

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)

2023-09-23

目次

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とも、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こと、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います。この書を利用するためには「データサイエンスを教えてみませんか」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、少しづつ、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時間、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かでしょうか。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、純粹数学を生業（なりわい）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、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。

少しづ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きています。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。日本語を使わなきこと、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 ?? 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。皆さんがある、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、皆さんがあなたと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、課題をみつけたり、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 Popper, 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、洞察を得るために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、英語での問い合わせ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、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言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ことが必要で、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言うことです。

そのことが、もう一步進んで、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、自然なことば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ますため、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. 創新と競争力の追求: 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、人的な作業負荷を軽減し、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、創新と競争力の追求、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はなく、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、提供されていると言うことです。

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、以下のようない点が挙げられます。

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と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方をしてしまいます。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。

そのためには、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、問い合わせ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。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います。

AI はまだ、発展途上で、ここに書いたことも、少し経つと、古くなってしまうことでしょう。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ない。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、アフリカでもな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応用できないことを覚えておこう。
9. 犯人探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、小さな一步を重ねよう。

(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的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ことが必要だと思っています。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。

むろん、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い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って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もある、問い合わせ、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と、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があります。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

だと思います。

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いってください。

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います。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います。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います。

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、なにを実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、その下に出力し、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言っているからです。

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。

指導してくだ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、アドバイスを受けたりす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います。

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、少しづつ説明しながら、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います。

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。少しづ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、紹介できなかつ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

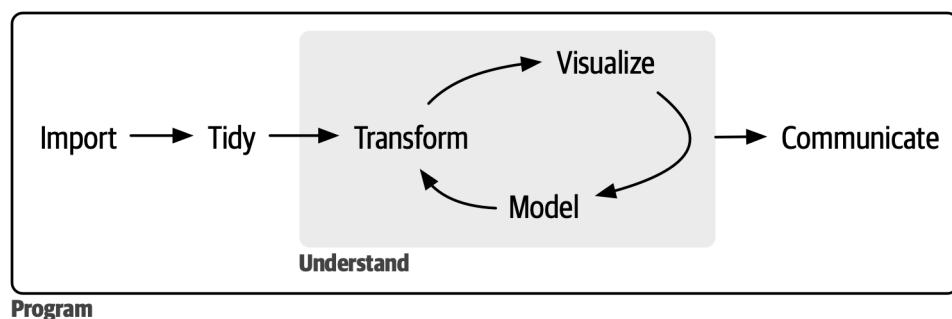
はじめてのデータサイエンス

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、少しづつ説明します。



はじめに書きましたが、基本的には、問い合わせ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、さらに問い合わせ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、一つづつ決まっています。

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、ブラウザー（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 tidyrr    1.3.0
#> v purrrr   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_dgp) 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" "Afghanistan"
#> $ 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 Asia"
#> $ 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 income"
#> $ 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、4.940878e+12（この文書では、幅の都合で、4.9e+12と表示されているかもしれません）は、Scientific notation と言われるもので、

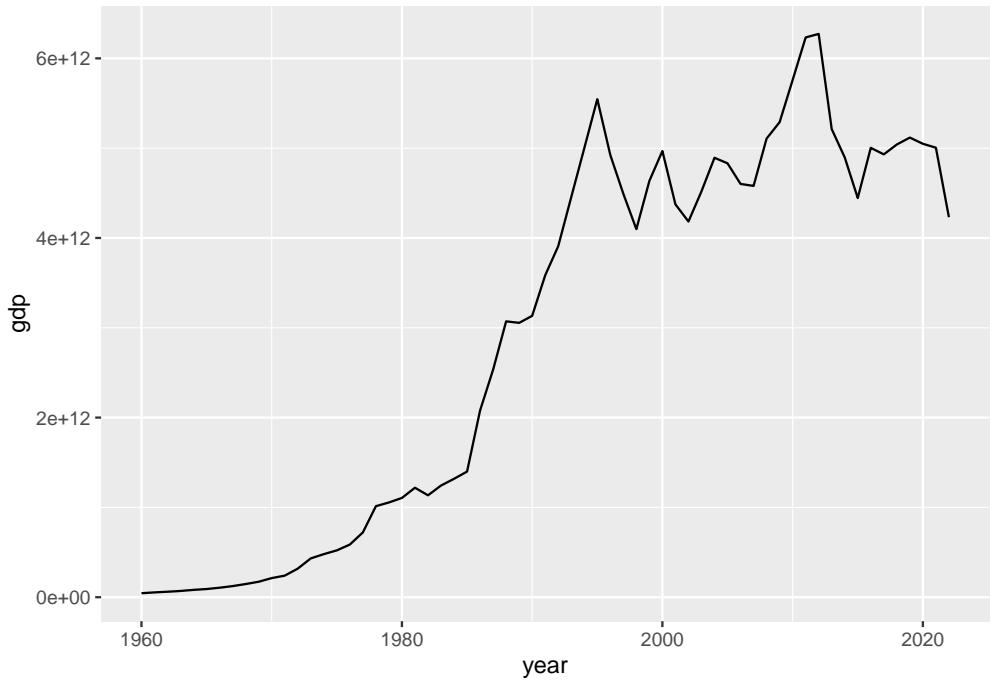
$$4.940878 \times 10^{12} = 4,940,887,800,000$$

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こと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、

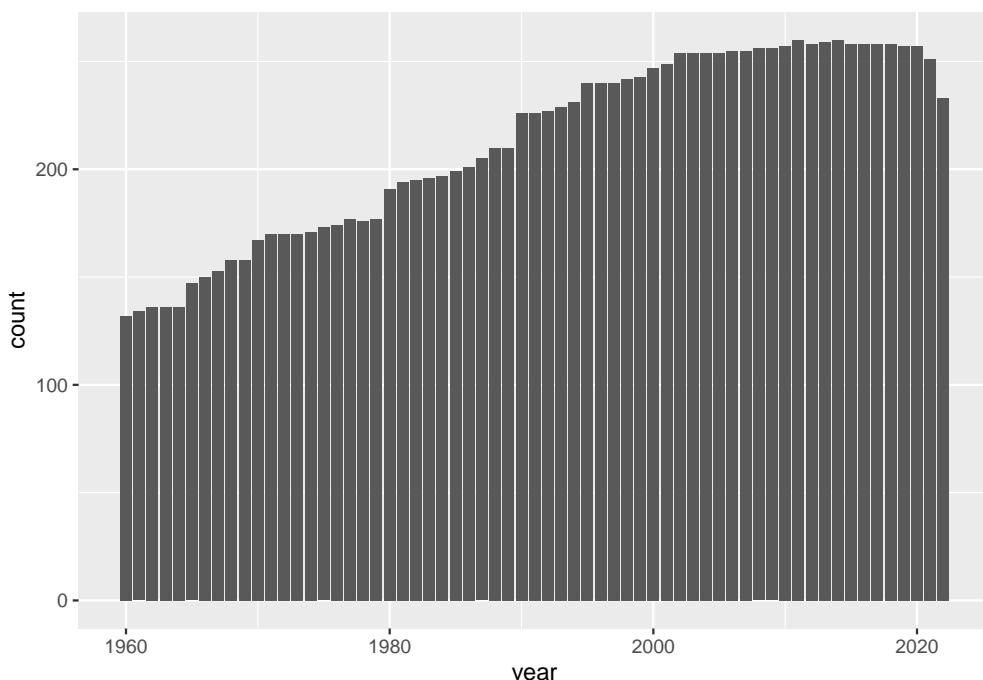
下落は、なにがあったのかも気になりませんか。これは、世界的な傾向なのでしょうか。日本だけでしょうか。他にも似た傾向の国があるのでしょうか。将来はどうなっていくのでしょうか。などなど。

