5.0
#> v ggplot2   3.4.3      v tibble    3.2.1
#> v lubridate 1.9.2      v tidyr     1.3.0
#> v purrr     1.0.2
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()   masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(sf)
#> Linking to GEOS 3.11.0, GDAL 3.5.3, PROJ 9.1.0; sf_use_s2()
#> is TRUE
library(showtext)
#> Loading required package: sysfonts
#> Loading required package: showtextdb
showtext_auto()
```

WDI パッケージの、`WDIcache()` に含まれる、収入の多寡を表す `income` レベルと `iso2c`

コードをまず用いる。

```
wdi_cache <- read_rds("./data/wdi_cache.RData")
```

```
wdi_cache$country %>% as_tibble() %>% glimpse()
#> Rows: 299
#> Columns: 9
#> $ iso3c      <chr> "ABW", "AFE", "AFG", "AFR", "AFW", "AGO"~
#> $ iso2c      <chr> "AW", "ZH", "AF", "A9", "ZI", "AO", "AL"~
#> $ country    <chr> "Aruba", "Africa Eastern and Southern", ~
#> $ region     <chr> "Latin America & Caribbean", "Aggregates~
#> $ capital    <chr> "Oranjestad", "", "Kabul", "", "", "Luan~
#> $ longitude  <chr> "-70.0167", "", "69.1761", "", "", "13.2~
#> $ latitude   <chr> "12.5167", "", "34.5228", "", "", "-8.81~
#> $ income     <chr> "High income", "Aggregates", "Low income~
#> $ lending    <chr> "Not classified", "Aggregates", "IDA", "~
```

```
wdi_income <- wdi_cache$country %>%
  filter(region != "Aggregates") %>%
  select(iso2c, income) %>%
  drop_na(iso2c) %>%
  mutate(income = factor(income, levels = c("High income", "Upper middle income",
glimpse(wdi_income)
#> Rows: 218
#> Columns: 2
#> $ iso2c      <chr> "AW", "AF", "AO", "AL", "AD", "AE", "AR", "~
#> $ income     <fct> High income, Low income, Lower middle incom~
```

32.1.1 geom_sf

See also, `coord_sf`, `geom_sf_label`, `geom_sf_text`, `stat_sf`.

This set of `geom`, `stat`, and `coord` are used to visualise simple feature (sf) objects. For simple plots, you will only need `geom_sf()` as it uses `stat_sf()` and adds `coord_sf()` for you. `geom_sf()` is an unusual geom because it will draw different geometric objects depending on what simple features are present in the data: you can get points, lines, or polygons. For text and labels, you can use `geom_sf_text()` and `geom_sf_label()`.

この `geom`、`stat`、および `coord` のセットは、Simple Feature (sf) 形式のデータを視覚化するために使用されます。単純なプロットでは、`stat_sf()` を使用し、`coord_sf()` を追加するので、`geom_sf()` だけが必要です。`geom_sf()` は、データにどのような単純な特徴が存在するかによって異なる幾何オブジェクトを描画するので、珍しい geom です。テキストとラベルについては、`geom_sf_text()` と `geom_sf_label()`

が使用可能です。

```
geom_sf(  
  mapping = aes(),  
  data = NULL,  
  stat = "sf",  
  position = "identity",  
  na.rm = FALSE,  
  show.legend = NA,  
  inherit.aes = TRUE,  
  ...  
)
```

32.2 Natural Earth Data

<https://www.naturalearthdata.com>

このサイトで、データを提供しています。

Manual: <https://cran.r-project.org/web/packages/rnaturalearth/rnaturalearth.pdf>

32.2.1 ne_countries

```
library(rnaturalearth)  
#> The legacy packages mapproj, rgdal, and rgeos, underpinning the sp package,  
#> which was just loaded, will retire in October 2023.  
#> Please refer to R-spatial evolution reports for details, especially  
#> https://r-spatial.org/r/2023/05/15/evolution4.html.  
#> It may be desirable to make the sf package available;  
#> package maintainers should consider adding sf to Suggests:.  
#> The sp package is now running under evolution status 2  
#> (status 2 uses the sf package in place of rgdal)  
#> Support for Spatial objects (`sp`) will be deprecated in {rnaturalearth} and will be removed in a future  
library(rnaturalearthdata)  
#>  
#> Attaching package: 'rnaturalearthdata'  
#> The following object is masked from 'package:rnaturalearth':  
#>  
#> countries110
```

```
ne_countries(  
  scale = 110,  
  type = "countries",
```

```
continent = NULL,  
country = NULL,  
geounit = NULL,  
sovereignty = NULL,  
returnclass = c("sp", "sf")  
)
```

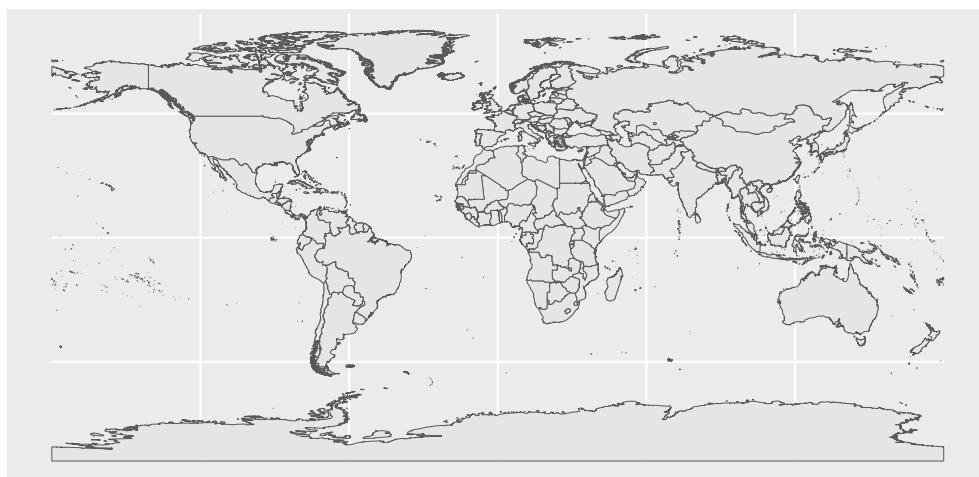
32.2.1.1 Arguments

- scale: scale of map to return, one of 110, 50, 10 or 'small', 'medium', 'large'
- type: country type, one of 'countries', 'map_units', 'sovereignty', 'tiny_countries'
- continent: a character vector of continent names to get countries from.
- country: a character vector of country names.
- geounit: a character vector of geounit names.
- sovereignty: a character vector of sovereignty names.
- returnclass: 'sp' default or 'sf' for Simple Features

32.2.1.2 例

There are three scales. Add `returnclass = "sf"` as an option to obtain data in `sf` format.

```
ne_countries(scale = "large", returnclass = "sf") %>%  
  ggplot() + geom_sf()
```



```
ne_countries(scale = "small", returnclass = "sf") %>%  
  ggplot() + geom_sf()
```



We will use medium scale data in the following.

```
ne_world <- ne_countries(scale = "medium", returnclass = "sf")
```

```
glimpse(ne_world)
```

```
#> Rows: 241
#> Columns: 64
#> $ scalerank <int> 3, 1, 1, 1, 1, 3, 3, 1, 1, 1, 3, 1, 5, ~
#> $ featurecla <chr> "Admin-0 country", "Admin-0 country", "~
#> $ labelrank <dbl> 5, 3, 3, 6, 6, 6, 6, 4, 2, 6, 4, 4, 5, ~
#> $ sovereignt <chr> "Netherlands", "Afghanistan", "Angola", ~
#> $ sov_a3 <chr> "NL1", "AFG", "AGO", "GB1", "ALB", "FI1~
#> $ adm0_dif <dbl> 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, ~
#> $ level <dbl> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
#> $ type <chr> "Country", "Sovereign country", "Sovere~
#> $ admin <chr> "Aruba", "Afghanistan", "Angola", "Angu~
#> $ adm0_a3 <chr> "ABW", "AFG", "AGO", "AIA", "ALB", "ALD~
#> $ geou_dif <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
#> $ geounit <chr> "Aruba", "Afghanistan", "Angola", "Angu~
#> $ gu_a3 <chr> "ABW", "AFG", "AGO", "AIA", "ALB", "ALD~
#> $ su_dif <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
#> $ subunit <chr> "Aruba", "Afghanistan", "Angola", "Angu~
#> $ su_a3 <chr> "ABW", "AFG", "AGO", "AIA", "ALB", "ALD~
#> $ brk_diff <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
#> $ name <chr> "Aruba", "Afghanistan", "Angola", "Angu~
#> $ name_long <chr> "Aruba", "Afghanistan", "Angola", "Angu~
#> $ brk_a3 <chr> "ABW", "AFG", "AGO", "AIA", "ALB", "ALD~
#> $ brk_name <chr> "Aruba", "Afghanistan", "Angola", "Angu~
#> $ brk_group <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ abbrev <chr> "Aruba", "Afg.", "Ang.", "Ang.", "Alb."~
```

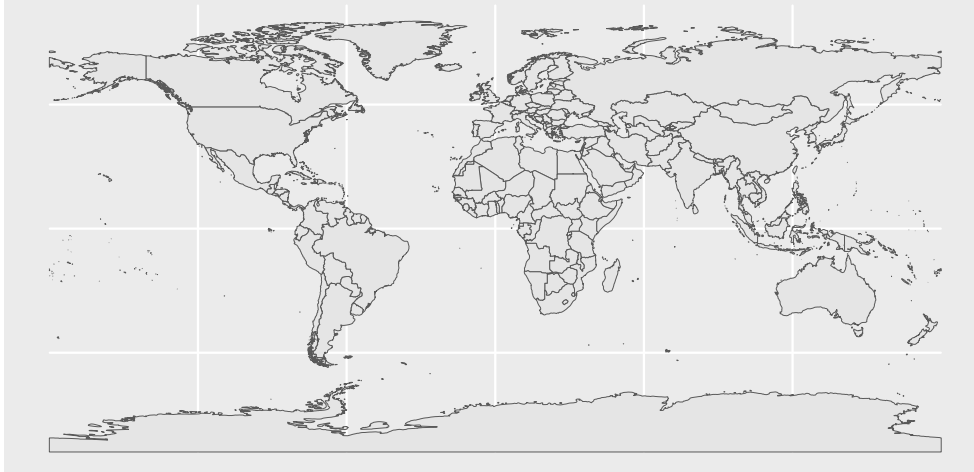
```

#> $ postal      <chr> "AW", "AF", "AO", "AI", "AL", "AI", "AN~
#> $ formal_en  <chr> "Aruba", "Islamic State of Afghanistan"~
#> $ formal_fr  <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ note_adm0  <chr> "Neth.", NA, NA, "U.K.", NA, "Fin.", NA~
#> $ note_brk   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ name_sort  <chr> "Aruba", "Afghanistan", "Angola", "Angu~
#> $ name_alt   <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ mapcolor7  <dbl> 4, 5, 3, 6, 1, 4, 1, 2, 3, 3, 4, 4, 1, ~
#> $ mapcolor8  <dbl> 2, 6, 2, 6, 4, 1, 4, 1, 1, 1, 5, 5, 2, ~
#> $ mapcolor9  <dbl> 2, 8, 6, 6, 1, 4, 1, 3, 3, 2, 1, 1, 2, ~
#> $ mapcolor13 <dbl> 9, 7, 1, 3, 6, 6, 8, 3, 13, 10, 1, NA, ~
#> $ pop_est    <dbl> 103065, 28400000, 12799293, 14436, 3639~
#> $ gdp_md_est <dbl> 2258.0, 22270.0, 110300.0, 108.9, 21810~
#> $ pop_year   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ lastcensus <dbl> 2010, 1979, 1970, NA, 2001, NA, 1989, 2~
#> $ gdp_year   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ economy    <chr> "6. Developing region", "7. Least devel~
#> $ income_grp <chr> "2. High income: nonOECD", "5. Low inco~
#> $ wikipedia  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ fips_10    <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ iso_a2     <chr> "AW", "AF", "AO", "AI", "AL", "AX", "AD~
#> $ iso_a3     <chr> "ABW", "AFG", "AGO", "AIA", "ALB", "ALA~
#> $ iso_n3     <chr> "533", "004", "024", "660", "008", "248~
#> $ un_a3      <chr> "533", "004", "024", "660", "008", "248~
#> $ wb_a2      <chr> "AW", "AF", "AO", NA, "AL", NA, "AD", "~
#> $ wb_a3      <chr> "ABW", "AFG", "AGO", NA, "ALB", NA, "AD~
#> $ woe_id     <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ adm0_a3_is <chr> "ABW", "AFG", "AGO", "AIA", "ALB", "ALA~
#> $ adm0_a3_us <chr> "ABW", "AFG", "AGO", "AIA", "ALB", "ALD~
#> $ adm0_a3_un <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ adm0_a3_wb <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ continent  <chr> "North America", "Asia", "Africa", "Nor~
#> $ region_un  <chr> "Americas", "Asia", "Africa", "Americas~
#> $ subregion  <chr> "Caribbean", "Southern Asia", "Middle A~
#> $ region_wb  <chr> "Latin America & Caribbean", "South Asi~
#> $ name_len   <dbl> 5, 11, 6, 8, 7, 5, 7, 20, 9, 7, 14, 10, ~
#> $ long_len   <dbl> 5, 11, 6, 8, 7, 13, 7, 20, 9, 7, 14, 10~
#> $ abbrev_len <dbl> 5, 4, 4, 4, 4, 5, 4, 6, 4, 4, 9, 4, 7, ~
#> $ tiny       <dbl> 4, NA, NA, NA, NA, NA, 5, 5, NA, NA, NA, 3, ~
#> $ homepart   <dbl> NA, 1, 1, NA, 1, NA, 1, 1, 1, 1, NA, 1, ~
#> $ geometry   <MULTIPOLYGON [°]> MULTIPOLYGON (((-69.89912 ~

```

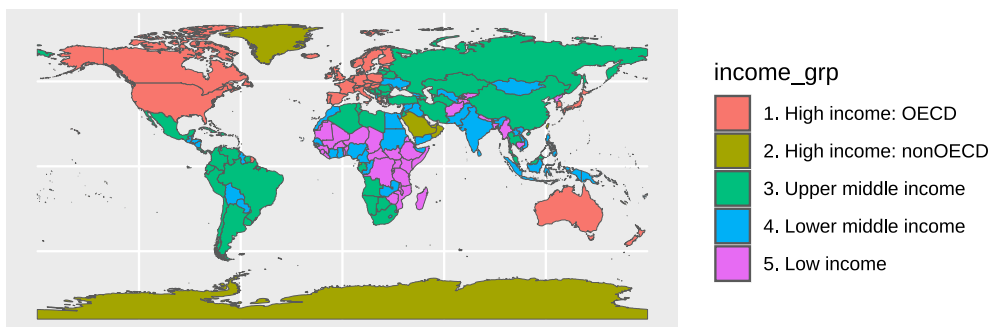
The last column is the geometry which contains map data in multi-polygon format.

```
ne_world %>% ggplot() + geom_sf()
```



This map data comes with various information.

```
ne_world %>% ggplot() + geom_sf(aes(fill = income_grp))
```



You can specify a 'continent', a 'region_un', a 'subregion' or 'region_wb'.