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 == 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>
```

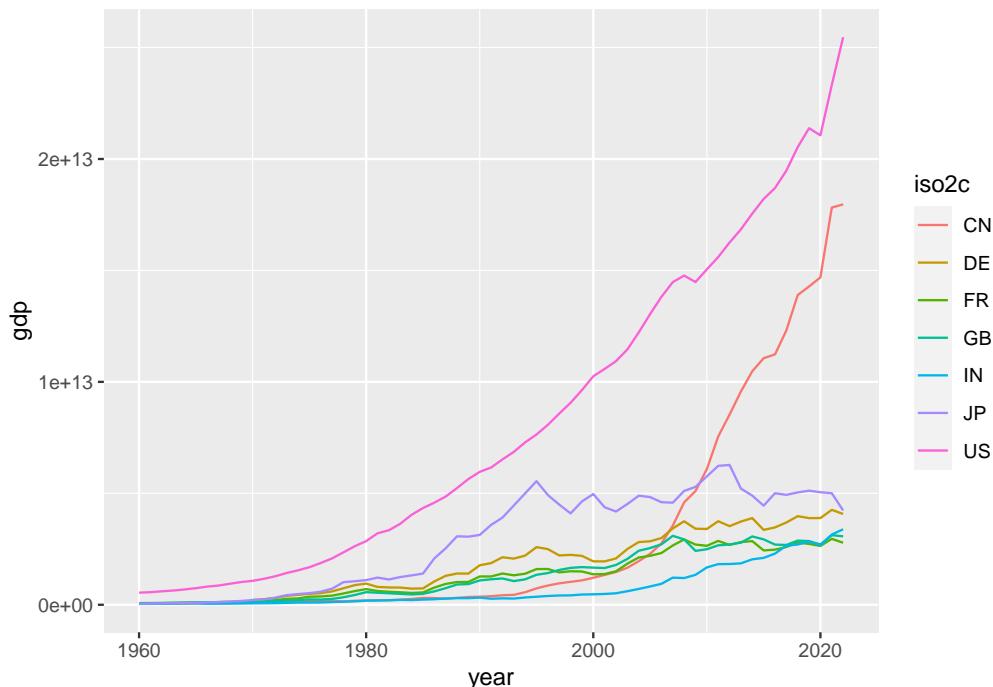
これは、グラフではありませんが、これも一つの可視化と考えられないことはありません。

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` するといういみで、`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 `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ペーセント表示に変えてください。というコードです。

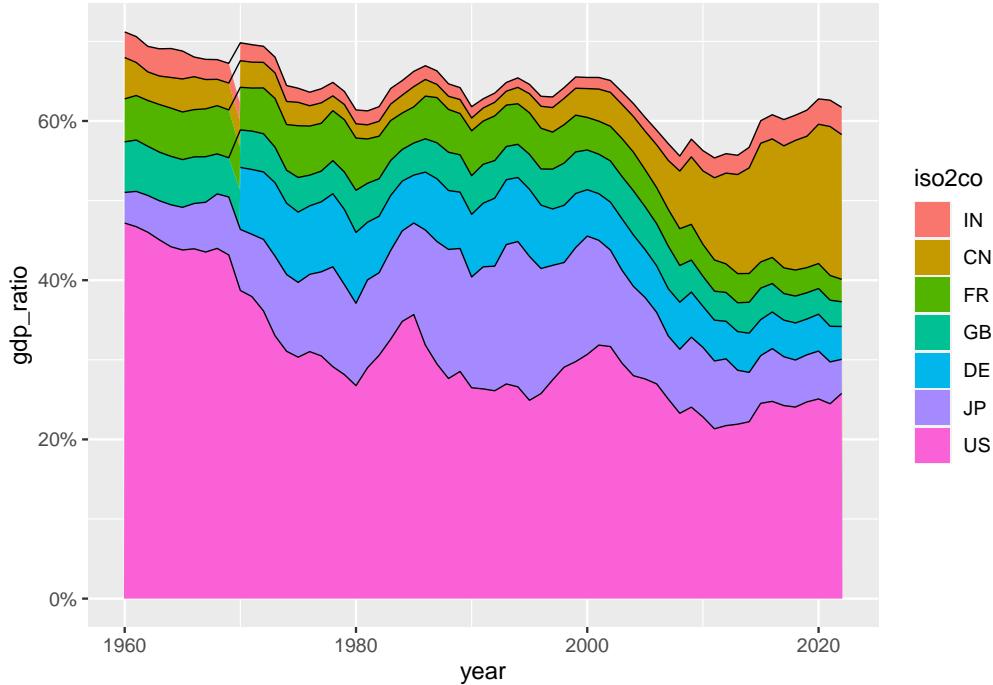
```
ggplot(aes(x = year, y = gdp_ratio, fill = iso2c)) + geom_area() +
geom_line(col = "black", position = "stack", linewidth = 0.3) +
scale_y_continuous(labels = scales::percent_format(accuracy = 1))
```

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。

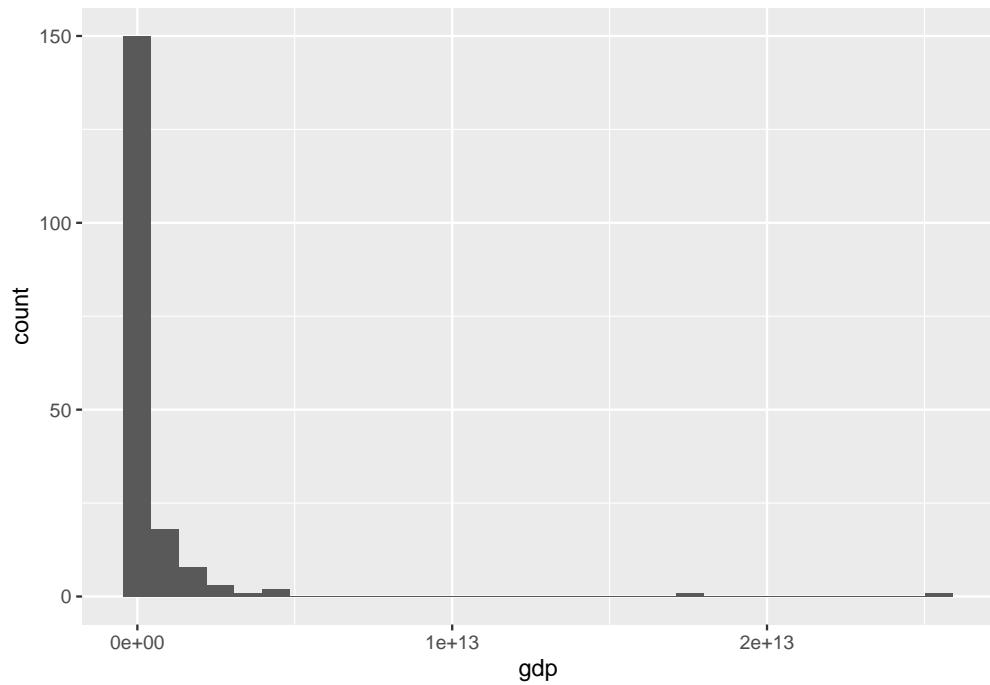
主要国で、60%～70% を占めていることがわかります。それぞれの国や、幾つかの国の影響力や、その時代による変化も、ある程度みることができるよう思います。

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`.
```