```
ne_world %>% as_tibble() %>% distinct(continent) %>% pull()
```

```
#> [1] "North America"      "Asia"
#> [3] "Africa"             "Europe"
#> [5] "South America"     "Oceania"
#> [7] "Antarctica"        "Seven seas (open ocean)"
```

```
ne_world %>% as_tibble() %>% distinct(region_un) %>% pull()
```

```
#> [1] "Americas"          "Asia"
#> [3] "Africa"            "Europe"
#> [5] "Oceania"           "Antarctica"
#> [7] "Seven seas (open ocean)"
```

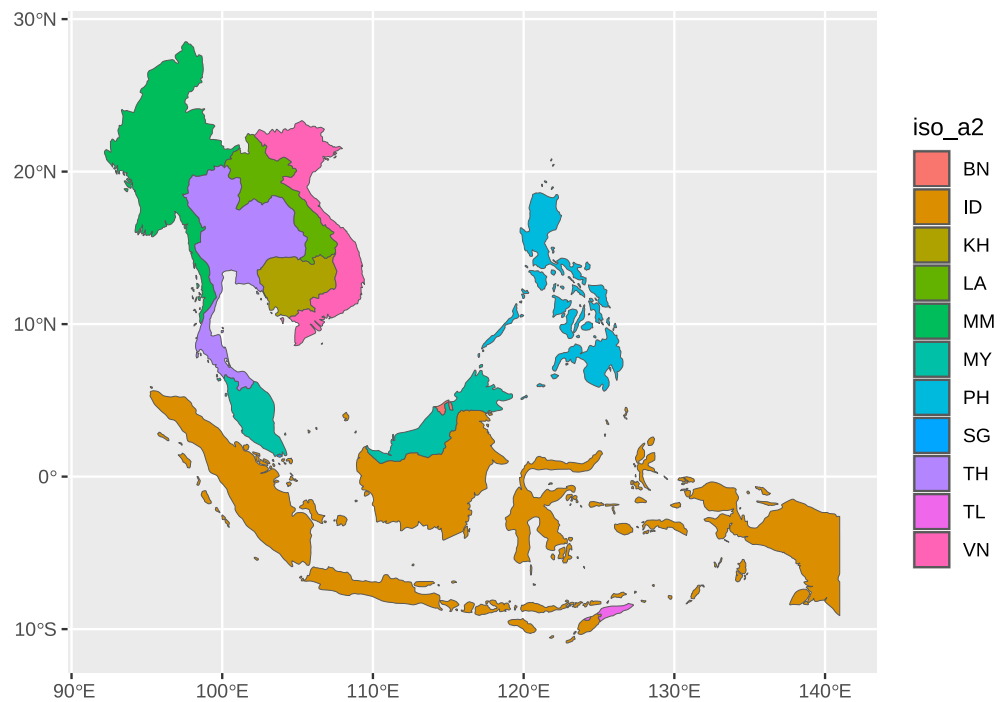
```
ne_world %>% as_tibble() %>% distinct(subregion) %>% pull()
```

```
#> [1] "Caribbean"         "Southern Asia"
#> [3] "Middle Africa"      "Southern Europe"
```

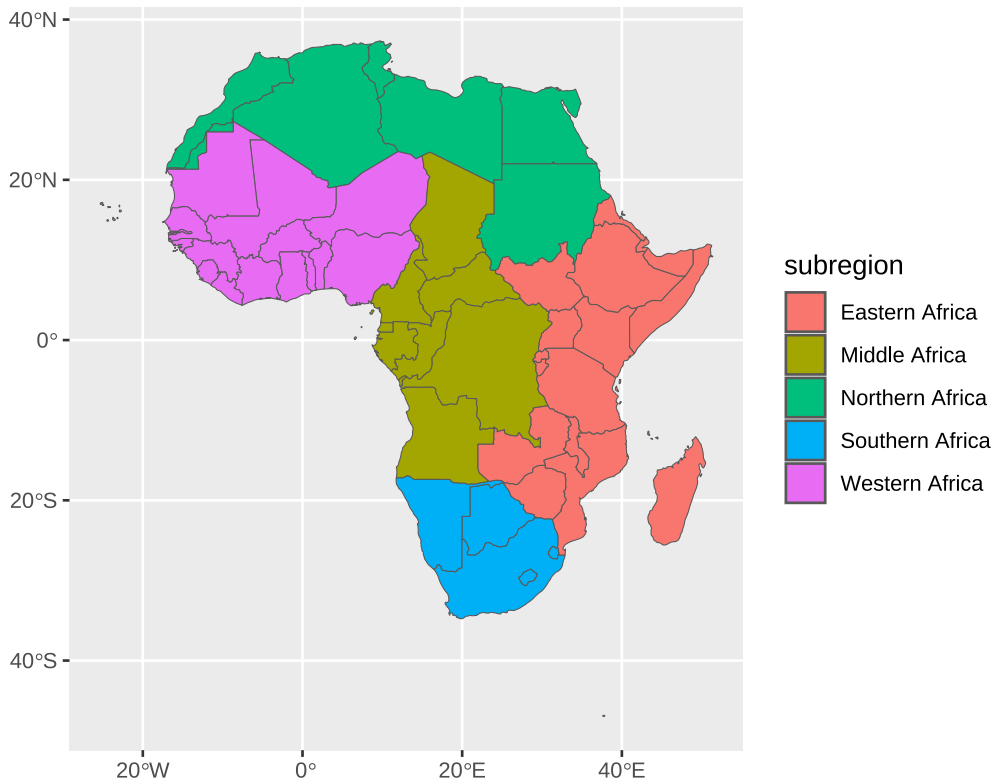
```
#> [5] "Northern Europe"      "Western Asia"
#> [7] "South America"       "Polynesia"
#> [9] "Antarctica"         "Australia and New Zealand"
#> [11] "Seven seas (open ocean)" "Western Europe"
#> [13] "Eastern Africa"      "Western Africa"
#> [15] "Eastern Europe"      "Central America"
#> [17] "Northern America"    "South-Eastern Asia"
#> [19] "Southern Africa"     "Eastern Asia"
#> [21] "Northern Africa"     "Melanesia"
#> [23] "Micronesia"         "Central Asia"
```

```
ne_world %>% as_tibble() %>% distinct(region_wb) %>% pull()
#> [1] "Latin America & Caribbean"
#> [2] "South Asia"
#> [3] "Sub-Saharan Africa"
#> [4] "Europe & Central Asia"
#> [5] "Middle East & North Africa"
#> [6] "East Asia & Pacific"
#> [7] "Antarctica"
#> [8] "North America"
```

```
ne_world %>% filter(subregion == "South-Eastern Asia") %>%
  ggplot() + geom_sf(aes(fill = iso_a2))
```

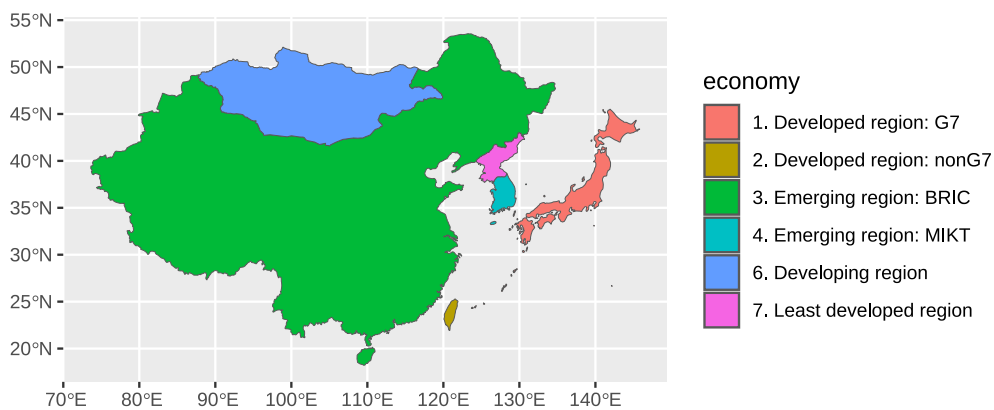


```
ne_world %>% filter(continent == 'Africa') %>%
  ggplot() + geom_sf(aes(fill = subregion))
```

32.2.1.3 type argument

```
ne_countries(type = "countries", country = c("Japan", "South Korea", "North Korea", "China", "Taiwan", "
  ggplot() + geom_sf(aes(fill = economy))
```



32.2.2 ne_states

Get natural earth world state (admin level 1) polygons

Description: returns state polygons (administrative level 1) for specified countries

32.2.2.1 Usage

```
ne_states(
  country = NULL,
  geounit = NULL,
  iso_a2 = NULL,
  spdf = NULL,
  returnclass = c("sp", "sf")
)
```

32.2.3 Arguments

- country: a character vector of country names.
- geounit: a character vector of geounit names.
- iso_a2: a character vector of iso_a2 country codes
- spdf: an optional alternative states map
- returnclass: 'sp' default or 'sf' for Simple Features

32.2.4 Value

- SpatialPolygons DataFrame or sf

```
ne_world_admin1 <- ne_states(returnclass = "sf")
```

```
glimpse(ne_world_admin1)
```

```
#> Rows: 4,596
#> Columns: 122
#> $ featurecla <chr> "Admin-1 states provinces lakes", "Admi~
#> $ scalerank <int> 3, 6, 2, 6, 3, 4, 4, 3, 4, 3, 4, 3, 3, ~
#> $ adm1_code <chr> "ARG-1309", "URY-8", "IDN-1185", "MYS-1~
#> $ diss_me <int> 1309, 8, 1185, 1186, 2694, 1936, 1937, ~
#> $ iso_3166_2 <chr> "AR-E", "UY-PA", "ID-KI", "MY-12", "CL--
#> $ wikipedia <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ iso_a2 <chr> "AR", "UY", "ID", "MY", "CL", "BO", "BO~
#> $ adm0_sr <int> 1, 1, 5, 5, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
#> $ name <chr> "Entre Ríos", "Paysandú", "Kalimantan T~
#> $ name_alt <chr> "Entre-Rios", NA, "Kaltim", "North Born~
#> $ name_local <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ type <chr> "Provincia", "Departamento", "Propinsi"~
#> $ type_en <chr> "Province", "Department", "Province", "~
```

```

#> $ code_local <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ code_hasc <chr> "AR.ER", "UY.PA", "ID.KI", "MY.SA", "CL~
#> $ note <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ hasc_maybe <chr> NA, NA, NA, NA, NA, NA, NA, NA, "BO.OR/~
#> $ region <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ region_cod <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ provnum_ne <int> 10, 19, 15, 1, 0, 8, 7, 0, 1, 20006, 8, ~
#> $ gadm_level <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
#> $ check_me <int> 20, 0, 20, 20, 20, 0, 0, 20, 10, 20, 10~
#> $ datarank <int> 3, 8, 1, 6, 3, 8, 6, 3, 6, 3, 5, 3, 3, ~
#> $ abbrev <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ postal <chr> "ER", "PA", "KI", "SA", NA, "LP", "OR", ~
#> $ area_sqkm <int> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ~
#> $ sameascity <int> -99, -99, -99, -99, 7, 6, 6, -99, -99, ~
#> $ labelrank <int> 3, 6, 2, 6, 7, 6, 6, 3, 4, 6, 7, 6, 3, ~
#> $ name_len <int> 10, 8, 16, 5, 18, 6, 5, 8, 6, 11, 5, 5, ~
#> $ mapcolor9 <int> 3, 2, 6, 3, 5, 2, 2, 5, 2, 5, 4, 3, 3, ~
#> $ mapcolor13 <int> 13, 10, 11, 6, 9, 3, 3, 9, 3, 9, 11, 13~
#> $ fips <chr> "AR08", "UY11", "ID14", "MY16", NA, "BL~
#> $ fips_alt <chr> NA, NA, NA, NA, NA, NA, NA, NA, "BL05", ~
#> $ woe_id <int> 2344682, 2347650, 2345723, 2346310, 560~
#> $ woe_label <chr> "Entre Rios, AR, Argentina", "PaysandÃ°~
#> $ woe_name <chr> "Entre Ríos", "Paysandú", "Kalimantan T~
#> $ latitude <dbl> -32.02750, -32.09330, 1.28915, 5.31115, ~
#> $ longitude <dbl> -59.28240, -57.22400, 116.35400, 117.09~
#> $ sov_a3 <chr> "ARG", "URY", "IDN", "MYS", "CHL", "BOL~
#> $ adm0_a3 <chr> "ARG", "URY", "IDN", "MYS", "CHL", "BOL~
#> $ adm0_label <int> 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, ~
#> $ admin <chr> "Argentina", "Uruguay", "Indonesia", "M~
#> $ geonunit <chr> "Argentina", "Uruguay", "Indonesia", "M~
#> $ gu_a3 <chr> "ARG", "URY", "IDN", "MYS", "CHL", "BOL~
#> $ gn_id <int> 3434137, 3441242, 1641897, 1733039, 669~
#> $ gn_name <chr> "Provincia de Entre Rios", "Departament~
#> $ gns_id <int> -988655, -908097, -2680740, -2405166, 1~
#> $ gns_name <chr> "Entre Rios", "Paysandu, Departamento d~
#> $ gn_level <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
#> $ gn_region <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ gn_a1_code <chr> "AR.08", "UY.11", "ID.14", "MY.16", "CL~
#> $ region_sub <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ sub_code <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ gns_level <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~

```

```

#> $ gns_lang <chr> "khm", "fra", "ind", "fil", "ara", "kor~
#> $ gns_adm1 <chr> "AR08", "UY11", "ID14", "MY16", "CI16",~
#> $ gns_region <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ min_label <dbl> 6.0, 8.0, 5.0, 7.0, 6.0, 6.6, 6.6, 6.0,~
#> $ max_label <dbl> 11.0, 11.0, 10.1, 11.0, 11.0, 11.0, 11.~
#> $ min_zoom <dbl> 6.0, 8.0, 4.6, 7.0, 6.0, 6.6, 6.6, 6.0,~
#> $ wikidataid <chr> "Q44762", "Q16576", "Q3899", "Q179029",~
#> $ name_ar <chr> " " , " " , ~ "
<# $ name_bn <chr> " " , " " , " " ~
#> $ name_de <chr> "Entre Ríos", "Paysandú", "Ostkalimanta~
#> $ name_en <chr> "Entre Ríos", "Paysandú", "East Kaliman~
#> $ name_es <chr> "Entre Ríos", "Paysandú", "Kalimantan O~
#> $ name_fr <chr> "Entre Ríos", "Paysandú", "Kalimantan o~
#> $ name_el <chr> "Ε Ρ " , "Π " , "Α Κ~
#> $ name_hi <chr> " " , " " , " " ~
#> $ name_hu <chr> "Entre Ríos", "Paysandú", "Kelet-Kalima~
#> $ name_id <chr> "Entre Ríos", "Departemen Paysandú", "K~
#> $ name_it <chr> "Entre Ríos", "dipartimento di Paysandú~
#> $ name_ja <chr> " エントレ・リオス州", " パイサントゥ県",~
#> $ name_ko <chr> " 엔트레리오스주", " 파사두주", " 칼리만탄주 ~
#> $ name_nl <chr> "Entre Ríos", "Paysandú", "Oost-Kaliman~
#> $ name_pl <chr> "Entre Ríos", "Paysandú", "Borneo Wscho~
#> $ name_pt <chr> "Entre Ríos", "Paysandú", "Kalimantan O~
#> $ name_ru <chr> " - " , " " , " " ~
#> $ name_sv <chr> "Entre Ríos", "Paysandú", "Kalimantan T~
#> $ name_tr <chr> "Entre Ríos eyaleti", "Paysandu Departm~
#> $ name_vi <chr> "Entre Ríos", "Paysandú", "Đông Kaliman~
#> $ name_zh <chr> " 恩特雷里奧斯省", " 派桑杜省", " 加里曼 ~
#> $ ne_id <dbl> 1159309789, 1159307733, 1159310009, 115~
#> $ name_he <chr> " " , " " , " " ~
<# $ name_uk <chr> " - " , " " , " " ~
#> $ name_ur <chr> " " , " " , " " ~
<# $ name_fa <chr> " " , " " , " " ~
<# $ name_zht <chr> " 恩特雷里奧斯省", " 派桑杜省", " 東加里曼 ~
#> $ FCLASS_ISO <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ FCLASS_US <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ FCLASS_FR <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ FCLASS_RU <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ FCLASS_ES <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ FCLASS_CN <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~
#> $ FCLASS_TW <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA,~

```

```

#> $ FCLASS_IN <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_NP <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_PK <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_DE <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_GB <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_BR <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_IL <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_PS <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_SA <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_EG <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_MA <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_PT <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_AR <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_JP <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_KO <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_VN <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_TR <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_ID <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_PL <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_GR <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_IT <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_NL <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_SE <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_BD <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_UA <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ FCLASS_TLC <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ geometry <MULTIPOLYGON [°]> MULTIPOLYGON (((-58.20011 ~

```

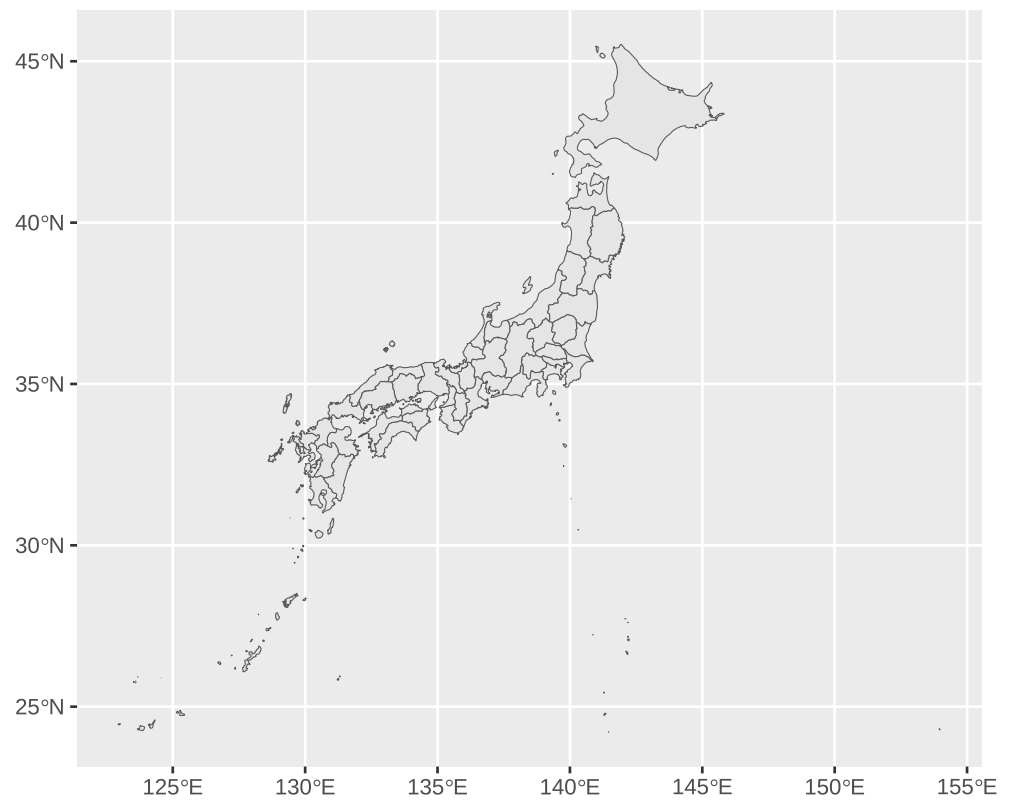
```

ne_world_admin1 %>% as_tibble() %>%
  filter(iso_a2 != "-1") %>% arrange(admin) %>%
  distinct(iso_a2, admin)
#> # A tibble: 243 x 2
#>   iso_a2 admin
#>   <chr> <chr>
#> 1 AF      Afghanistan
#> 2 AX      Aland
#> 3 AL      Albania
#> 4 DZ      Algeria
#> 5 AS      American Samoa
#> 6 AD      Andorra
#> 7 AO      Angola