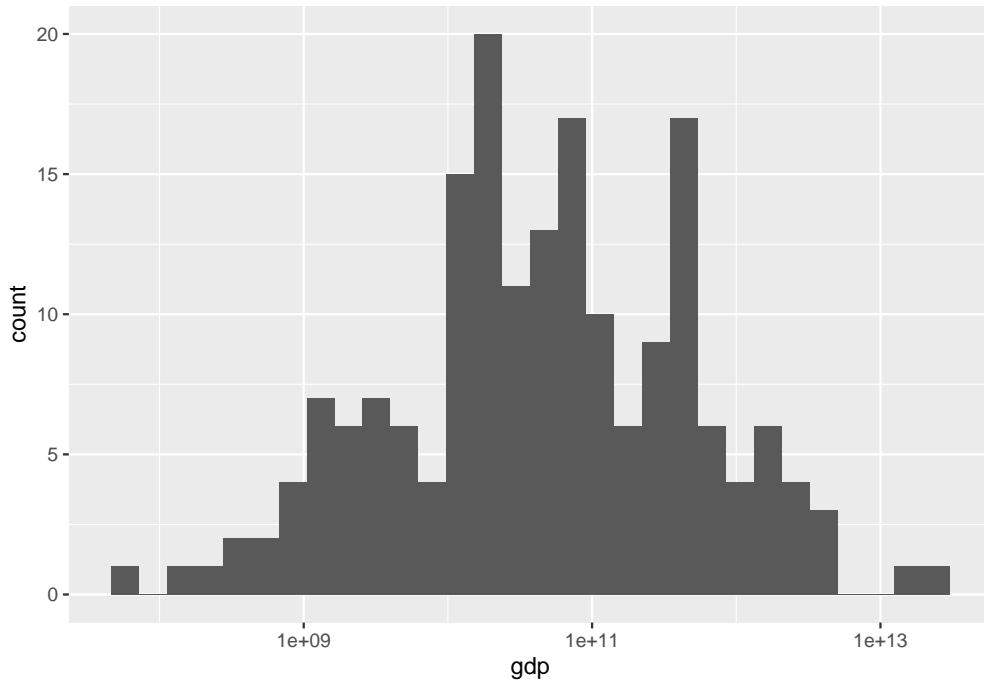


小さいところに集中していることがわかりますが、`scale_x_log10()`を加え、対数軸をとってみます。

$\log_{10}(1000) = 3$, $\log_{10}(1000000) = 6$, $\log_{10}(1000000000) = 9$ 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`.
```

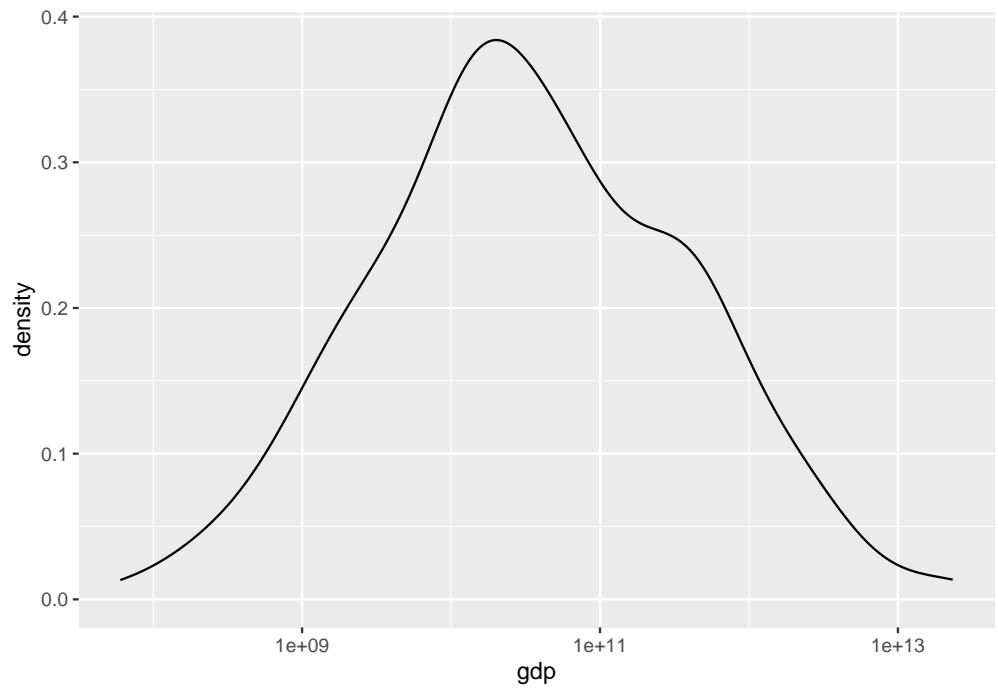


幅を変更したり、分ける個数を変更するには `binwidth = 0.5` や、`bins = 20` を、
`geom_histogram()` のかっこの中に入れます。

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。

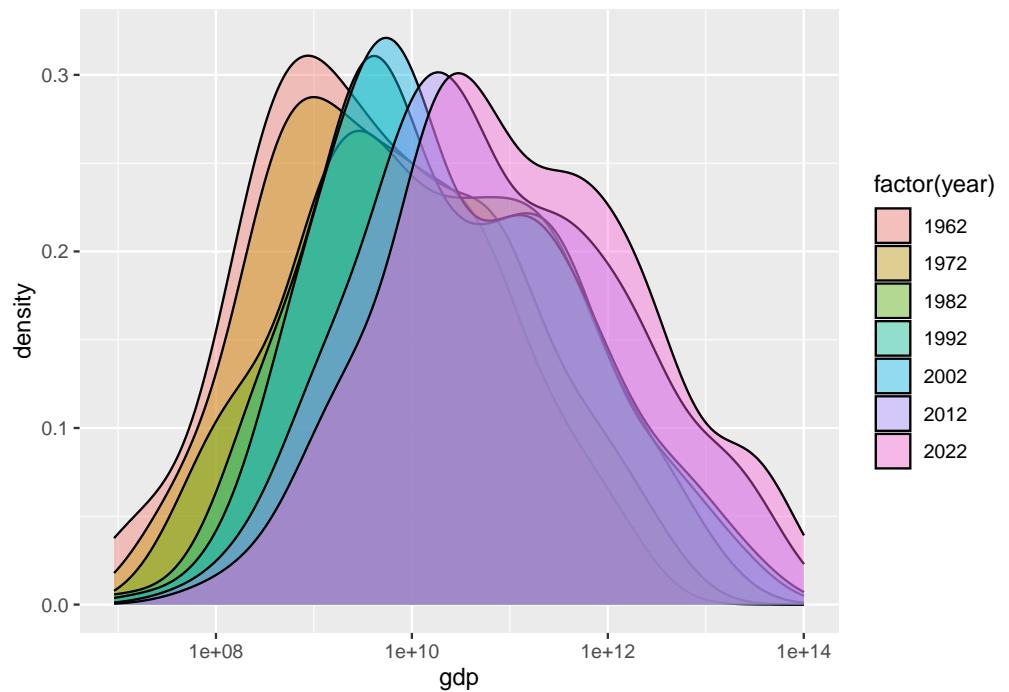
```
df_gdp |> drop_na(gdp) |>
  filter(year == 2021) |> filter(region != "Aggregates") |>
  ggplot(aes(gdp)) + geom_density() + scale_x_log10()
```



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 の変化を見てみます。alpha の値は透明度です。

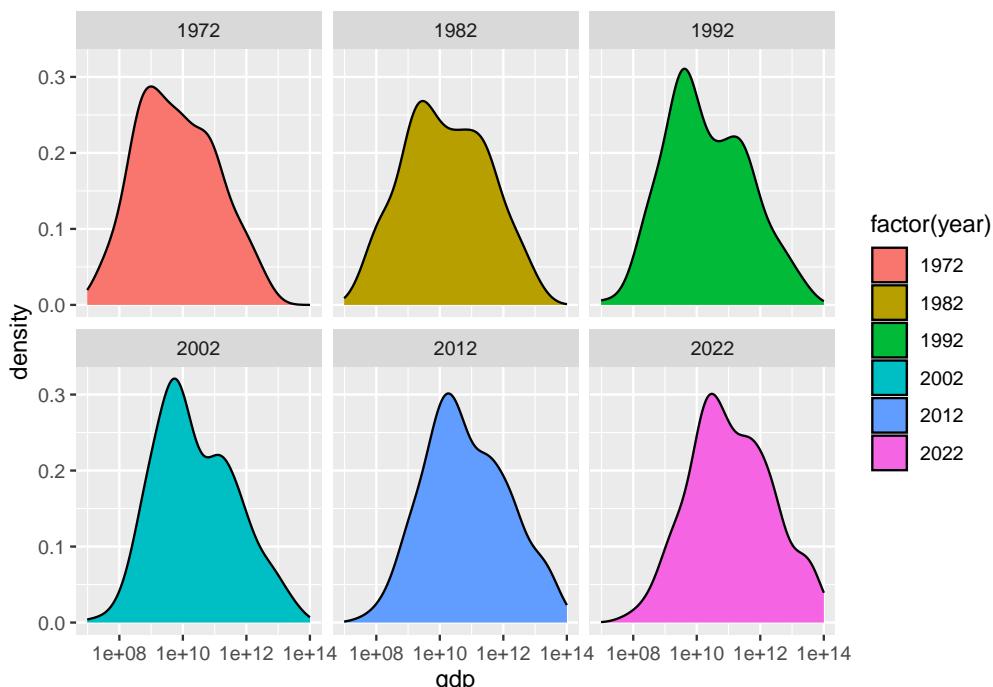
```
df_gdp |> drop_na(gdp) |>
  filter(year %in% c(1962, 1972, 1982, 1992, 2002, 2012, 2022)) |>
  ggplot(aes(gdp, fill = factor(year))) + geom_density(alpha = 0.4) + scale_x_log
```



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。

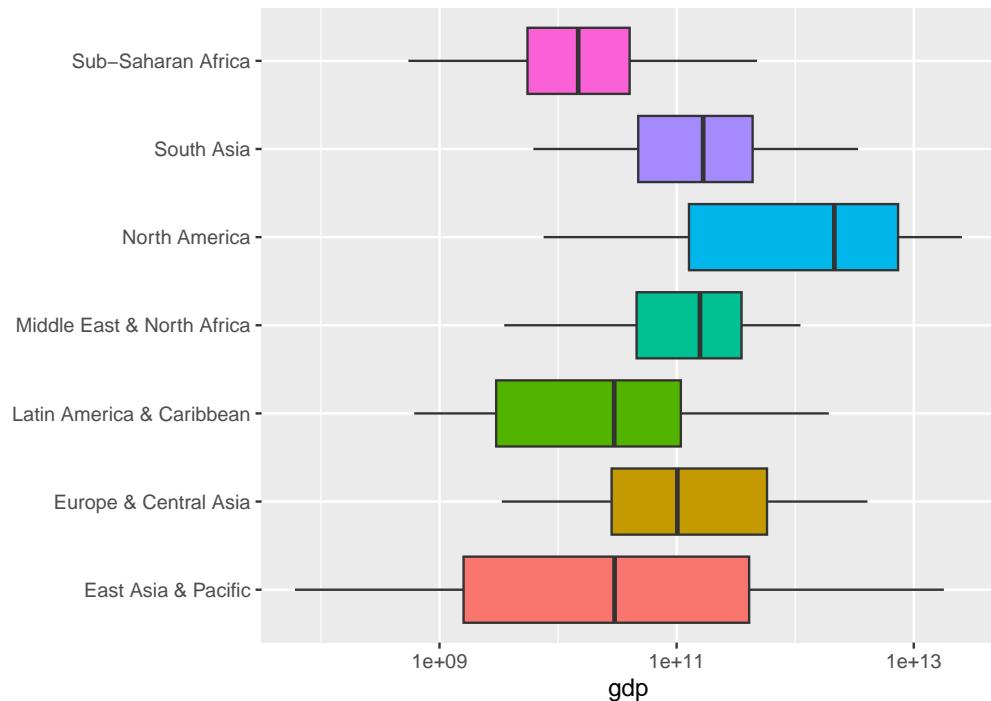
```
df_gdp |> drop_na(gdp) |>
  filter(year %in% c(1972, 1982, 1992, 2002, 2012, 2022)) |>
  ggplot(aes(gdp, fill = factor(year))) +
  geom_density() + scale_x_log10() + facet_wrap(~year)
```



3.2.7.12 Fig 10. 地域ごとの GDP の分布

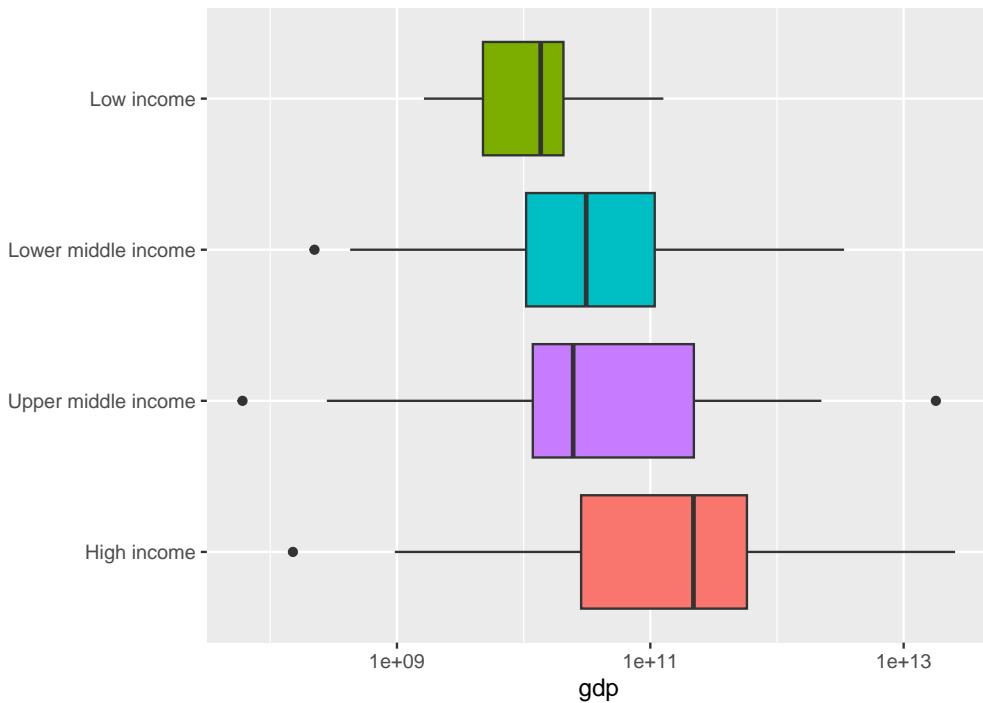
いくつかのグループごとに分布をみてみることも可能です。それには、箱ひげ図（Boxplot）が有効です。箱ひげ図では、そのグループの国を値の大きさの順にならべて、四等分し、その、真ん中の二つが箱の部分に対応しています。また、真ん中の太い線は、中央値（median）を表しています。詳しくは、後ほど説明します。