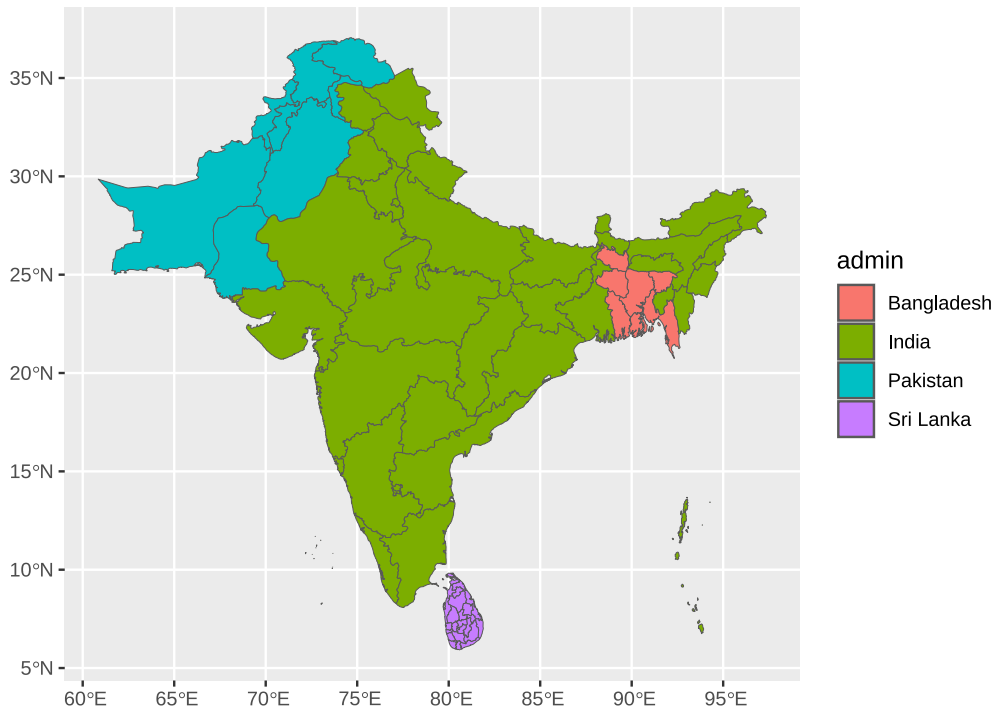
```

```
#> 8 AI Anguilla
#> 9 AQ Antarctica
#> 10 AG Antigua and Barbuda
#> # i 233 more rows
```

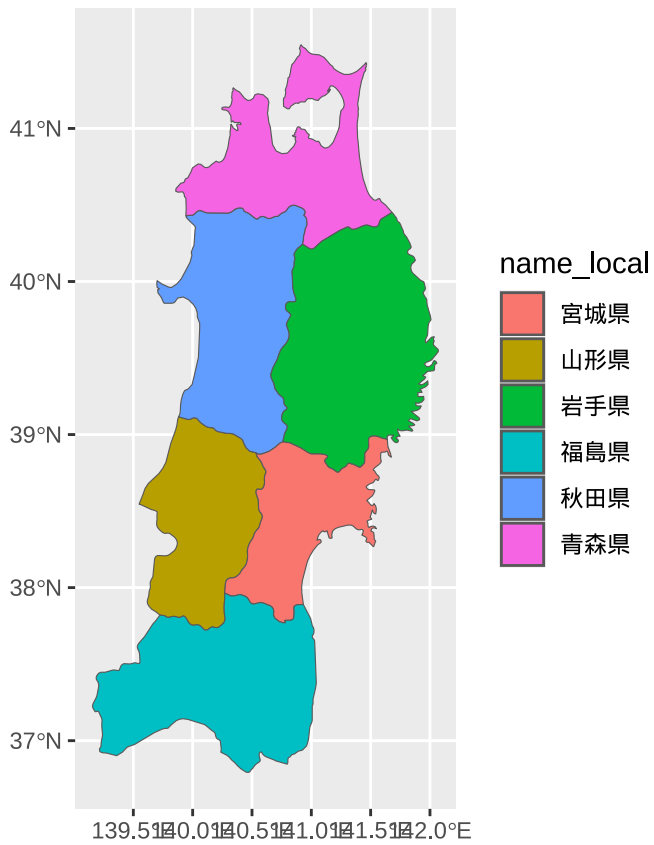
```
country <- "Japan"
ne_world_admin1 %>% filter(admin == country) %>%
  ggplot() + geom_sf()
```



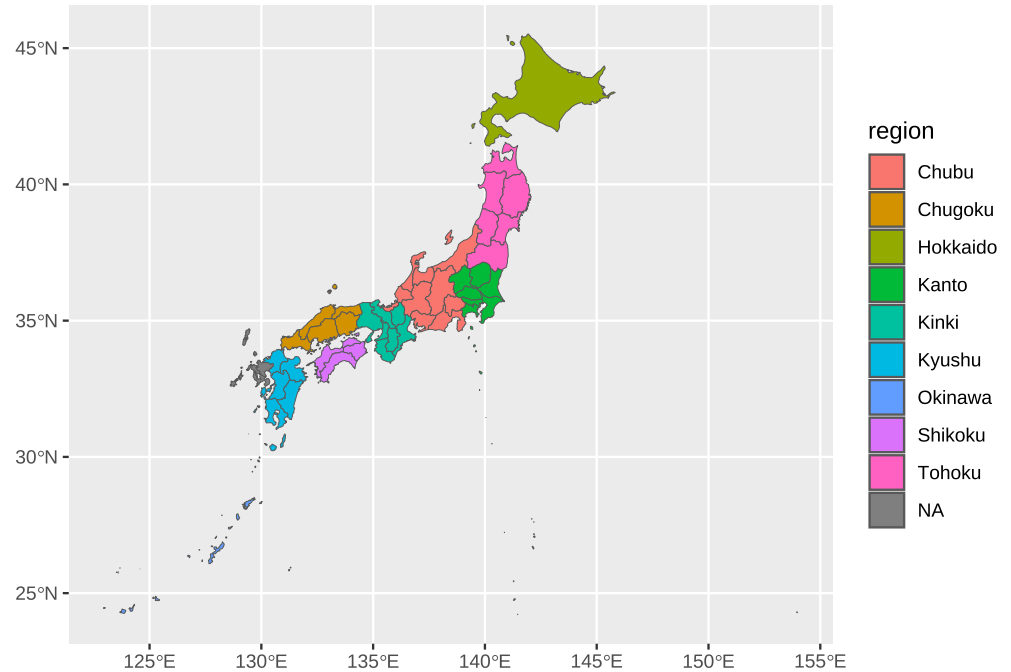
```
iso2s <- c("IN", "PK", "BD", "LK")
ne_world_admin1 %>% filter(iso_a2 %in% iso2s) %>%
  ggplot() + geom_sf(aes(fill = admin))
```



```
regions <- c("Tohoku")
ne_world_admin1 %>% filter(region %in% regions) %>%
  ggplot() + geom_sf(aes(fill = name_local))
```



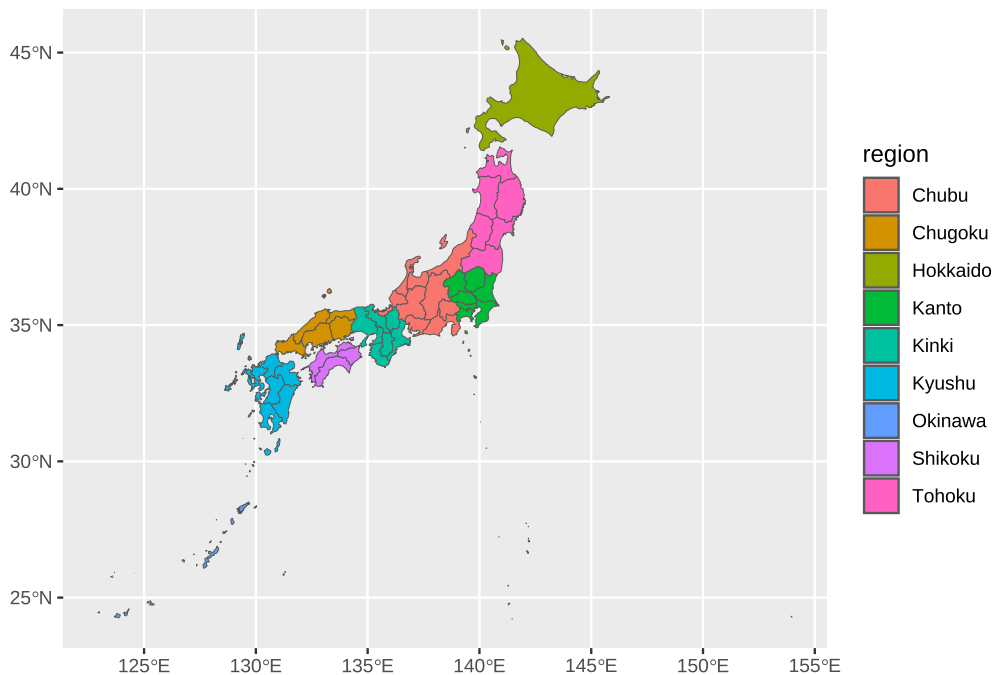
```
ne_world_admin1 %>% filter(iso_a2 == "JP") %>%
  ggplot() + geom_sf(aes(fill = region))
```



```
ne_world_admin1 %>% as_tibble() %>% filter(admin %in% "Japan") %>%
  select(name_local, region) %>% filter(is.na(region))
#> # A tibble: 2 x 2
#>   name_local region
#>   <chr>      <chr>
#> 1 佐賀県      <NA>
#> 2 長崎県      <NA>
```

```
ne_world_admin1 %>% mutate(region = case_when(
  name_local == "佐賀県" ~ "Kyushu",
  name_local == "長崎県" ~ "Kyushu",
  TRUE ~ region)) %>%
  as_tibble() %>% filter(admin %in% "Japan") %>%
  select(name_local, region) %>% filter(is.na(region))
#> # A tibble: 0 x 2
#> # i 2 variables: name_local <chr>, region <chr>
```

```
ne_world_admin1 %>% mutate(region = case_when(
  name_local == "佐賀県" ~ "Kyushu",
  name_local == "長崎県" ~ "Kyushu",
  TRUE ~ region)) %>%
  filter(iso_a2 == "JP") %>%
  ggplot() + geom_sf(aes(fill = region))
```

32.3 geodata Package

- URL: <https://gadm.org>
- `geodata`: Download Geographic Data
- Functions for downloading of geographic data for use in spatial analysis and mapping. The package facilitates access to climate, crops, elevation, land use, soil, species occurrence, accessibility, administrative boundaries and other data.
- Package Link: <https://CRAN.R-project.org/package=geodata>
- Manual: <https://cran.r-project.org/web/packages/geodata/geodata.pdf>

```
library(geodata)
```

```
#> Loading required package: terra
#> terra 1.7.46
#>
#> Attaching package: 'terra'
#> The following object is masked from 'package:tidyr':
#>
#>   extract
```

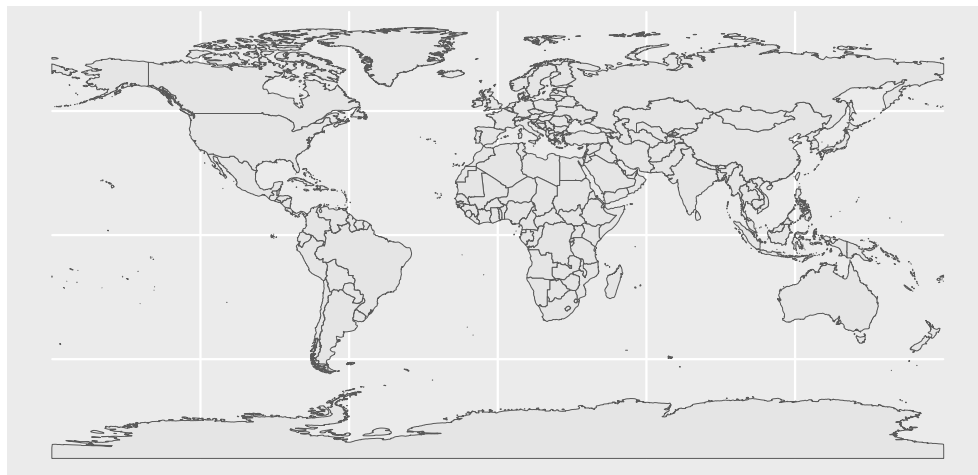
```
world(resolution=5, level=0, path = "./data")
```

```
#> class      : SpatVector
#> geometry   : polygons
#> dimensions  : 231, 2 (geometries, attributes)
#> extent     : -180, 180, -90, 83.65625 (xmin, xmax, ymin, ymax)
#> coord. ref. : +proj=longlat +datum=WGS84 +no_defs
```

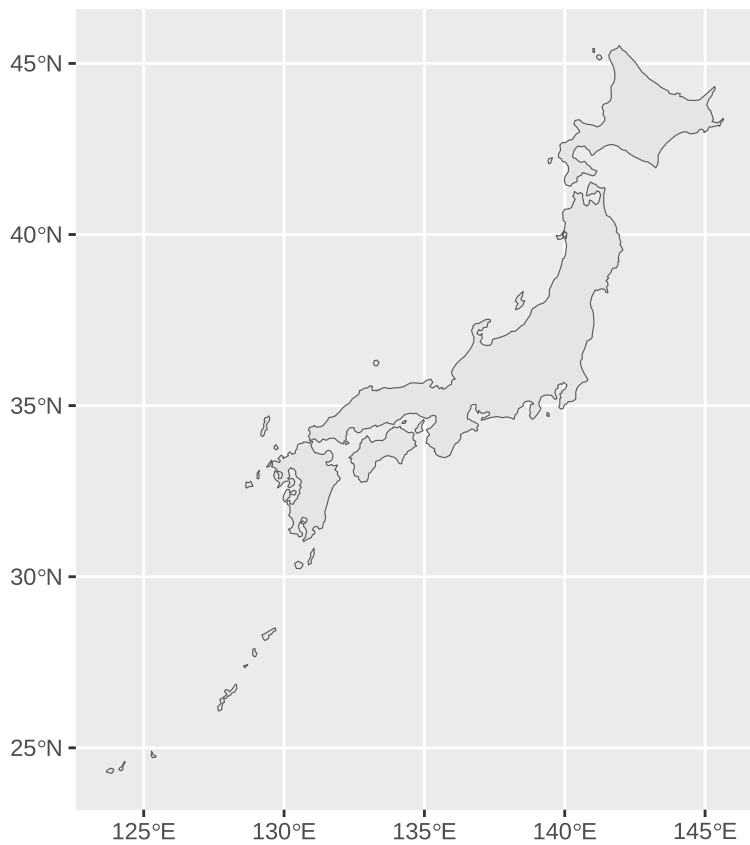
```
#> names      : GID_0      NAME_0
#> type       : <chr>      <chr>
#> values     : ABW       Aruba
#>           : AFG       Afghanistan
#>           : AGO       Angola
```