```
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

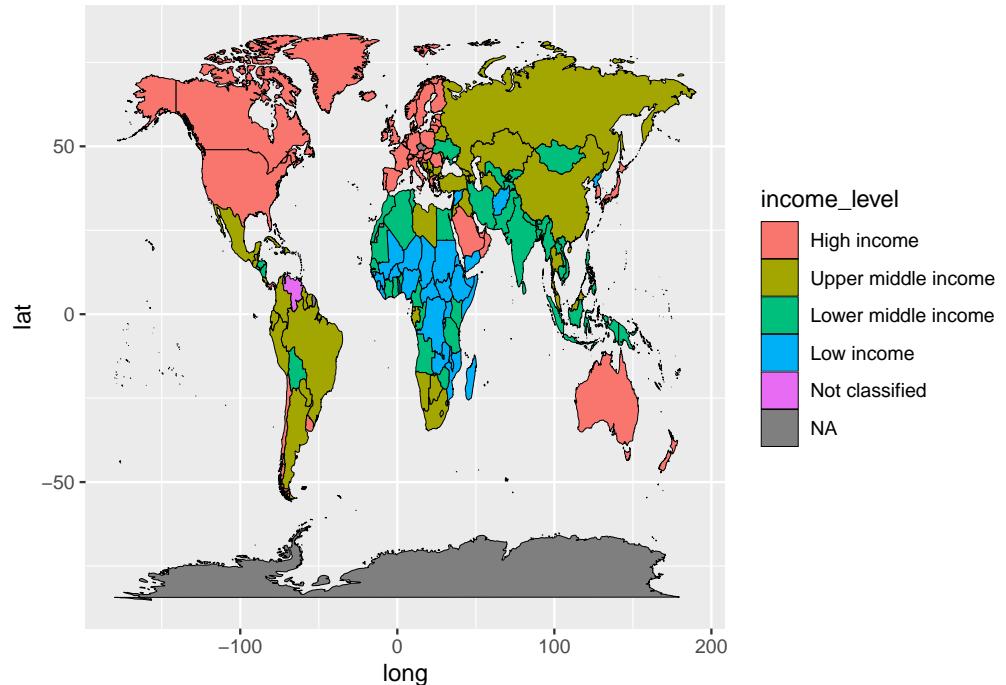
地図で、国の収入の多寡 (income level) をみてみましょう。

```
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
```

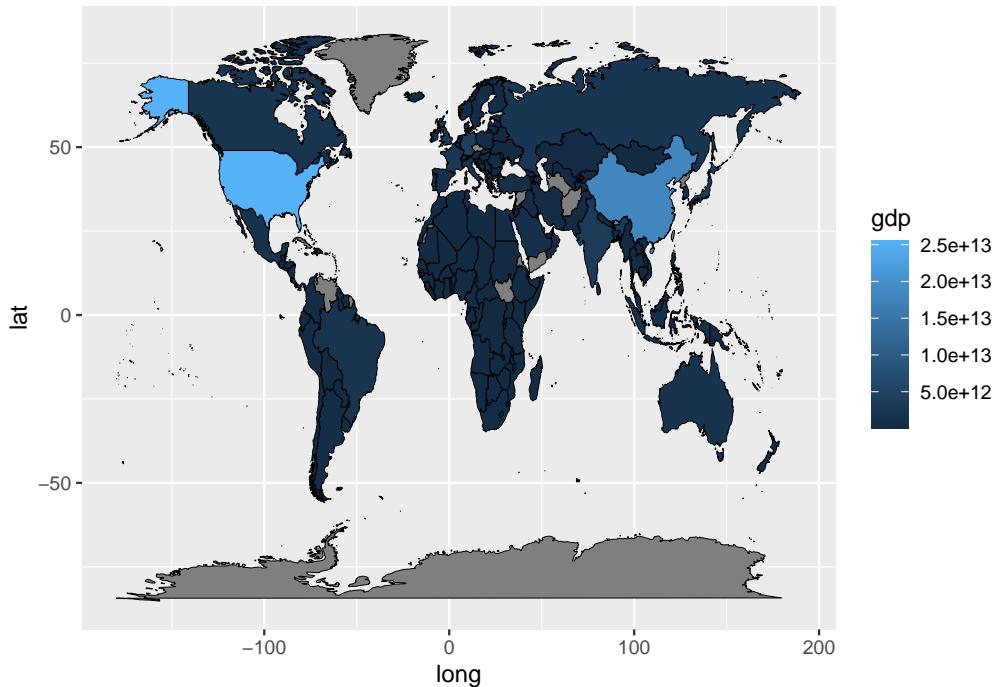
3.2.7.15 Fig 12. Income Level による色分け地図

```
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、問い合わせ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 に変えてみましょう。

```
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。

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づつ、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、ご理解いただけると思います。

- 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言われています。
- 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ななく、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、わかることが色々とある。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、問い合わせ持ち、他の視覚化などを用いて、さらに、理解を深めることができます。
- 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います。

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います。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、ブラウザー (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、このサイトがお薦めです。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、以下のようないい處があります。

- 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、経済的状況などによって、グループに分けてある帶もあります。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事ができます。

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) とも呼ばれますぐ、正確な定義があるわけではなく、各

機関によって、少しづつ考え方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読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いく必要があります。

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なども、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、少しづつ、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、記録を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から。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れたり、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ター（何を意味するかは少しづ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、ブラウザー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：最近のブラウザーには、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、ブラウザーによって、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：最近は、ブラウザーの翻訳機能の質も向上しているので、不要かもしれませんのが、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：最近は、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、画像になっているから、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に便利です。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良いでしょう。

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、縦棒だったりと、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、データを見る経験して行きたいと思います。

第Ⅰ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いって構いません。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、使いやすくなっています。少しづつ、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づつ慣れていく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、少しづ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 の分野に分かれています。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、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できます。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、挑戦してみてはいかがでしょうか。

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）からも利用可能です。
 - 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ことが有効です
-

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、見していくとともに、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、ブラウザー (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事ができます。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ます。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ことができます。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っ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を感じる場合があります。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 GPD 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 GPD 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、翻訳機能を使って、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。記録しておくと、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の値が大きいため、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なになってくること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 & Pasific 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帶のダウントロードからも、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帶からも、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 カ国のものに変わります。
3. 上の、Styles を選択し、Chart 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. Chart 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つたり、少し古いデータまでしかなかつ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。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推し、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 (ポインター) 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) : カテゴリーごとに、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、カテゴリー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、カテゴリー (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) : カテゴリー : 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ます、WDIのように、規格が統一されていません。

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ます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年には何番目だかわかりますか。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ます。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。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 円などと出ています。

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（※）ランクイング（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ことができます。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、見ることができます。

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で書いていきます。

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います。

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

がついています。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 で利用することができます。

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思います。
 - 左下の、Console タブに、`tinytex::install_tinytex()` をコピーして、入力し、Enter または、Return キーを押し実行します。
 - もう一度、Kit 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います。

- 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。（この作業はしなくとも、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ネーム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
る可能性がありますが、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われます。

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見ることが大切ですが、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良いでしょう。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が。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自体は、とても有効なことですので、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つてないフォルダーを作成します。

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、RStuio 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、いつく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 :

- 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.)
- 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.)
- 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.)
- 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、

- まず、自分の、PositCloud アカウントにログインする。
- 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ます、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が多いと思うので、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unkか、テキスト・チャunk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` と最初に書けば、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、Mathematica 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 だけではなく、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。少しづつ、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まし、R Markdwon も今後ともサポートされることですので、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が、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、問い合わせなど

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

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。少しづ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 しておくと良いでしょう。

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。ブラウザー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。

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 shou
```

```
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ますので、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、参考程度にしていただくのがよいかと思います。

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います。
 - 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 ...
```

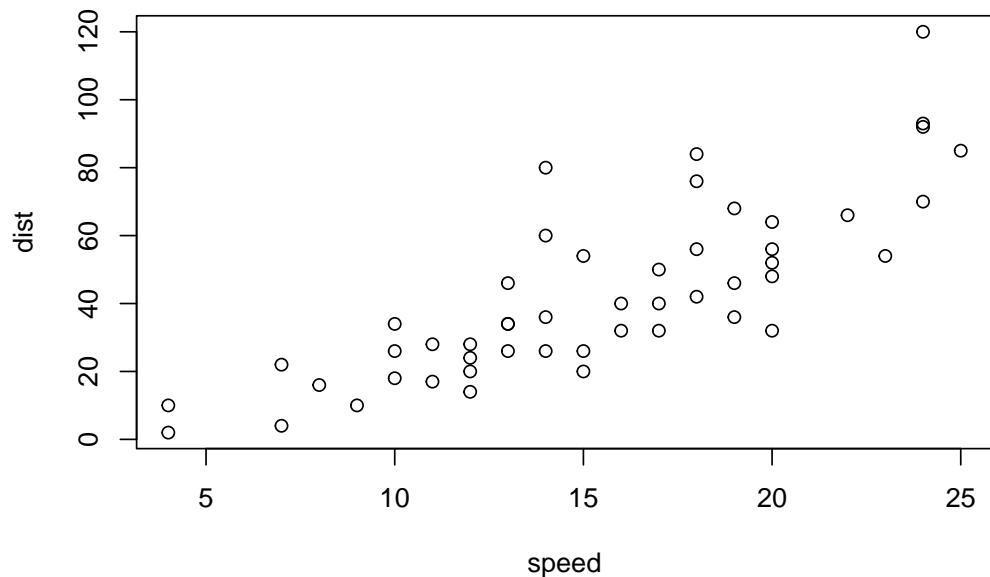
`str(cars)` は、`cars` という、R に付属している、データの構造 (structure) を表示します。`data.frame` とありますが、これは、矩形になったデータ（各列の長さがおなじ）の一番簡単なクラスの名前で、2 変数、それぞれが、50 個の数値データ (numerical data) からなっていることがわかります。

`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、それを用いると、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` といれても、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、十分すぎるくらいの、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。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。

日本語に戻したいときは、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 なども、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、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良いでしょう。`datasets` のリストを見るには、`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、少しづ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, underscores

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 recovered

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、問い合わせれば、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、パッケージー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` などいくつかに、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 しておくと良いでしょう。

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ことがあります。特に問題が起らなければ、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。解説を読んでいただけでは、データサイエンスは身につきません。

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。教科書のようには、できませんが、参考になる部分もあるかと思いますので、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、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)` などで、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、下のように設定します。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 tidyrr    1.3.0
#> v purrrr   1.0.2
#> -- 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 cor
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` キーを推しながら、`-` 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、`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。

次の、summary() を使うと、Species の三種類がなにかわかりますが、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。

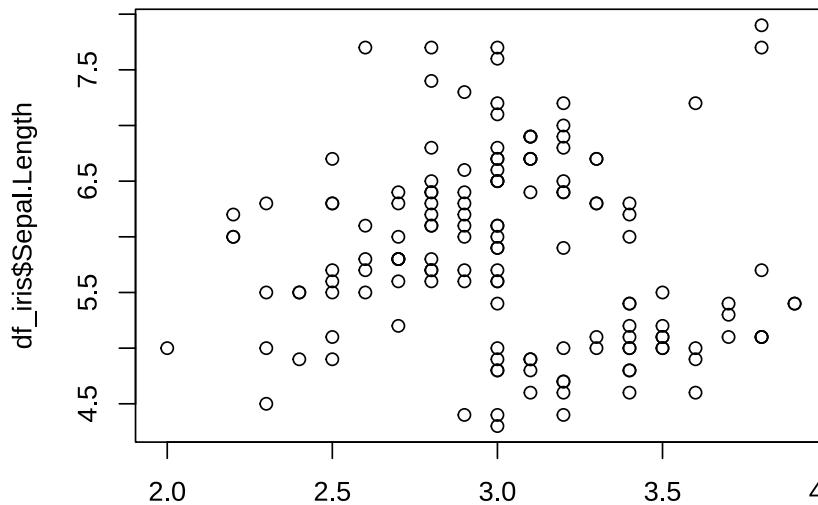
これは、あやめ (iris) の、長さや幅のデータでしたが、単位は何でしょうか。それも、`help` で調べることができます。Help の検索ボックスに、`iris` と入れてみてください。この場合は、Description (概要) に次のように書かれています。

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、萼片の長さと幅、花弁の長さと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

`df_iris$Sepal.Length`

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、説明などができます。

ここまででは、`tidyverse` を使わずにできることですが、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、下のものと同じです。

```
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、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, 0.4, 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, 0.4, 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。