```
world5 <- readRDS("../data/gadm/gadm36_adm0_r5_pk.rds")
world5 %>% as_tibble() %>% glimpse()
#> Rows: 231
#> Columns: 2
#> $ GID_0 <chr> "ABW", "AFG", "AGO", "ALA", "ALB", "AND", "~
#> $ NAME_0 <chr> "Aruba", "Afghanistan", "Angola", "Åland", ~
```

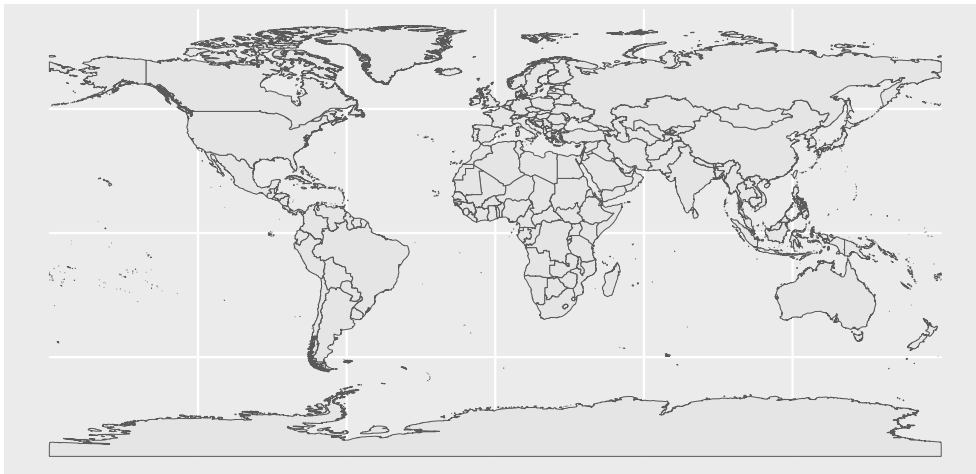
```
world5 %>% st_as_sf() %>% ggplot() + geom_sf()
```



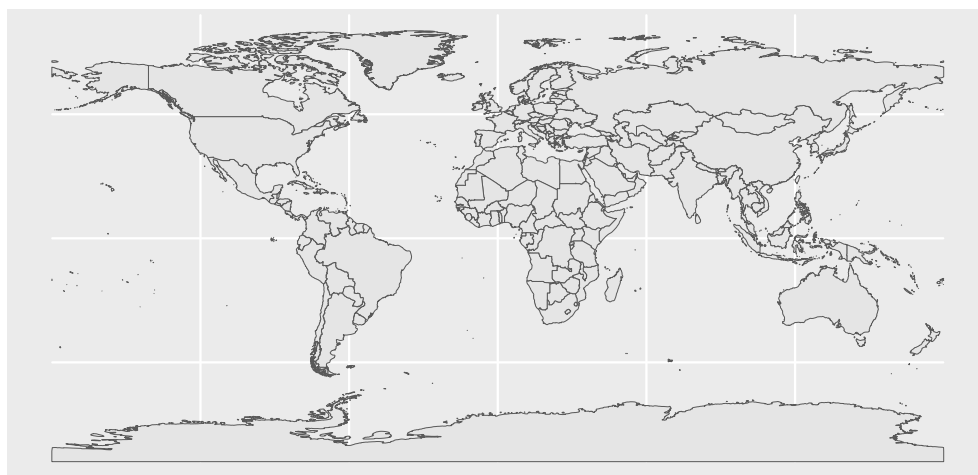
```
world5 %>%
  st_as_sf() %>% filter(GID_0 == "JPN") %>%
  ggplot() + geom_sf()
```



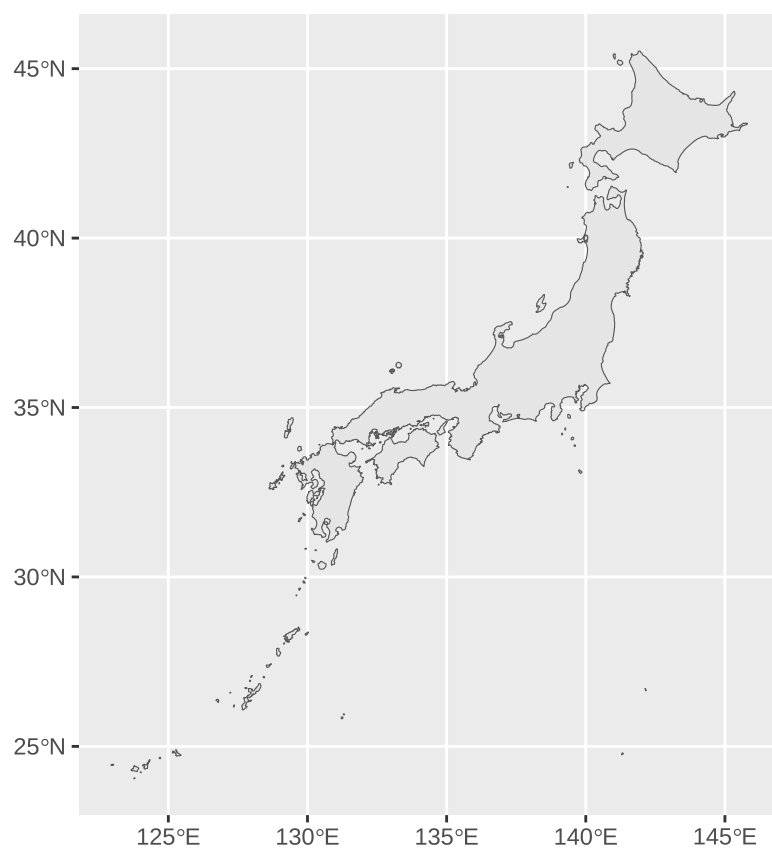
```
world(resolution=1, level=0, path = "./data") %>%  
  st_as_sf() %>% ggplot() + geom_sf()
```



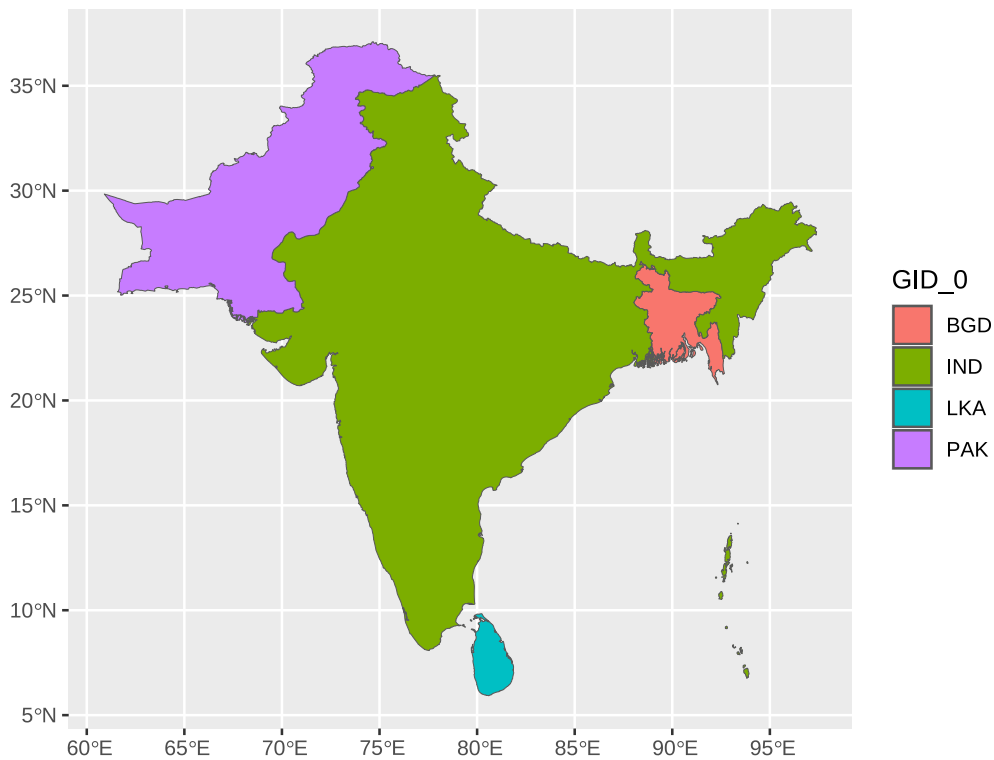
```
world(path = "./data") %>%  
  st_as_sf() %>% ggplot() + geom_sf()
```



```
world(resolution=1, level=0, path = "./data") %>%
  st_as_sf() %>% filter(NAME_0 == "Japan") %>%
  ggplot() + geom_sf()
```

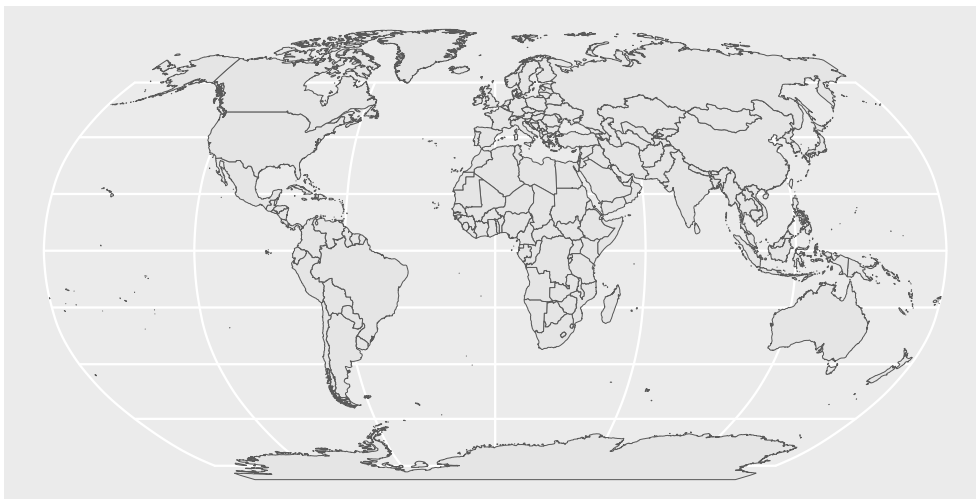


```
world(resolution=1, level=0, path = "./data") %>%
  st_as_sf() %>% filter(NAME_0 %in% c("India", "Pakistan", "Bangladesh", "Sri Lanka")) %>%
  ggplot() + geom_sf(aes(fill = GID_0))
```



```
world5_df <- st_as_sf(world5) %>%
  st_set_crs("+proj=longlat +datum=WGS84") %>%
  st_transform(., "+proj=robin")

ggplot() +
  geom_sf(data = world5_df)
```



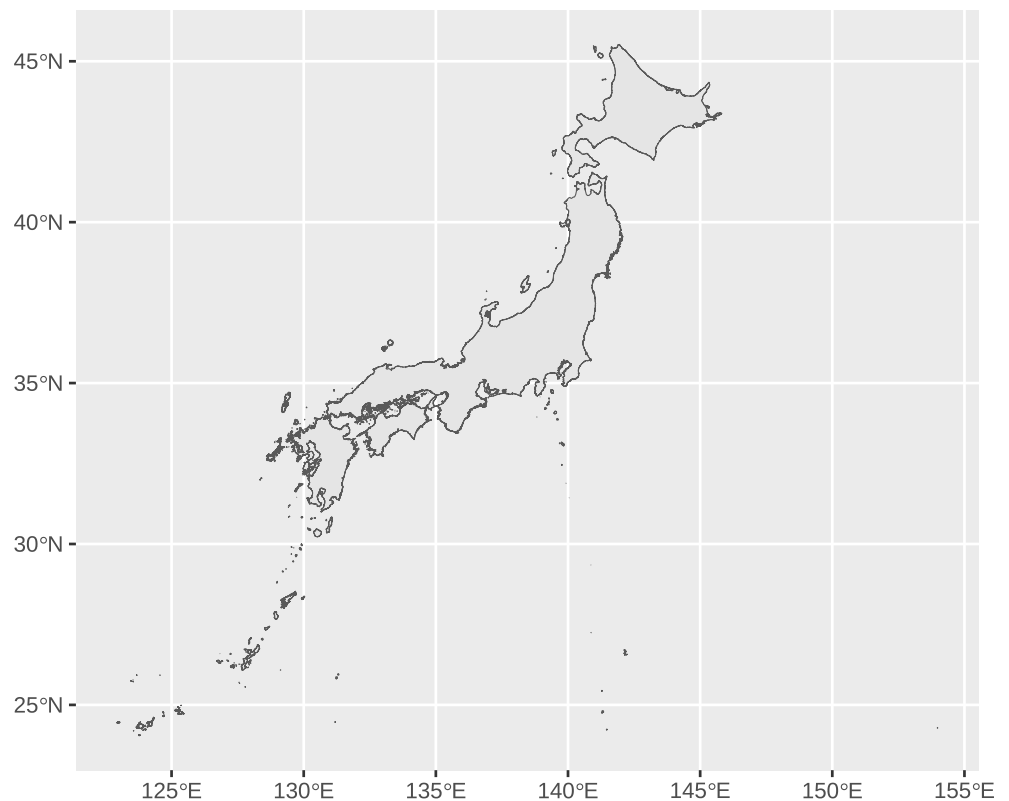
32.3.1 gadm Administrative boundaries

- Get administrative boundaries for any country in the world. Data are read from files that are downloaded if necessary.

- Usage: `gadm(country, level=1, path, version="latest", resolution=1, ...)`
- Arguments
 - `country`: character. Three-letter ISO code or full country name. If you provide multiple names they are all downloaded and `rbind`-ed together
 - `level`: numeric. The level of administrative subdivision requested. (starting with 0 for country, then 1 for the first level of subdivision)
 - `path`: character. Path for storing the downloaded data. See `geodata_path`
 - `version`: character. Either “latest” or GADM version number (can be “3.6”, “4.0” or “4.1”)
 - `resolution`: integer indicating the level of detail. Only for version 4.1. It should be either 1 (high) or 2 (low)

```
gadm0 <- gadm("JPN", level = 0, path = "./data/")
```

```
gadm0 %>% st_as_sf() %>%  
  ggplot() + geom_sf()
```

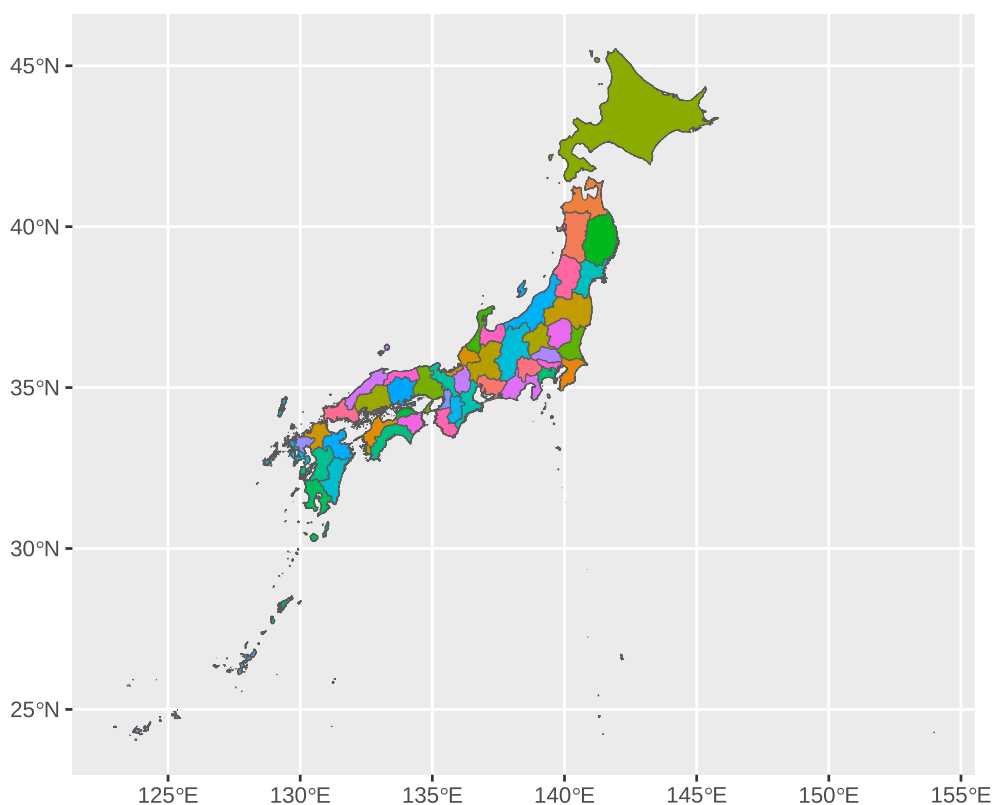


```
gadm1 <- gadm("JPN", level = 1, path = "./data/") %>% st_as_sf()  
gadm1 %>% glimpse()  
#> Rows: 47  
#> Columns: 12  
#> $ GID_1      <chr> "JPN.1_1", "JPN.2_1", "JPN.3_1", "JPN.4_~  
#> $ GID_0      <chr> "JPN", "JPN", "JPN", "JPN", "JPN", "JPN"~
```

```
#> $ COUNTRY <chr> "Japan", "Japan", "Japan", "Japan", "Jap~
#> $ NAME_1 <chr> "Aichi", "Akita", "Aomori", "Chiba", "Eh~
#> $ VARNAME_1 <chr> "Aiti", NA, NA, "Tiba/Tsiba", NA, "Hukui~
#> $ NL_NAME_1 <chr> "愛知県", "秋田県", "青森県", "千葉県", ~
#> $ TYPE_1 <chr> "Ken", "Ken", "Ken", "Ken", "Ken", "Ken"~
#> $ ENGTYP_1 <chr> "Prefecture", "Prefecture", "Prefecture"~
#> $ CC_1 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ HASC_1 <chr> "JP.AI", "JP.AK", "JP.AO", "JP.CH", "JP.~
#> $ ISO_1 <chr> "JP-23", "JP-05", "JP-02", "JP-12", "JP-~
#> $ geometry <GEOMETRY [°]> MULTIPOLYGON (((137.0974 34....,~
```

```
gadm1 %>%
```

```
  ggplot() + geom_sf(aes(fill = NAME_1)) + theme(legend.position = "none")
```