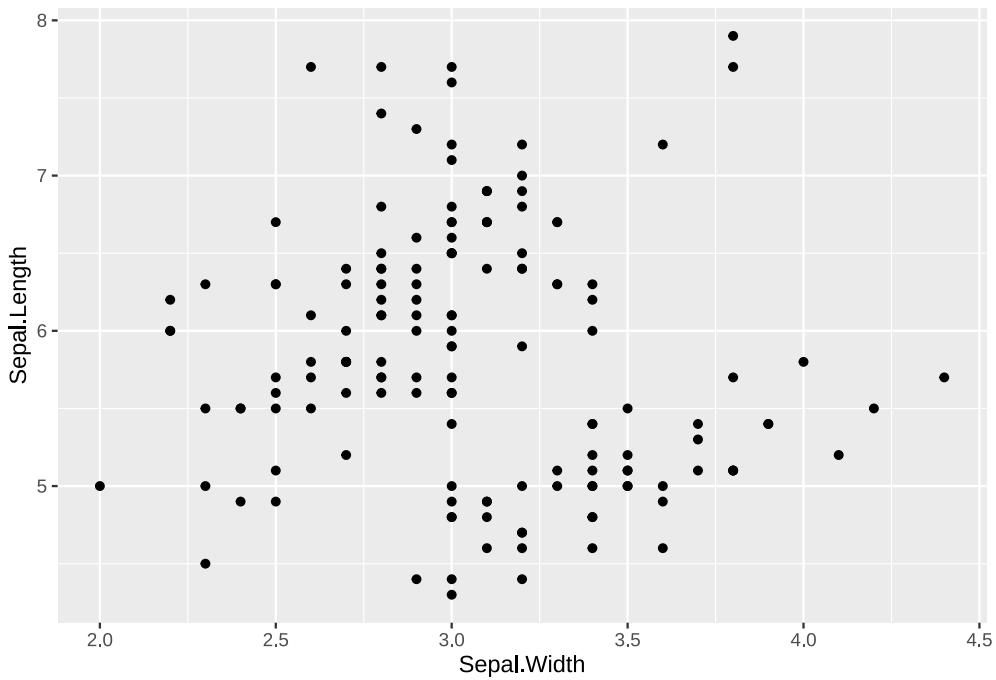
```
typeof(df_iris$Sepal.Length)
#> [1] "double"

class(df_iris$Sepal.Length)
#> [1] "numeric"
```

17.2.2.3 ggplot2: グラフの描画

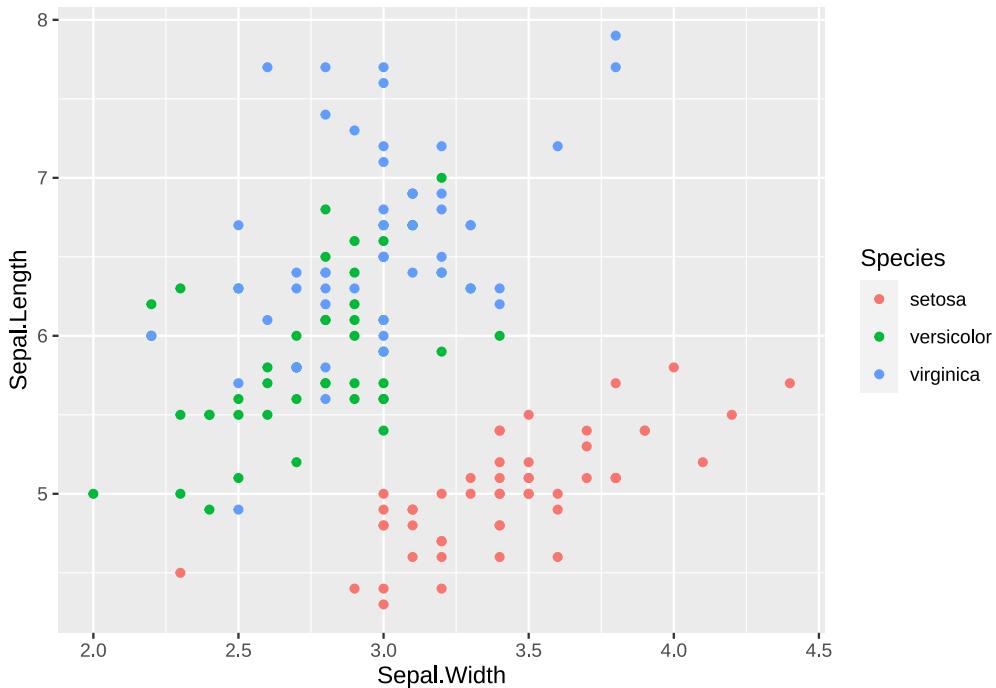
`plot` の拡張ですが、tidyverse パッケージ群の核をなすものでもあります。詳細は、視覚化で扱います。

```
df_iris |> ggplot(aes(Sepal.Width, Sepal.Length)) + geom_point()
```