```
gadm2 <- gadm("JPN", level = 2, path = "./data/") %>% st_as_sf()
```

```
gadm2 %>% glimpse()
```

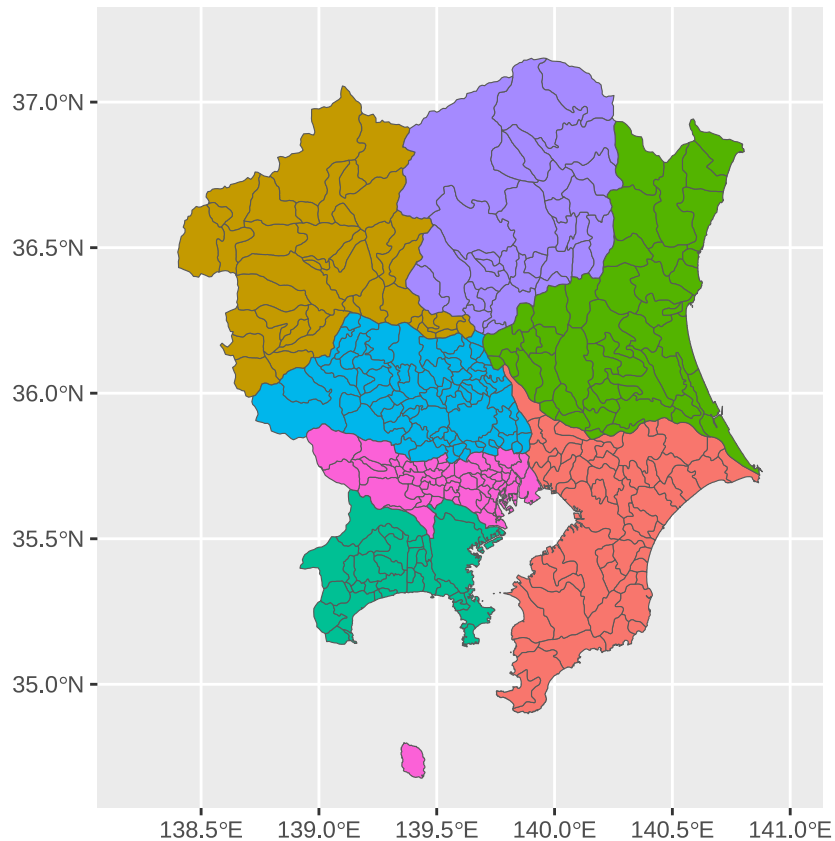
```
#> Rows: 1,811
```

```
#> Columns: 14
```

```
#> $ GID_2 <chr> "JPN.1.1_1", "JPN.1.2_1", "JPN.1.3_1", "~
#> $ GID_0 <chr> "JPN", "JPN", "JPN", "JPN", "JPN", "JPN"~
#> $ COUNTRY <chr> "Japan", "Japan", "Japan", "Japan", "Jap~
#> $ GID_1 <chr> "JPN.1_1", "JPN.1_1", "JPN.1_1", "JPN.1_~
#> $ NAME_1 <chr> "Aichi", "Aichi", "Aichi", "Aichi", "Aic~
```

```
#> $ NL_NAME_1 <chr> "愛知県", "愛知県", "愛知県", "愛知県", ~
#> $ NAME_2 <chr> "Agi", "Aisai", "Anjō", "Chiryū", "Chit~
#> $ VARNAME_2 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ NL_NAME_2 <chr> "阿久比町", "愛西市", "安城市", "知立市"~
#> $ TYPE_2 <chr> "Machi", "Shi", "Shi", "Shi", "Shi", "Ma~
#> $ ENGTPE_2 <chr> "Town", "City", "City", "City", "City", ~
#> $ CC_2 <chr> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, ~
#> $ HASC_2 <chr> NA, NA, NA, NA, "JP.AI.CG", NA, NA, "JP.~
#> $ geometry <GEOMETRY [°]> POLYGON ((136.8802 34.94398...,~
```

```
gadm2 %>% filter(NL_NAME_1 %in% c("埼玉県", "群馬県", "栃木県", "茨城県", "千葉県"))
ggplot() + geom_sf(aes(fill = NAME_1)) +
ylim(34.7,37.2) + xlim(138.2,141) +
theme(legend.position = "none")
```



32.4 参考文献

- Data Visualization for International Relations: Chapter 7 Choropleth Maps: https://bookdown.org/alhdzsz/data_viz_ir/maps.html
- R Graphics Cookbook: 13.18 Creating a Choropleth Map: <https://r-graphics>

.org/recipe-miscgraph-choropleth

- Geocomputation with R : First Edition in bookdown サイト
 - 日本語翻訳版 (<http://124.219.182.167/geo/>)

第 V 部

PART V EXAMPLES

第 33 章

Example 1

第 34 章

参考文献

参考文献とともに、おすすめのサイトなどについても、個人的に、学ばせていただいたものを中心に書いていきます。

34.1 教科書

すでに何度かご紹介していますが、Tidyverse の中心的開発者でもある、Hadley Wickham 等の書いた、次の本に勝るものは、ないと個人的には考えています。

R for Data Science (r4ds と略) : <https://r4ds.hadley.nz>

詳細はリンク先をみていただければと思いますが、これを書いている段階で、第二版 (2e) がちょうど出版、公開されました。わたしは、第一版で最初勉強し、この文書を準備し始めた頃には、第二版の暫定版が公開されていたから、そちらは、全てではありませんが、読ませていただき、参考にさせていただきました。多くの部分が、この本に則っています。

第一版も、bookdown パッケージで書かれています。この書が、bookdown で書かれ、公開されたことから、他のさまざまな、データサイエンスの教科書も、たとえば、bookdown の Archive のようなところに公開されて、活用されてきたことが、R のデータサイエンスへの利用が、飛躍的に進んだ原因であると同時に、RMarkdown や、bookdown の開発と相まって、このような、Public Domain での出版が進んだ原因だと、個人的には確信しており、個人ではなく、RStudio Co. (現在は、Posit Co.) という会社組織で進んで行ったことが、安定度も含めて、大きな意味があったと思っています。その意味でも、世界を変えた一冊だと考えています。

一つだけコメントさせていただければ、印刷体だけでなく、Public Domain での出版ということで、プロの翻訳家が手を出さず、日本語訳が出なかったことが、日本での知名度が上がらなかった原因であるとも思います。しかし、Web Browser (ホームページ閲覧ソフト) の自動翻訳技術も飛躍的に進歩し (今後ますます進歩するでしょうから)、Web 上に公開することで、このような技術書は、絶対的ではないという状況になったことも、大きな変化かなと考えています。そのような学び方の改革をもご紹介し、その橋渡しができればと願って始めたのがこの文書のプロジェクトでもあります。

34.1.1 練習問題

R for Data Science のもう一つの特徴は、同時に、練習問題集も、早いうちから公開されたことです。learnr というパッケージを使い、対話型 (interactive) な練習問題が、Primers (入門書) として、Posit Primers (最初は、RStudio Primers) として開発されたことです。採点など、試験などとして使うことを考えると、まだ、完全とは言えませんが、学びのためには、十分の質のものだと思います。

r4ds のあとに書かれているものは、教科書の対応する章ですが、これは、第一版のもので、第二版では、必ずしも対応していませんが、残しておくことにしました。ここにリストされたものがすべて完成したのは、2022 年の終わり頃かと思います。簡単に、日本語の表題を、なるべく本書の表題と近い名前にして添えておきます。

本書では、1 から 4 と、7 に重点を置いています。

34.1.1.1 Posit Primers <https://posit.cloud/learn/primers>

1. R と Tidyverse の基礎：The Basics – r4ds: Explore, I
 - 視覚化の基礎：Visualization Basics
 - プログラミング基礎：Programming Basics
2. データの変形：Work with Data – r4ds: Wrangle, I
 - 表の形式 Tibble：Working with Tibbles
 - dplyr によるデータの抽出：Isolating Data with dplyr
 - dplyr によるデータの変形：Deriving Information with dplyr
3. データの視覚化：Visualize Data – r4ds: Explore, II
 - 探索的データ分析：Exploratory Data Analysis
 - 棒グラフ：Bar Charts
 - 度数分布：Histograms
 - 箱ひげ図：Boxplots and Counts
 - 散布図：Scatterplots
 - 折れ線グラフと地図：Line plots and maps
 - 点データの重なり：Overplotting
 - さまざまな調整：Customize plots
4. Tidy Your Data – r4ds: Wrangle, II

- データ表の行と列の交換：Reshape Data - a bit old
 - 行または列の結合と分離：Separate and Unite
 - 二つのデータ表の結合：Join Data Sets
5. プログラミング：Iterate – r4ds: Program
- 繰り返し処理入門：Introduction to Iteration
 - 同時変換：Map
 - 表などの同時変換：Map Shortcut
 - ベクトルの変換応用：Multiple Vectors
 - 応用編：List Columns
6. 関数の活用：Write Functions – r4ds: Program
- 関数の基礎：Function Basics
 - 関数の書き方：How to Write a Function
 - 引数の対応：Argument Matching
 - 環境と適用範囲：Environments and Scoping
 - 条件分岐：Control Flow
 - 複雑な条件分岐：Advanced Control Flow
 - R による再帰型プログラム：Loops in R
7. 再現可能性を重視したレポート：Report Reproductively – r4ds: Communicate
- ビデオや説明へのリンク：Link to Videos and Explanations
8. 対話型アプリの作成：Build Interactive Web Apps

34.2 MOOCs などのオンラインコース

わたしは、MOOCs (massive open online courses) がちょうど始まった 2012 年ごろ、その 10 年前ぐらいに、スタートした、OCW (open courseware) を勉強していたこともあり、MOOCs のコースをかなり最初の段階からいくつか、視聴していました。

データサイエンスで人気があったのは coursera でホストしていた、Johns Hopkins University が提供する、10 コースからなる、JHU Data Science というコースと、edX でホストしていた、Harvard University が提供する、9 コースからなる Data Science。

下にリンクと、そのコースを下にした、オンラインなど容易に手に入ると思われる教科書へのリンクをつけておきます。

- edX: HarvardX Data Science - 9 courses. Textbook:
 - "Introduction to Data Science" by Rafael A. Irizarry.
 - Free Online Book by Rafael A. Irizarry.
- coursera: JHU Data Science - 10 courses. List of Companion Books:
 - "R Programming for Data Science" by Roger Peng.
 - Free Online Book by Roger Peng.
 - "Exploratory Data Analysis with R" by Roger Peng.
 - Free online Book by Roger Peng.
 - "Report Writing for Data Science in R" by Roger Peng
 - "Statistical Inference for Data Science" by Brian Caffo
 - "Regression Modeling for Data Science in R" by Brian Caffo

どちらも、素晴らしいコースだと思いますが、個人的には、edX Harvard X のものは、アカデミック（学問的・学際的）で、coursera の方は、技術的（コンピュータ科学）のような印象を受けました。

いまでは、Data Science と検索すれば、膨大なコースが見つかりますが、個人的には、Data Science というものに、実際に触れた最初がこのにコースだったので、影響されていると思います。

また、2018 年ごろからでしょうか、会社組織で、データサイエンス教育のようなものを提供するところが、続々と出てきました。DataCamp や、DataQuest などなど、たくさんあります。

MOOCs の方は、受講証明書を必要とせず、各単元の最後のテストの評価が必要なければ、基本的に無償で受けられますが、DataCamp などは、大体、最初の一コースだけ無償で、そのあとは有償になっているかと思います。

有償のコースで、アドバイスをもらったりなど、指導してもらえたり、わからないところの質問をなんでも聞けるというのは、とてもよいと思いますが、オンライン上のコミュニティでも、かなり無償でサポートが得られますし、最近では、AI による、サポートもどんどん進化していますから、個人的には、AI が家庭教師についてくれるような世界へと進んでいくのではないかと、そして、基本的な部分は、それで十分ではないかと思っています。

34.3 その他の参考書

個人的に、読んだもの、参考にしたものもいくつかありますが、あとは、必要に応じて、参照していただくのがよいと思います。日本語のものは、少ないですが、英語のものであっても、自動翻訳なども上手に使える、問題なく読めると思います。

上でも書いた、Bookdown サイトと、そこにリンクされている、Archives のリンクをつけておきます。膨大な、電子書籍がリンクされていますが、それは、そこに投稿するシステムも完備しているからでもあります。みなさんも、電子書籍を書いてみませんか。

BOOKDOWN: <https://bookdown.org>

Archive Page: <https://bookdown.org/home/archive/>

日本語で書かれたものも、数は多くありませんが、いくつかあります。

34.4 統計関連

統計学についてほとんど書きませんでした。必要ではないという意味では全くありません。しかし、データサイエンスというと、まず、数学、そして、統計学を勉強して、コンピュータプログラミングも勉強してはじめて、始められるのですよね。というような声には、抗（あらが）って、書いてきました。最初から、これらで苦しまなくても、必要になった時に、少しずつ勉強していけばよいと考えているからです。

ひとつ一つステップを踏んで学んでいくよりも、まずは、データに慣れること、グラフをみて、問いを持つこと、そして、さらに、他のグラフを見ること。そう考えると、問いを持ちながら、視覚化を通して、データから特徴を読み取り、さらに問いを深めて、次の視点から、またデータを見ていくことが、何よりも大切だと考えているからです。

そのあとで、統計を勉強してみよう。プログラミングや、コンピュータサイエンスを勉強してみよう、数理モデルについて、もう少し深く知りたい、もう少し、基本的な数学も学び直してみたいとなれば、個人的にはとても嬉しいです。

34.4.1 CRAN

最後の方に、R 本体をホストし、保持して下さっている、CRAN (The Comprehensive R Archive Network) をあげるのは、非常に失礼だと思いますが、CRAN にも、基本的な文書はあります。

マニュアル (The R Manuals) : <https://cran.r-project.org>

(左の帯の Documentation の中の Manuals)

さまざまなものがリストされています。この下の方に、Contributed という項があり、そこから、Contributed Documentation という項目にリンクがあります。

(左の帯の Documentation の中の Contributed)

ここに、日本語のものも掲載されています。わたしは、全部読んだわけではありませんが、間瀬茂さんのものは、統計解析の道具として R を使う場合には、非常に有効だと思います。

- R 入門：An Introduction to R (PDF)

- R 言語定義：The R language definition (PDF)
- R の拡張を書く：Writing R Extensions (PDF)
- R のデータ取り込み／出力：R Data Import/Export (PDF)
- R のインストールと管理：R Installation and Administration (PDF)
- R 基本統計関数マニュアル：R Statistical Function Help Pages (PDF, 404 pages, 2009-06-05)

34.4.2 その他のオンライン文書

- BellCurve 統計 WEB：<https://bellcurve.jp/statistics/>
 - 統計用語集：<https://bellcurve.jp/statistics/glossary/initial/a/>

付録 A

日本語の扱いについて

A.1 日本語・中国語・韓国語

文字化けが、起こることが多く、対応が、一定せず、特に、図の表示において、Windows や、macOS や、Linux などの、OS ごとに、フォントが違ったり、それを、図のタイトルなどに、使ったりが、難しかったのですが、どうやら、現在は、どの場合も、次の設定で、解決しているようです。下の例を確認してください。特に、フォントについては、好みも関係しますから、難しいですが、ここでは、どのプラットフォーム（OS）でも、共通に扱えることを中心に書きます。

```
# showtext を、インストールしていない場合は、一回だけ、右上の三角をクリックして実行  
install.packages('showtext')
```

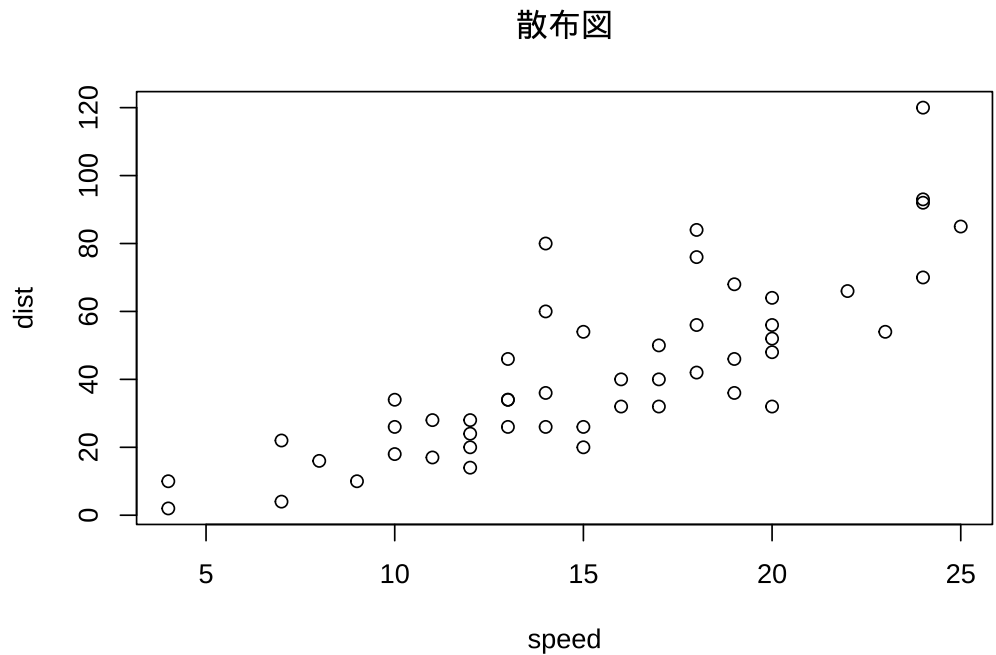
A.1.1 パッケージをロード

library によって、Package をロード（いつでも使えるように）します。

```
library(tidyverse)  
library(showtext)  
showtext_auto()
```

A.2 Base R でタイトルに日本語

```
plot(cars, main=" 散布図")
```



A.3 列名や、データに日本語

```
df_iris <- iris
colnames(df_iris) <- c(" 萼長", " 萼幅", " 葉長", " 葉幅", "Species" )
tab <- data.frame(Species = c("setosa", "versicolor", "virginica"),
                  " 種別" = c(" ヒオウギアヤメ", " ブルーフラッグ", " バージニカ"))
df_iris <- df_iris %>% left_join(tab, by=c("Species" = "Species")) %>% select(-5)
df_iris %>% slice(1:2)
#>   萼長 萼幅 葉長 葉幅      種別
#> 1  5.1  3.5  1.4  0.2 ヒオウギアヤメ
#> 2  4.9  3.0  1.4  0.2 ヒオウギアヤメ
```

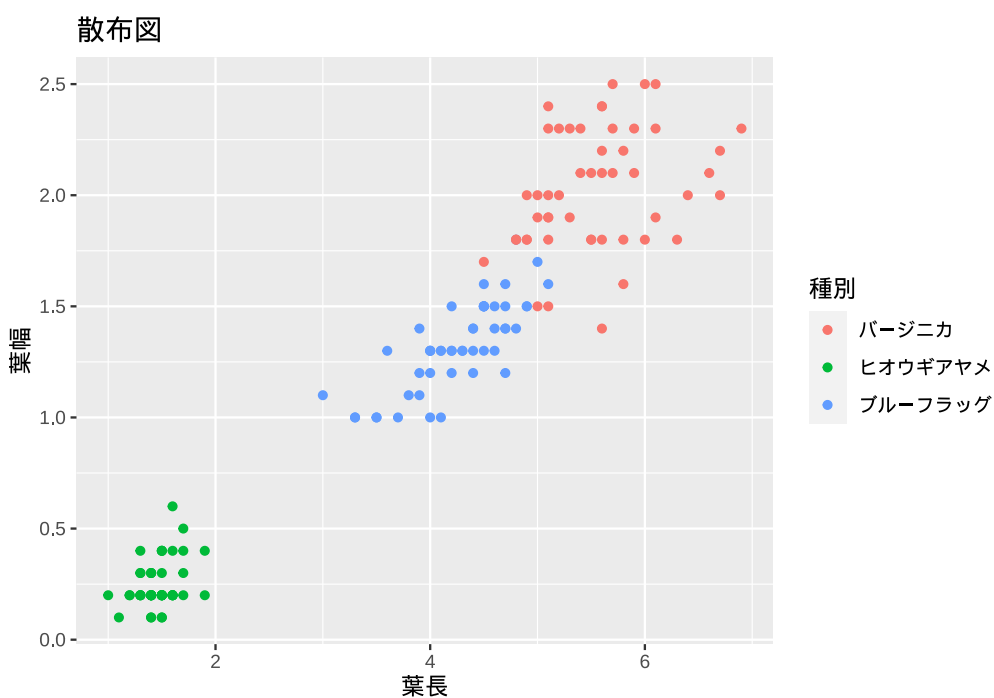
A.4 kable で表示

```
knitr::kable(df_iris[1:6, ])
```

萼長	萼幅	葉長	葉幅	種別
5.1	3.5	1.4	0.2	ヒオウギアヤメ
4.9	3.0	1.4	0.2	ヒオウギアヤメ
4.7	3.2	1.3	0.2	ヒオウギアヤメ
4.6	3.1	1.5	0.2	ヒオウギアヤメ
5.0	3.6	1.4	0.2	ヒオウギアヤメ
5.4	3.9	1.7	0.4	ヒオウギアヤメ

A.5 ggplot でグラフを作成

```
ggplot(df_iris, aes(x = `葉長`, y = `葉幅`, col = `種別`)) +  
  geom_point() + labs(title = " 散布図", x = " 葉長", y = " 葉幅")
```



A.6 備考：

実は、一番難しいのが、PDF の作成だと思いますが、一応、上のものも、PDF を作成することが可能です。下のリンクのファイルを、いろいろな、形式で、出力してみてください。R Notebook と、PDF に出力したもののリンクを付けておきます。

- R Notebook
 - 右上の Code ボタンから、Rmd ファイルも取得できます。
- R Notebook RMD
 - Rmd の中身がテキストで表示されますから、コピーして、新規作成した、RNotebook ファイルに貼り付けることも可能です。
- PDF
 - 通常の PDF も、やはり PDF 形式の、beamer presentation も作成できます。

詳細は、これらのファイルに記載されていますから、参考にしてください。

A.7 参考：日本語の表示について

日本語が適切に表示されない!?

簡単ではなく、未解決の部分が何かなどを含め、わたしも十分理解できているか不明であるが、理解できていると思われる範囲で、備忘録のように記します。

R を使うという場合に限っても、R Studio IDE を使う場合、RStudio Cloud を使う場合、Google colab を使う場合、他のプラットフォームで使う場合で違ってくると思われるが、上にあげた、三種類のプラットフォームで確かめられるものについて書いていく。上に書いた以外に、R Studio IDE を、Windows 上で使う場合と、Mac 上で使う場合 (Mac のシステムは Unix 系であるが、さまざまな Linux) でも、状況が異なるかもしれません。そこで、場合分けをして書いていくほうが安全ですが、それは、極力避け、どれにでも適用可能な方法を模索しながら書いていこうと思います。個人的に、日常的に分断を避ける努力をすることが大切だと思っていることも背景にある。さらに、ソフトウェア開発者は、むろん、そのような差異を理解して、どの環境でも、可能なように設計することを心がけていると思われますし、そのようなものが、R Project の正規のパッケージとして採用されていくべきだとも考えていますので、多少、理想も入っているが、これを基本として書いていこうと思います。十分なチェックができていないものもあるので、不具合などは、ぜひ、お知らせ願いたい。この文章も少しずつ、改善していければと思う。

通常、日本語、中国語、韓国語などが適切に表示できない場合は、文字のエンコーディング (Encoding: どのような情報として記録されているか) と、フォントの問題、さらに、システムがこれらをどう処理しているかの問題があると思われます。しかし、R の利用者として考えると、文字化けが起きたり、適切に文字が表示されないのは、以下の三つに分けられるようです。

1. データファイルなどを読み込んだときに適切に表示されない
2. 図の中のタイトルなどが、適切に表示されない
3. R Markdown の出力において、適切に表示されない

A.7.1 データファイルの読み込み

- tidyverse に含まれる readr には、guess_encoding が含まれており、一般的には、たとえば、

```
read_csv("./data/file_name.csv")
```

 とすると、一番可能性の高いエンコーディングで読み込まれるようになっています。
- 使い方: guess_encoding(file, n_max = 10000, threshold = 0.2) とあり、10000 行で推測されたエンコーディング、または、確率を計算することを初期設定値 (Default) にしています。Help によると、すべての行をチェックする場合は、n_max = -1 とすることが書かれています。
- これで問題がない場合が多いです。他の、readr 関数も同様です。しかし、これは、

あくまでも、CSV などのテキストファイルについてです。

- なお、`read_csv` などにも、`guess_max = min(1000, n_max)` もありますが、これは、column type を決めるためのものである。
- `read.csv()` など、base R では、`fileEncoding = “”, encoding = “unknown”` がオプションに含まれていたため、指定して読み込むことが通常でした。

A.7.2 図の中のテキスト

- 基本的には、日本語を表示できるフォントがインストールされていて、図の表示の前に `library(showtext); showtext_auto()` となっていれば、これ以降の図は、問題なく、表示されるはずです。
- 通常使っている、コンピュータ以外で、使うときに、フォントが入っていない場合などは、`tinytex` の命令を使って、`tinytex::tlmgr_install("ipaex")` とすれば、PDF を作成するときも、IPA フォント (International Phonetic Alphabet) を使えます。
- 二種類以上のフォントを使い分けたいときは、名前をつけて、それを `family = name` で指定する。
 - `showtext: Using Fonts More Easily in R Graphs` 参照。

A.7.3 R Markdown の出力

- PDF 作成には、 $\text{T}_{\text{E}}\text{X}$ システムを使っているため、日本語を扱えるように、`tex-engine` や、`document class` や、`mainfont` を設定する必要があるが、R Markdown ファイルの YAML に、以下を加えれば問題がないようである。

```
header-includes:
```

- ```
- \usepackage{xCJK}
- \setCJKmainfont{ipaexm.ttf}
- \setCJKsansfont{ipaexg.ttf}
- \setCJKmonofont{ipaexg.ttf}
```

### A.7.4 bookdown

`bookdown` を使って、電子書籍を出版する場合には、`bookdown` (リンク) を参照してください。

日本語におけるテンプレートは、こちらにあります。まずは、ページの下にある、README を読んでください。

### A.7.5 参考としたもの

#### A.7.5.1 showtext: Using Fonts More Easily in R Graphs

- <https://CRAN.R-project.org/package=showtext>
  - <https://cran.r-project.org/web/packages/showtext/readme/README.html>
  - showtext: Using Fonts More Easily in R Graphs:
    - \* <https://cran.r-project.org/web/packages/showtext/vignettes/introduction.html>
    - \* <https://fonts.google.com>

#### A.7.5.2 sysfonts: Loading Fonts into R

- <https://CRAN.R-project.org/package=sysfonts>
  - <https://cran.r-project.org/web/packages/sysfonts/sysfonts.pdf>

#### A.7.5.3 foods4all: Examples of Graphs

- [https://foods4all.github.io/examples/examples\\_of\\_graphs.html](https://foods4all.github.io/examples/examples_of_graphs.html)
  - 77.2 Japanese Environments 日本語環境 (昔の記事 : Last Updated: 2020-04-22)

## 付録 B

# IT ツール

いくつかの便利なツールについて紹介します。

### B.1 Git と GitHub

Git はバージョン管理システムで、GitHub はそれを、活用し、かつ他のメンバーと協力して開発など、さまざまな活動をするためのサイトです。公開が基本となっています。非公開にすることも可能ですが、公開することで、世界中のひとたちと協力していくことが可能になりますので、その利点も学んでいただければと思います。

#### B.1.1 概要

RStudio で R を使っている場合、Git-GitHub-RStudio の連携で使うことをお勧めします。しかし、これらは、三つとも、まったく異なるものですから、簡単な概要を書いておくことにします。

##### B.1.1.1 Git

これは、ファイルのバージョン（更新履歴）の管理システムで、単独で機能します。他の、プログラムなどに関係しない、他の文書ファイルであっても、バージョンを管理する場合に活用できます。特に、テキスト・ファイルの場合には、どこがどう改訂されているかを確認することもできます。また、基本的には、Unix の Shell プログラムで動作させるのが一般的です。Mac は、Unix システムの上に構築されているため、最初から、ユーティリティ（Utility）>ターミナル（Terminal）で、Shell コマンドが利用可能になっていますが、Windows の場合には、bash と呼ばれる Shell プログラムをインストールすることをお勧めします。Windows システムについてよくご存知の方は、他の方法を使っていただいて構いませんが、Git のインストールの時に、Git bash を選択して、簡単にインストールできますし、Unix システムの基本を理解するチャンスでもあり、Mac と同じ環境で説明できますから、ここでは、そちらを使います。Shell コマンドは、R Studio の中のターミナルを使って、利用することも可能です。（注：Windows のコマンド・プロンプト、

または、パワー・シェルをお使いの方は、利用環境が変化する可能性がありますから、そのまま使われる方が良いかもしれません。）

基本的なコマンドとしては、以下のものがあります。いまは、このようなものがある程度に、眺めておいてください。

- `git init`: 特定のディレクトリ（フォルダ）でバージョン管理を始める時に使います。
- `git status`: 現在の状況を確認するときに使います。
- `git diff file_name`: ファイルへの変更を確認します。
- `git log`: 過去の `commit` による履歴を確認する時に使います。
- `git add file_name`: ステージングという中間的な場所に登録します。
- `git commit -m "log message here"`: ステージングにあるものを、確定させます。引用符で囲まれた短いコメントを加えます。50 文字が上限です。
- `git help`: Help のリストが表示されます。
  - 例：`git help init` などと入力すると、説明を見ることができます。

#### B.1.1.2 Git Hub

Git でバージョン管理されているディレクトリ（フォルダ）の状態を示す、クラウドサービスです。更新されている、状態を確認するとともに、変更履歴なども確認できます。また、Git Hub サービスを利用して、ファイルを公開、共有することも可能です。Pages サイトを利用することで、ホームページとして HTML ファイルなどを公開することもできるため、レポートを公開したり、この電子書籍のように、`bookdown` パッケージを利用して作成したものを、インターネット上に公開することも可能です。

お気付きかと思いますが、この電子書籍も、リンクされている、他の文書も、URL（アドレス）をみると、GitHub になっていますし、パッケージや、データなども、GitHub へのリンクが示されている割合が、非常に高いと思います。

最初に、Git で管理されている、ディレクトリ（フォルダ）（これを、ローカル・リポジトリと言います）と、GitHub 内のリポジトリ（リモート・リポジトリまたはアップストリーム・リポジトリと言います。ここでは、リモート・リポジトリと呼ぶことにします）を結びつけるステップが必要です。

#### B.1.1.3 RStudio 連携

コマンドライン（シェル）で行う作業や、ローカル・リポジトリを、リモート・リポジトリに結びつける作業を、RStudio の中で行うことが可能です。

### B.1.2 はじめかた

#### 1. Git のインストール

- Windows と Mac で異なりますので注意が必要です。Mac については、**Mac** と書

いてあるところを読んでください。

- **Windows** の場合は、git-scm にアクセスしてダウンロード、インストールしてください。セットアップ (Setup) で、2箇所、注意点があります、それ以外は、すべて、初期設定のままで変更は必要ありません。
  - Choosing the default editor used by Git: 設定で、エディタ (Editor) を設定しますが、vi, vim に慣れていない方は、nano を選択することをお勧めします。(nano<sup>\*1</sup> は、メニューが下に出るので、それを見て操作することが可能なエディターです。)
  - Adjusting Your Path Environment: Windows のコマンドライン・ツール (command line prompt) を使っていない方は、Git Bash のインストールを選択してください。さらに、Git and optional Unix tools from the Windows Command Prompt を選択することをお勧めしますが、上で書いたように、Windows のコマンド・プロンプトになれておられる方で、それを使い続けたいかたは、Use Git from Git Bash only を選択されるのが良いかもしれません。
  - 最後に、RStudio の設定 (Tools > Global Option) で、Terminal から、Git Bash を選択<sup>\*2</sup>し、Tools から、New Terminal を選択<sup>\*3</sup>します。
- **Mac** は、最初から、Install されていると思います。ユーティリティ (Utility) > ターミナル (Terminal) を開いて<sup>\*4</sup>、`git --version` とすると、インストールされているバージョンが表示されると思います。バージョンがでない場合には、Install するかと聞かれます。このときに、Git だけをインストールすることも、Xcode という開発環境を同時にインストールすることも可能です。(インストールするように指示が出なければ、App Store から、インストールできます。もし、そのあとで、git などのコマンドで `xcrun: error` などとエラーが出たら、`xcode-select --install` としてください。) インストールが終了したら、もう一度、`git --version` と Terminal に入力して、結果を確認してください。

## 2. GitHub のアカウント取得

- GitHub サイト に、アカウントを作成します。アカウント名は、短く、分かりやすく、覚えやすいものをよく考えて決めてください。Email Address だけで、無償で作成できます。

## 3. RStudio の左下の窓枠の Terminal タブ<sup>\*5</sup>から、GitHub アカウントに連携する設定を行います。下の2行を、1行ずつ、コピーして、Terminal に入力してください。

```
git config --global user.name "Your Name" # GitHub の User Name
git config --global user.email "your@email.com" # GitHub に登録したメールアドレス
```

<sup>\*1</sup> GNU nano は、コンソールウィンドウで動作する小型でフレンドリーなテキストエディタです。(GNU nano is a small and friendly text editor running in the console window.)

<sup>\*2</sup> 自動的に選択されているかもしれません

<sup>\*3</sup> すでにタブがあればこの作業は不要です

<sup>\*4</sup> RStudio を既にお使いの方は、左下の窓枠から、Terminal タブを選択できますので、それを使うことも可能です。

<sup>\*5</sup> Terminal がない場合は、Tools > Terminal > New Terminal とすると表示されます。

4. RStudio の、Tools > Global Option の、Git/SVN タブを開き、Git Executable とあるところに、Git 実行プログラムのある場所を入れます。
  - **Windows** の場合は、C:/Program Files/Git/bin/git.exe だと思いますが、Browse ボタンから確認してください。
  - **Mac** の場合は、/usr/bin/git になるかと思いますが、Browse ボタンから確認してください。
5. その下に、Create RSA key とありますから、それを押し、Create ボタンを押してください\*6。すると、View RSA key から、暗号キーも確認できます。（この作業は、Terminal から、ssh-keygen -t rsa として作成することも可能です。この作業で、~/.ssh/ 内に、SSH キーが記述されたファイルが作成されます。）
6. GitHub アカウントで公開鍵を利用できるようにします。まず、RStudio の方で作成した、RSA key (Tools の、Global Option の Git/SVN) の下にある view を押すと見ることができ、上に、書いてあるように、そこに出てきたものを、コピーします。次に、GitHub にログインし、右上のアイコンの右の三角から、設定 (Setting) を選択し、SSH 公開鍵 (SSH and GPG Keys) を選択します。新しい公開鍵を追加 (New SSH Key) を選択すると、SSH キーを貼り付けることができます。（リポジトリの左上にある、アカウント名をクリックし現れるダッシュボードの左上の大きなアイコンをクリックしても「アカウント設定」が現れ、SSH and GPG Keys を見つけることができると思います。）

(なお、RStudio ではなく、Terminal からコピーするときは、Unix では、pbcopy < ~/.ssh/id\_rsa.pub などとします。Windows の場合は、pbcopy が使えない可能性があるので、そのときは、Terminal から、Git Bash を使い、use < ~/.ssh/id\_rsa.pub とします。Terminal に慣れておられない方には、上に紹介した、RStudio からコピーする方が簡便かと思います。)

7. これで設定終了ですので、R Studio を再起動させてください。

### B.1.3 GitHub にあるリモート・リポジトリ (Remote Repo) から始める場合

1. GitHub にログインして、既存のリポジトリを開きます。
2. Code の、Clone から、リンク先のアドレスを入手。https と SSH を選べますが、SSH を選び、コピーします。
3. RStudio から、New Project とし、Version Control を選択し、ディレクトリーを決めたら、上でコピーした、ものを、貼り付けて、Project を作成します。

この手続きで、リモート・リポジトリのファイルがすべて、RStudio のプロジェクトに入ります。

---

\*6 Passphrase (optional) と出ますが、無視して下さって構いません。

実はこの手続きで、公開されている、他のリポジトリも取り込むことができます。ただし、編集して、改訂していくには、自分のリポジトリに、繋ぐことになります。そのときは、次の項目を見てください。

### B.1.4 自分のコンピュータのリポジトリ (Local Repo) から始める場合

1. RStudio から新しい、プロジェクト (Project) を作成 `test0` としておきましょう。
2. GitHub に、新しい、レポジトリを作成して繋げる
  - 自分の GitHub アカウントに、新しい、レポジトリをプロジェクトと同じ名前 `test0` で作成します。同じ名前でもかまわないのですが、関連がしやすいので、同じ名前がお勧めです。
  - Quick Setup というページが表示されますから、その、下の Set up in Desktop or の右から、https と SSH を選べますが、SSH を選び、コピーします。
  - プロジェクトの中の左下の窓枠の、Terminal から次を実行します。

```
echo "# Project test0" >> README.md # README の表題を書きます。
```

```
git init
```

```
git add README.md
```

```
git commit -m "First commit. README.md"
```

```
git branch -M main
```

```
git remote add origin `git@github.com:icuh-suzuki/test0.git`
```

```
わたしのアカウントの場合には、コピーしたものを貼り付けると、上のようになります。
```

```
git push -u origin main
```

再起動すると、プロジェクトの右上の窓枠に、Git と表示されていると思います。開いて、いくつかファイル名があれば、これを選択し、Commit ボタンを押し、短い message を書いて、Commit とし、そのあとで、Push ボタンを押してください。HEAD -> main と表示されれば、OK です。

### B.1.5 GitHub Pages について

GitHub は、Web 上のサービスですが、たとえ、リポジトリの中に、HTML ファイルなどが存在しても、ブラウザで表示させることはできません。公開する方法として、わたしは、以下のようにしています。

公開しブラウザで読めるようにしたいファイルを、docs というディレクトリに入れます。Setting を選択し、左の帯から、Pages を選び、Branch を、main、/docs として、Save します。すると、docs に入っているファイルは、リンクを指定すれば、ブラウザで、見ることができます。

リンクは、まず、リポジトリの右上の、About の右のギアマークから、Use your GitHub Pages website に、チェックを入れると、URL が表示されますから、その URL に続けて、ファイル名を加えたものが、そのファイルの、リンクとなります。

## B.1.6 Bookdown パッケージによる、電子書籍の執筆

bookdown を利用して、始める方法を簡単に記します。

Git-GitHub-RStudio の設定は済んでいると仮定して書きます。また、Bookdown を利用するには、bookdown パッケージのインストールが必要です。RStudio の Tools から、Install Packages を選択して、インストールしてください。また、bs4\_book スタイルを利用する場合は、RStudio などのバージョンによっては、bslib と downlit も、インストールする必要があるようです。

以下では、R Studio で新しい、プロジェクトとして始める場合と、テンプレート・リポジトリを使う方法を説明します。

### B.1.6.1 RStudio での設定

1. 新しい book プロジェクトを始めます、標準設定の book と、bs4\_book を選択できます。このディレクトリ（フォルダ。ローカル・リポジトリと呼びます）の名前をリモート・リポジトリの名前と同じにして、作成します\*7。
2. Files の、Rename 機能を使って \_book ディレクトリ（フォルダ）の名前を、docs に変更します。
3. \_bookdown.yml を編集し次の行を加えます（私は 2 行目に入れています）。  
output\_dir: docs  
GitHub Pages の機能を使って、公開するための変更です。
4. @GitHub: 新しいリポジトリ（リモート・リポジトリと言います）を作成します。ローカル・リポジトリと同じ名前にし、簡単な説明を加えます。
5. 左下の窓枠から、Terminal タブを選択します。以下は例です。git remote add origin のところは、適当に変更してください。

```
echo "# ds-book" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:icuh-suzuki/ds4aj.git # 変更
git push -u origin main
```

5. RStudio を一旦終了し、もう一度、そのプロジェクトを立ち上げると、右上の窓枠に、Git タブが表示されるはずです。
6. Build から、Build Book ボタンを利用すれば、作成されます。
7. Git タブから: Commit “first build” とし、Push をします。
8. @GitHub: Settings から、Pages > main > docs と変更すると、公開されます。

---

\*7 異なる名前でも可能ですが、特別な理由がない限り同じにしておいた方が良いでしょう



9. Code に戻り、About の右のギアマークから、

#### ■B.1.6.1.1 他の PC での作業

1. Login to GitHub account
2. Copy SSH address under Code>Clone
3. Create a new project using Version Control:Git with the SSH address by setting the directory name
4. Edit README.md and test Git Commit and Push

#### B.1.6.2 テンプレートを利用する方法

■B.1.6.2.1 bookdown-demo bookdown の電子書籍 (bookdown: Authoring Books and Technical Documents with R Markdown) の、はじめてみよう (Get started) から、GitHub リポジトリ `rstudio/bookdown-demo` にリンクがついています。

1. 自分の、GitHub アカウントに、ログインします。
2. `rstudio/bookdown-demo` にアクセスし、Use this template から、create a new repository を選択します。
3. 自分のアカウントの中に、コピーされたリポジトリが作られます。
4. 上で説明した「GitHub にあるリモート・リポジトリ (Remote Repo) から始める場合」を利用してください。

よくできているテンプレートです。しかし、このままでは、公開されません。bookdown: Authoring Books and Technical Documents with R Markdown には、いくつかの公開方法が書かれていますから、参照してください。

■B.1.6.2.2 `jtr13/bookdown-template` YouTube ビデオ 5 分間で、bookdown の本を作成する方法 (How to create a bookdown book in 5 minutes) で紹介されているものです。

1. 自分の、GitHub アカウントに、ログインします。
2. `jtr13/bookdown-template` にアクセスし、Use this template から、create a new repository を選択します。
3. 自分のアカウントの中に、コピーされたリポジトリが作られます。
4. 上で説明した「GitHub にあるリモート・リポジトリ (Remote Repo) から始める場合」を利用してください。

ビデオの中では、簡単に、GitHub Pages (GitHub のホームページサービス) に、公開する方法が説明されています。最も、大切な部分は、テンプレートがコピーされた、自分

のリポジトリの Settings から、Pages を選び、Branch を main > docs とする部分です。このようにすることで、公開されます。上の、「GitHub Pages」で説明していることと同じです。

■B.1.6.2.3 日本語テンプレート bs4\_book スタイルでの、簡単な、日本語テンプレートを作成しました。

1. 自分の、GitHub アカウントに、ログインします。
2. icu-hsuzuki/bs4\_book\_template にアクセスし、Use this template から、create a new repository を選択します。
3. 自分のアカウントの中に、コピーされたリポジトリがつくれます。
4. 上で説明した「GitHub にあるリモート・リポジトリ (Remote Repo) から始める場合」を利用してください。

まだ、不十分ですが、少しずつ、改訂していきます。使い方は、リポジトリの README に、注意点などは、最後の章 Bookdown に書いておく予定です。

## B.1.7 大きなファイルに関すること

100M 未満のファイルだけを利用するのがよいですが、それより大きいものを、GitHub に挙げようとして、問題が起こることがあります。その対処法を書いておきます。

個人的には、大きなデータや、本などを、Build して、100M 以上の、非常に大きな、TeX 関連のファイルができることができました。

備忘録も含めて、書いておきます。

### B.1.7.1 大きなファイルの取り除きかた

下のサイトからの引用です。非常に分かりやすく書かれています。Terminal で作業を行います。いま、Add した場合と、いくつか、前のステップで、Add した場合で、対応の方法が異なります。

- Tutorial: Removing Large Files from Git
  - Scenario 1: The Large File Was Just Added in the Most Recent Commit
    - \* `git rm --cached big_file_name`
    - \* `git commit --amend -C HEAD`
  - Scenario 2: The Large File Was Committed Prior to The Most Recent Commit
    - \* Locating the Last “Good” Commit: `git log --oneline`
    - \* Initiate a Rebase Between the Last “Good” Commit and the Current Commit: `git rebase -i 8464da4`

- \* This will open up a file in your Git editor (in my case, Vim), that looks something like this:
  - pick -> edit
- \* `git rm --cached csv_building_damage_assessment.csv`
- \* `git commit --amend -C HEAD`

### B.1.8 複数のコンピュータから利用する方法

わたしも、いくつかのコンピュータから、同じ GitHub アカウントにアクセスして作業しています。

同じ SSH キーを、複数のコンピュータで利用することも可能です。特に、コンピュータの更新時に、移行する場合は、元の環境をそのまま使うことも可能で便利です。

ただし、基本的には、それぞれのコンピュータで別々の SSH キーを使用するのがお勧めです。問題が発生した時に、個別のコンピュータの課題として解決することができます。

(たとえば、RSA 形式で作成した) 複数の SSH キーを使用するときは、GitHub アカウントに公開鍵を追加する必要があります。

GitHub アカウントに別の公開鍵を追加するには、GitHub にログインし、右上のアイコンの右の三角から、設定 (Setting) を選択し、SSH 公開鍵 (SSH and GPG Keys) を選択します。新しい公開鍵を追加 (New SSH Key) を選択すると、SSH キーを貼り付けることができます。(リポジトリの左上にある、アカウント名をクリックし現れるダッシュボードの左上の大きなアイコンをクリックしても「アカウント設定」が現れ、SSH and GPG Keys を見つけることができますと思います。)

コピーを貼り付ける時には、RStudio の、Global Option の、Git/SVN タブから、View public key を見ると、コピーできるようになっています。

Terminal からコピーするときは、Unix では、`pbcopy < ~/.ssh/id_rsa.pub` などとします。Windows の場合は、`pbcopy` が使えない可能性があるので、そのときは、Terminal から、Git Bash を使い、`use < ~/.ssh/id_rsa.pub` とします。Terminal に慣れておられない方には、上に紹介した、RStudio からコピーする方が簡便かと思います。

SSH キーの最後には、コンピュータ名とコンピュータのアカウント名などが入っていると思います。

この設定をすれば、どちらのマシンからでも SSH 経由で Github リポジトリにアクセスできるようになります。

### B.1.9 複数のアカウントを一つのコンピュータから利用する方法

わたしも複数の GitHub アカウントを利用しています。

- `~/.ssh` 内に複数 (例では三つ)、`ssh-keygen -t rsa` でファイル作成

- id\_rsa , id\_sub1\_rsa, id\_sub2\_rsa
- 上の複数のコンピュータから利用する時に説明してあるように、SSH キーを GitHub に登録
- ~/.ssh 内の config ファイル (~/.ssh/config) を編集 (nano などを利用)
- ~/.gitconfig, ~/.gitconfig\_sub1, ~/.gitconfig\_sub2 を作成

詳しくは、参考にしたサイトを参照してください。

### B.1.10 共同作業をする場合

以下では、ひとり管理者（とりまとめ役）がいて、その人がまずは、レポジトリを作成し、何人かの編集者で、共同編集することを想定して、書きます。内容は本でなくても、プログラムでも、ノートでも構いませんが、イメージしやすくするため、本を編集することを想定して書いていきます。

1. 編集に関わる全員が、それぞれ、個人の、GitHub アカウントを作成します。
2. リポジトリをフォーク (Fork) : 共同編集する (本の) レポジトリ (オリジナル・レポジトリと呼ぶことにします\*8) にいき、フォークします。フォークすることで、編集者のリポジトリに、コピーが作成され、大元のレポジトリ (オリジナル・レポジトリ) の変更を依頼することができます。名前をつけますが、元のものと同じでも、フォークしたことがわかるようにしておいても良いでしょう。
  - 一回フォーク (Fork) したら、2 度目以降は、オリジナル・レポジトリから、フォーク (Fork) する必要はありません。ただし、改訂されているかもしれないので、更新 (Sync fork) する必要があります。GitHub 上の、フォーク (Fork) した、自分のレポジトリを見ると、更新されている場合は、オリジナル・レポジトリに、Sync することができますから、Sync し\*9、最新の状態にしておきます。よくみると、オリジナル・レポジトリとの関係が読み取れるかと思います。
3. (フォークした) リポジトリをクローン: 編集者のコンピュータにフォークして得られたリポジトリを、クローンし、自分のコンピュータ内に、ローカル・リポジトリを作成します。
4. RStudio の右上の、Git タブの右にある、New Branch に、新しい、Branch (編集用枝) を作成します。すると、その右に、Branch 名が記載されます。それが、現在の Branch 名です。その右の三角を押すと、main など、すべての Branch が見えるはずですが、編集のための Branch が表示されている (アクティブ) になっていることを確認します\*10。

\*8 upstream repository と呼ぶこともありますが、自分のリモート・リポジトリを upstream repository と呼ぶこともあるので、ここでは、オリジナル・リポジトリと呼ぶことにします。

\*9 Synk fork ボタンを押すと This branch is out-of-date などと表示され、Update branch を選択すると、update されます

\*10 New Branch の右に、現在どの Branch がアクティブになっているかが表示されています。そこに現れている Branch に、commit などで、編集を加えていくことになります。

5. 編集: 編集者のコンピュータ (RStudio 上) で編集を行います。
6. 編集結果をコミット: 編集者のコンピュータで編集した後に、簡単な説明を書いて、コミットします。
7. 編集結果をプッシュ: フォークした GitHub の自分のリポジトリに、プッシュします。
  - 実際にどうなるかをみるために、Build をする場合もあるかもしれませんが、すると、たくさんのファイルが書き換えられます。管理者も、一つ一つ丁寧に見るのは大変ですから、実際に、書き換えた、Rmd ファイルなどだけに、チェックを入れて、Commit + Push するのも良いでしょう。
  - この時点で、RStudio 上の、Branch を main に戻しておくとも良いでしょう<sup>\*11</sup>。
8. プル依頼を作成: フォークした自分の (リモート) リポジトリで、編集の内容を簡単に記述して、プルリクエスト (Pull Request) をします。
  - 編集して、プッシュした、Branch を選択し、オリジナル・レポジトリと異なった状態であることを確認します。
  - Pull Request を選択します。
  - オリジナル・レポジトリの管理者は、修正をしやすいように、修正の要点を書いておきます。
  - ここまでで終了です。管理者にメールが届きます。
  - このあと、編集者が何をするか、確認してください。
9. 管理者: 編集結果の確認: オリジナル・リポジトリの所有者は、編集結果を確認し、承認する場合は、main に、統合 (マージ) します。
  - 統合 (マージ) すると、Pull Request をした編集者に、メールが届きます。
10. 変更の同期: 編集者は、承認された、オリジナル・リポジトリを、フォークした自分のリポジトリに、同期 (Sync Fork) させます。
  - 自分のコンピュータ上の、ローカル・リポジトリの main branch も (アクティブな、Branch 名を確認して) Pull をして、更新しておくとも良いでしょう。
  - 少し、複雑な、変更の場合には、注意が出ます。詳細は省略しますが、Terminal で、一つの選択肢の、`git config pull.rebase true # rebase` を入力して、それから、もう一度、Pull をすると良いでしょう。これによって、オリジナル・リポジトリと同じ状態を維持することができます。

#### B.1.10.1 要点

- フォークして作成したりポジトリ (リモートも、ローカルも) の、main branch は、オリジナル・レポジトリと同じ状態にしておく。

---

<sup>\*11</sup> New Branch の右のところ、選択します

- 編集は、編集用に、branch を作成して、その branch を、リモート・リポジトリにプッシュして、プル・リクエストを行う。
- プル・リクエストの結果が反映されたら、リモートも、ローカルも、同期しておく。
- プル・リクエストが反映される前に、また編集するときは、別の branch を作成して、また、プル・リクエストをするのがよいでしょう。
- 管理者は、merge で、編集結果を反映するときに、注意を要しますから、管理者が編集結果を反映しやすいように、丁寧に、プル・リクエストのときの説明をしておくのがよいと思います。

### B.1.11 参考にしたサイト

#### B.1.11.1 Git - GitHub - RStudio 関連

- git – everything is local
  - About、Documentation など
- GitHub Docs: Hellow World
  - 基本的なことがコンパクトにまとまっている GitHub のサイトです。日本語もサポートしています。
- Introduction to Data Science, by Rafael A. Irizarry
  - Git and GitHub
  - edX の、データサイエンスのコースの教科書に入っています。よく、まとまっていると思います。原語は英語ですが、Google などの翻訳機能を使っても、十分理解することができると思います。Git と GitHub の概要から、Bookdown パッケージによる、電子書籍の執筆の前までは、基本的に、この教科書を参考にしていますが、それぞれのステップでの、スクリーンショットもたくさん掲載されており、確認がしやすいようになっています。
- Git-GitHub-RStudio: 個人的メモ

#### B.1.11.2 Bookdown 関連

- The bookdown book: <https://bookdown.org/yihui/bookdown/>
- The bookdown package reference site: <https://pkgs.rstudio.com/bookdown>
- How to create a bookdown book in 5 minutes: <https://www.youtube.com/watch?v=m5D-yoH416Y>

#### B.1.11.3 複数アカウント・複数のキー関連

- Using the same github account from multiple PCs
- 複数の GitHub アカウントを使い分けたい時の設定方法と Tips
- Adding a new SSH key to your GitHub account

## 付録 C

# Bookdown

## C.1 About

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports; for example, a math equation  $a^2 + b^2 = c^2$ .

### C.1.1 Usage

Each **bookdown** chapter is an .Rmd file, and each .Rmd file can contain one (and only one) chapter. A chapter *must* start with a first-level heading: **# A good chapter**, and can contain one (and only one) first-level heading.

Use second-level and higher headings within chapters like: **## A short section** or **### An even shorter section**.

The `index.Rmd` file is required, and is also your first book chapter. It will be the homepage when you render the book.

### C.1.2 Render book

You can render the HTML version of this example book without changing anything:

1. Find the **Build** pane in the RStudio IDE, and
2. Click on **Build Book**, then select your output format, or select “All formats” if you'd like to use multiple formats from the same book source files.

Or build the book from the R console:

```
bookdown::render_book()
```

To render this example to PDF as a `bookdown::pdf_book`, you'll need to install XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): <https://yihui.org/tinytex/>.

### C.1.3 Preview book

As you work, you may start a local server to live preview this HTML book. This preview will update as you edit the book when you save individual .Rmd files. You can start the server in a work session by using the RStudio add-in “Preview book”, or from the R console:

```
bookdown::serve_book()
```

## C.2 Hello bookdown

All chapters start with a first-level heading followed by your chapter title, like the line above. There should be only one first-level heading (#) per .Rmd file.

### C.2.1 A section

All chapter sections start with a second-level (##) or higher heading followed by your section title, like the sections above and below here. You can have as many as you want within a chapter.

#### An unnumbered section

Chapters and sections are numbered by default. To un-number a heading, add a `{.unnumbered}` or the shorter `{-}` at the end of the heading, like in this section.

## C.3 Cross-references

Cross-references make it easier for your readers to find and link to elements in your book.

### C.3.1 Chapters and sub-chapters

There are two steps to cross-reference any heading:

1. Label the heading: `# Hello world {#nice-label}`.
  - Leave the label off if you like the automated heading generated based on your heading title: for example, `# Hello world = # Hello world {#hello-world}`.
  - To label an un-numbered heading, use: `# Hello world {-#nice-label}` or `{# Hello world .unnumbered}`.



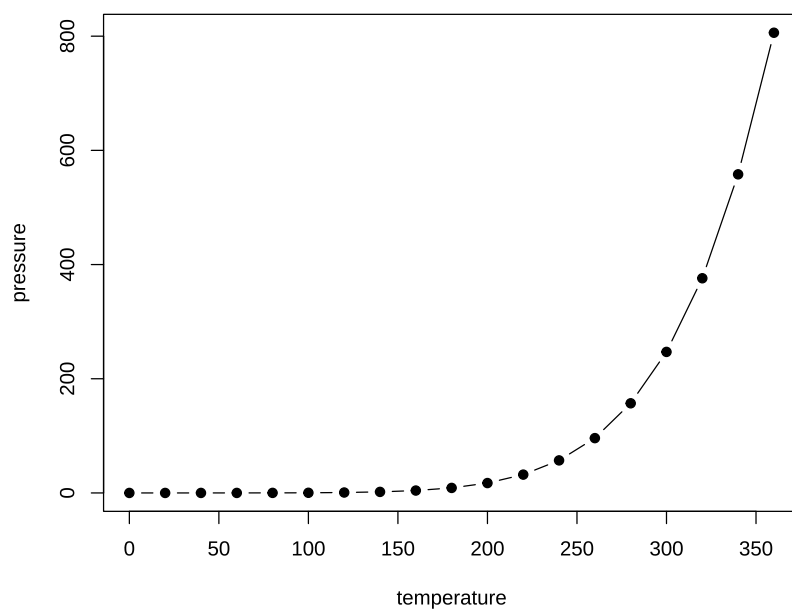
2. Next, reference the labeled heading anywhere in the text using `\@ref(nice-label)`; for example, please see Chapter C.3.
  - If you prefer text as the link instead of a numbered reference use: any text you want can go here.

### C.3.2 Captioned figures and tables

Figures and tables *with captions* can also be cross-referenced from elsewhere in your book using `\@ref(fig:chunk-label)` and `\@ref(tab:chunk-label)`, respectively.

See Figure C.1.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```



☒C.1 Here is a nice figure!

Don't miss Table C.1.

```
knitr::kable(
 head(pressure, 10), caption = 'Here is a nice table!',
 booktabs = TRUE
)
```

## C.4 Parts

You can add parts to organize one or more book chapters together. Parts can be inserted at the top of an `.Rmd` file, before the first-level chapter heading in that same file.

表C.1 Here is a nice table!

| temperature | pressure |
|-------------|----------|
| 0           | 0.0002   |
| 20          | 0.0012   |
| 40          | 0.0060   |
| 60          | 0.0300   |
| 80          | 0.0900   |
| 100         | 0.2700   |
| 120         | 0.7500   |
| 140         | 1.8500   |
| 160         | 4.2000   |
| 180         | 8.8000   |

Add a numbered part: # (PART) Act one {-} (followed by # A chapter)

Add an unnumbered part: # (PART\\*) Act one {-} (followed by # A chapter)

Add an appendix as a special kind of un-numbered part: # (APPENDIX) Other stuff {-} (followed by # A chapter). Chapters in an appendix are prepended with letters instead of numbers.

## C.5 Footnotes and citations

### C.5.1 Footnotes

Footnotes are put inside the square brackets after a caret `^[]`. Like this one <sup>\*1</sup>.

### C.5.2 Citations

Reference items in your bibliography file(s) using `@key`.

For example, we are using the **bookdown** package (Xie, 2023) (check out the last code chunk in `index.Rmd` to see how this citation key was added) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015) (this citation was added manually in an external file `book.bib`). Note that the `.bib` files need to be listed in the `index.Rmd` with the YAML `bibliography` key.

The `bs4_book` theme makes footnotes appear inline when you click on them. In this example book, we added `csl: chicago-fullnote-bibliography.csl` to the

---

<sup>\*1</sup> This is a footnote.

`index.Rmd` YAML, and include the `.css` file. To download a new style, we recommend: <https://www.zotero.org/styles/>

The RStudio Visual Markdown Editor can also make it easier to insert citations: <https://rstudio.github.io/visual-markdown-editing/#/citations>

## C.6 Blocks

### C.6.1 Equations

Here is an equation.

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (\text{C.1})$$

You may refer to using `\@ref(eq:binom)`, like see Equation (C.1).

### C.6.2 Theorems and proofs

Labeled theorems can be referenced in text using `\@ref(thm:tri)`, for example, check out this smart theorem C.1.

**Theorem C.1.** *For a right triangle, if  $c$  denotes the length of the hypotenuse and  $a$  and  $b$  denote the lengths of the **other** two sides, we have*

$$a^2 + b^2 = c^2$$

Read more here <https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html>.

### C.6.3 Callout blocks

The `bs4_book` theme also includes special callout blocks, like this `.rmdnote`.

You can use **markdown** inside a block.

```
head(beaver1, n = 5)
#> day time temp activ
#> 1 346 840 36.33 0
#> 2 346 850 36.34 0
#> 3 346 900 36.35 0
#> 4 346 910 36.42 0
#> 5 346 920 36.55 0
```

It is up to the user to define the appearance of these blocks for LaTeX output.

You may also use: `.rmdcaution`, `.rmdimportant`, `.rmdtip`, or `.rmdwarning` as the block name.

The R Markdown Cookbook provides more help on how to use custom blocks to design your own callouts: <https://bookdown.org/yihui/rmarkdown-cookbook/custom-blocks.html>

## C.7 Sharing your book

### C.7.1 Publishing

HTML books can be published online, see: <https://bookdown.org/yihui/bookdown/publishing.html>

### C.7.2 404 pages

By default, users will be directed to a 404 page if they try to access a webpage that cannot be found. If you'd like to customize your 404 page instead of using the default, you may add either a `_404.Rmd` or `_404.md` file to your project root and use code and/or Markdown syntax.

### C.7.3 Metadata for sharing

Bookdown HTML books will provide HTML metadata for social sharing on platforms like Twitter, Facebook, and LinkedIn, using information you provide in the `index.Rmd` YAML. To setup, set the `url` for your book and the path to your `cover-image` file. Your book's `title` and `description` are also used.

This `bs4_book` provides enhanced metadata for social sharing, so that each chapter shared will have a unique description, auto-generated based on the content.

Specify your book's source repository on GitHub as the `repo` in the `_output.yml` file, which allows users to view each chapter's source file or suggest an edit. Read more about the features of this output format here:

[https://pkgs.rstudio.com/bookdown/reference/bs4\\_book.html](https://pkgs.rstudio.com/bookdown/reference/bs4_book.html)

Or use:

```
?bookdown::bs4_book
```

## 付録 D

# To Do リスト

この文書について

1. はじめに
  - 1.1 データサイエンスとは
    - 1.1.1 なぜ、データサイエンスが最近注目されているのか
    - 1.1.2 なぜ、一人ひとりが、データサイエンスを、学ぶ必要があるのでしょうか
    - 1.1.3 AI の活用
  - 1.2 「データサイエンスを始めましょう」の特徴
    - 1.2.1 学習者として想定しているのは
    - 1.2.2 オープン・パブリックデータの活用
    - 1.2.3 世界のデータをみること
    - 1.2.4 目標としていること
  - 1.3 学習方法について
  - 1.4 参考
2. 学ぶ内容
  - 2.1 データサイエンス入門
  - 2.2 第一部 パブリックデータ
  - 2.3 第二部 基本
  - 2.4 第三部 国際機関などのデータの活用
  - 2.5 第四部 探索的データ分析 Exploratory Data Analysis
  - 2.6 第五部 分析例
  - 2.7 付録
3. はじめてのデータサイエンス
  - 3.1 データサイエンスの実際
  - 3.2 R のパッケージを活用
    - 3.2.1 準備 Setup
    - 3.2.2 データ取得 Import data

- 3.2.3 データ構造の確認
  - 3.2.4 必要に応じて整形 Transform data
  - 3.2.5 視覚化 data visualization
  - 3.2.6 データの理解 Understand data
  - 3.2.7 さまざまな視覚化
  - 3.3 練習
  - 3.4 プロジェクト
  - 3.5 まとめ
4. 世界開発指標
- 4.1 世界銀行 (World Bank)
    - 4.1.1 世界銀行オープンデータ
    - 4.1.2 データカタログ
    - 4.1.3 世界開発指標 (WDI)
    - 4.1.4 オープンデータの定義 (Open Data Defined)
  - 4.2 世界開発指標 (WDI)
    - 4.2.1 世界のさまざまな課題
    - 4.2.2 ダッシュボード・データの取得・データコード
  - 4.3 世界銀行以外の国際機関のパブリックデータ
  - 4.4 持続的開発目標 (SDGs) データ
  - 4.5 課題

## PART I PUBLIC DATA

### 11. Public Data

- 5.1 オープンデータ
  - 5.1.1 Open Government Data Toolkit: Open Data Defined
- 5.2 日本から世界を見る
- 5.3 世界銀行 (World Bank)
  - 5.3.1 世界開発指数 (World Development Indicator (WDI))
  - 5.3.2 World Bank: WDI - World Development Indicators
  - 5.3.3 世界開発指標 (World Development Indicator)
  - 5.3.4 例
- 5.4 世界のさまざまな課題から見る
- 5.5 OECD
- 5.6 UN Data
- 5.7 Our World in Data
- 5.8 Eurostat

## PART II BASICS

### 21. R Studio で R

- 6.1 はじめに

- 6.2 R と R Studio
- 6.3 R と R Studio のインストール
  - 6.3.1 R のインストール
  - 6.3.2 R Studio のインストール
  - 6.3.3 動作確認
  - 6.3.4 トラブル・シューティング (1)
- 6.4 クラウド - Posit Cloud
  - 6.4.1 クラウドサービス How to Start Posit Cloud
  - 6.4.2 PositCloud Shared Project
  - 6.4.3 Posit Primers and Cheat Sheet
- 6.5 R のその他の利用方法
- 6.5.1 Google colab で R
- 6.5.2 CoCalc で R と RMarkdown

## 22. R Markdown

- 7.1 Reproducible and Literate Programming
  - 7.1.1 目的、問いなど
  - 7.1.2 データについて
  - 7.1.3 コードについて
  - 7.1.4 グラフについて
  - 7.1.5 まとめ：R Markdown の目的
- 7.2 準備：パッケージのインストール
- 7.3 R Notebook
- 7.4 日本語のテンプレート
- 7.5 R Markdown いくつかの Output
- 7.6 YouTube Video - rmarkdown -7.6.1 RMarkdown で PDF を作成するときの注意

## 23. R Basics

- 8.1 プロジェクト - Project
- 8.2 コンソールで実行 - Run in Console
  - 8.2.1 最初の四つ
  - 8.2.2 アサインメント、ヘルプ
  - 8.2.3 おすすめ
  - 8.2.4 練習
- 8.3 RStudio について
  - 8.3.1 四つの窓枠とタブ Four Panes and Tabs
- 8.4 R Script 実行記録
  - 8.4.1 R Script の作成
  - 8.4.2 R Script による実行
  - 8.4.3 練習
  - 8.4.4 Tips

- 8.5 パッケージ - Packages
  - 8.5.1 パッケージのインストール
  - 8.5.2 備考
- 8.6 練習問題 Posit Primers
  - 8.6.1 最初の演習 The Basics – r4ds: Explore, I
- 8.7 参考文献 References
- 8.8 YouTube Video - getstarted

## PART III INSTITUTIONAL DATA

### 31. World Bank

- 9.1 World Development Indicator (WDI)
  - 9.1.1 指標 Indicators (WDI)
  - 9.1.2 指標 WDI (World Development Indicators)
  - 9.1.3 指標のコード、WDI code を探してみよう
  - 9.1.4 指標 WDI の例
  - 9.1.5 練習 1. - 調べてみたい WDI 指標とそのコード
- 9.2 WDI パッケージ
  - 9.2.1 指標 WDI 検索
  - 9.2.2 指標 WDI データのダウンロード
- 9.3 可視化 Visualization
  - 9.3.1 グラフ 1
  - 9.3.2 グラフ 2
  - 9.3.3 テンプレート Templates -9.4 課題 Assignment

## PART IV EDA

### 41. 探索的データ解析

- 10.1 探索的データ解析 (EDA) とは
- 10.2 探索的データ解析 (EDA) の一例
  - 10.2.1 データの取得と読み込み - Data Import
  - 10.2.2 データ変形・整形 - Data Transformation
  - 10.2.3 可視化 Data Visualization
  - 10.2.4 データモデリング Data Modeling

## PART V EXAMPLES

### 51. Example 1

## APPENDIX

### 81. 日本語の扱いについて

- A.1 日本語・中国語・韓国語
  - A.1.1 パッケージをロード



- A.2 Base R でタイトルに日本語
- A.3 列名や、データに日本語
- A.4 kable で表示
- A.5 ggplot でグラフを作成
- A.6 備考：
- A.7 参考：日本語の表示について
  - A.7.1 データファイルの読み込み
  - A.7.2 図の中のテキスト
  - A.7.3 R Markdown の出力
  - A.7.4 bookdown
  - A.7.5 参考としたもの

## 85. IT ツール

- B.1 Git と GitHub
  - B.1.1 概要
  - B.1.2 はじめかた
  - B.1.3 GitHub にあるリモート・リポジトリ (Remote Repo) から始める場合
  - B.1.4 自分のコンピュータのリポジトリ (Local Repo) から始める場合
  - B.1.5 GitHub Pages について
  - B.1.6 Bookdown パッケージによる、電子書籍の執筆
  - B.1.7 大きなファイルに関すること
  - B.1.8 複数のコンピュータから利用する方法
  - B.1.9 複数のアカウントを一つのコンピュータから利用する方法
  - B.1.10 共同作業をする場合
  - B.1.11 参考にしたサイト

## 90. Bookdown

## 91. To do list



## 参考文献

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2023). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.35.