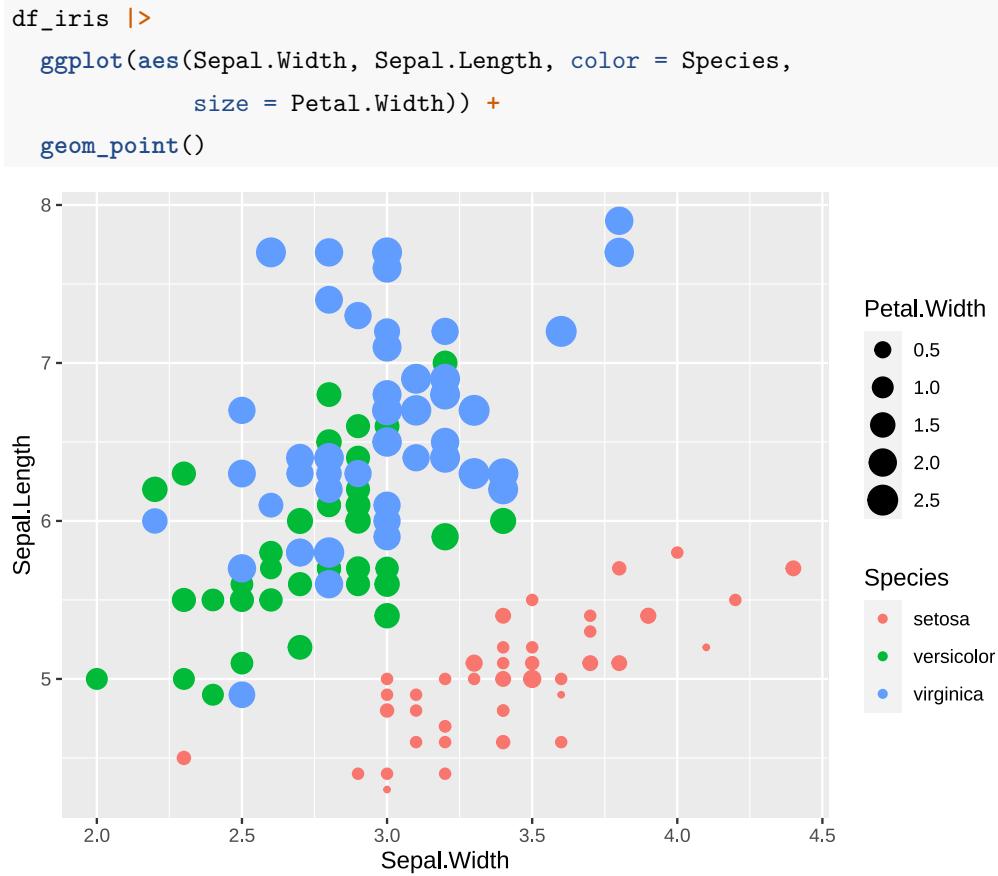


さまざまな描画が可能ですが、一番、一般的な、散布図、plot に対応するものを書きました。ggplot の中の、aes (aesthetic) の部分に、x 軸、y 軸に対応する変数 (列名) を書きます。種類 (Species) ごとに色を変える場合には、color = Species とします。

```
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。



少しづ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 AFG   AFG    1960    32.5  8622466      NA
#> 2 Afghanistan AFG   AFG    1961    33.1  8790140      NA
#> 3 Afghanistan AFG   AFG    1962    33.5  8969047      NA
#> 4 Afghanistan AFG   AFG    1963    34.0  9157465      NA
#> 5 Afghanistan AFG   AFG    1964    34.5  9355514      NA
#> 6 Afghanistan AFG   AFG    1965    35.0  9565147      NA
#> 7 Afghanistan AFG   AFG    1966    35.5  9783147      NA
#> 8 Afghanistan AFG   AFG    1967    35.9  10010030     NA
#> 9 Afghanistan AFG   AFG    1968    36.4  10247780     NA
#> 10 Afghanistan AFG  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も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 message

df_wdi_extra
#> # A tibble: 16,758 x 15
#>   country     iso2c iso3c    year status lastupdated lifeExp
#>   <chr>       <chr> <chr> <dbl> <lgls> <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、少しづ見ていきながら、現実世界のデータを扱えるようにしていきましょう。

`glimpse` を使ってみて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, ~

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-
#> $ 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、表で見ていても、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) が決まっていて、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、ftable (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 は整数值で 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りますが、`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、書かなくとも同じです。
- * `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 練習

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

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

17.5 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、わから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。

17.5.1 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。

17.5.1.1 base R

- `unique()`

17.5.1.2 tidyverse

- `slice()`, `slice_head()` , `slice_max()`
- `glimpse()`
- `ggplot()` , `geom_point()`

17.5.1.3 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 tidyverse 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 tidyverse   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()` 変数を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、medianなど）を求めるに対応します。
- `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、二つの表の扱い方は、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	<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>

^{*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。

「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が、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。参考：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、少しづ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()` はその値までの最大、`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 versoin 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、変形などをし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、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、平均を求めていきます。

```
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, ||

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 message

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も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, 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

df_wdi_extra
#> # A tibble: 16,758 x 15
#>   country     iso2c iso3c    year status lastupdated lifeExp
#>   <chr>       <chr> <chr> <dbl> <lgls> <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>
```

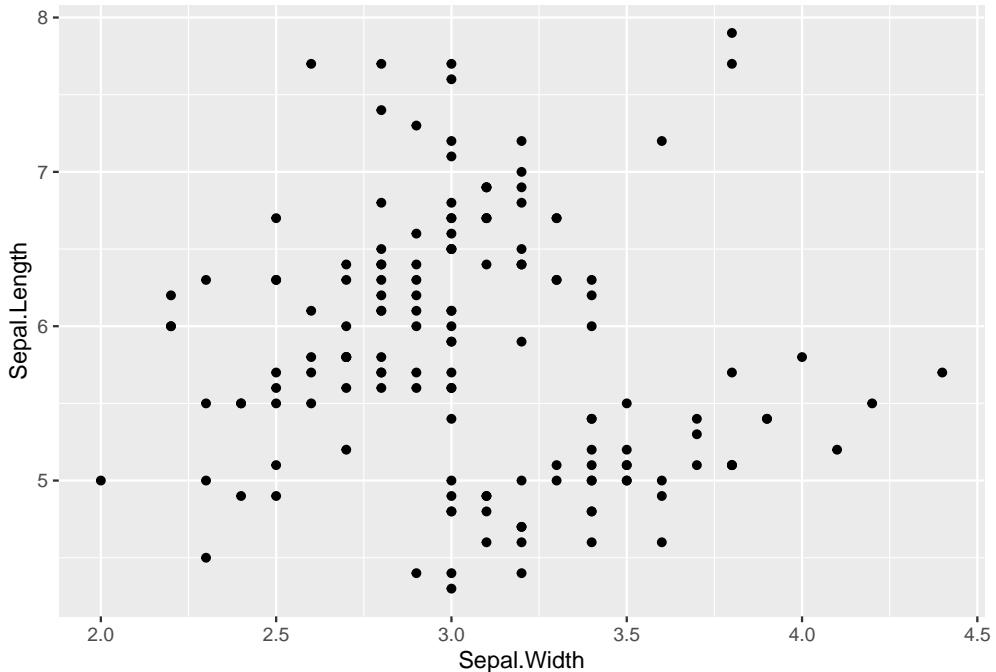
18.4.2 ggplot2

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

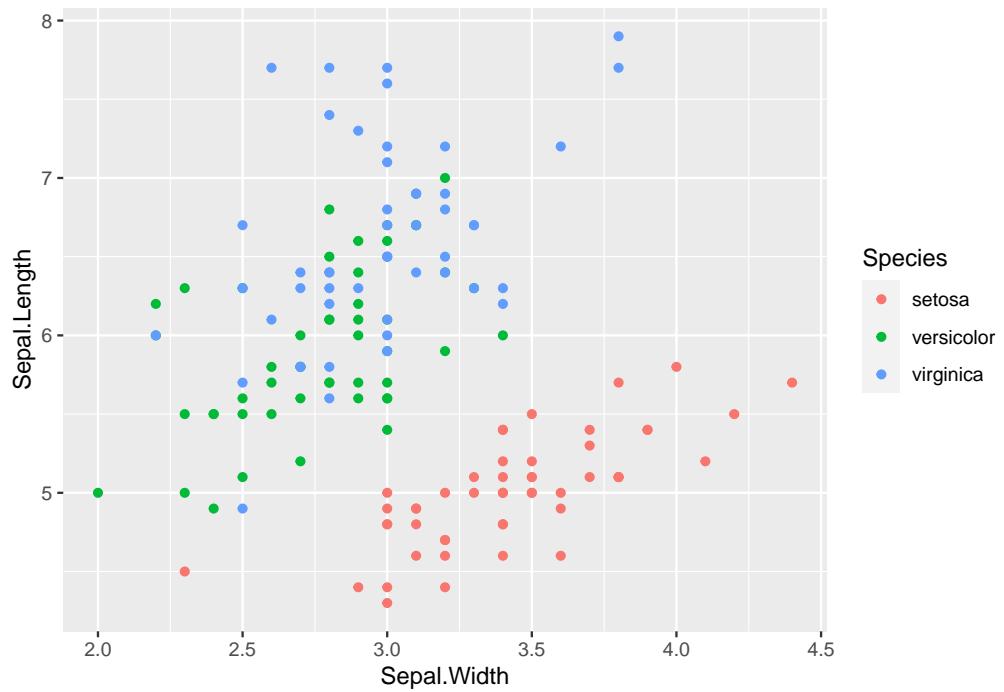
詳細は、次の、視覚化のところで説明しますが、視覚化のためには、データの変形、整理が必要であることを、理解していただくために、散布図と、箱ひげ図の基本的な場合のみ、以下のような形式で利用します。

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ができます。

```
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。あとで見てみましょう。Google Public Data Explorer を使ったときに、Income Level というものがありました。覚えていませんか。

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)
```

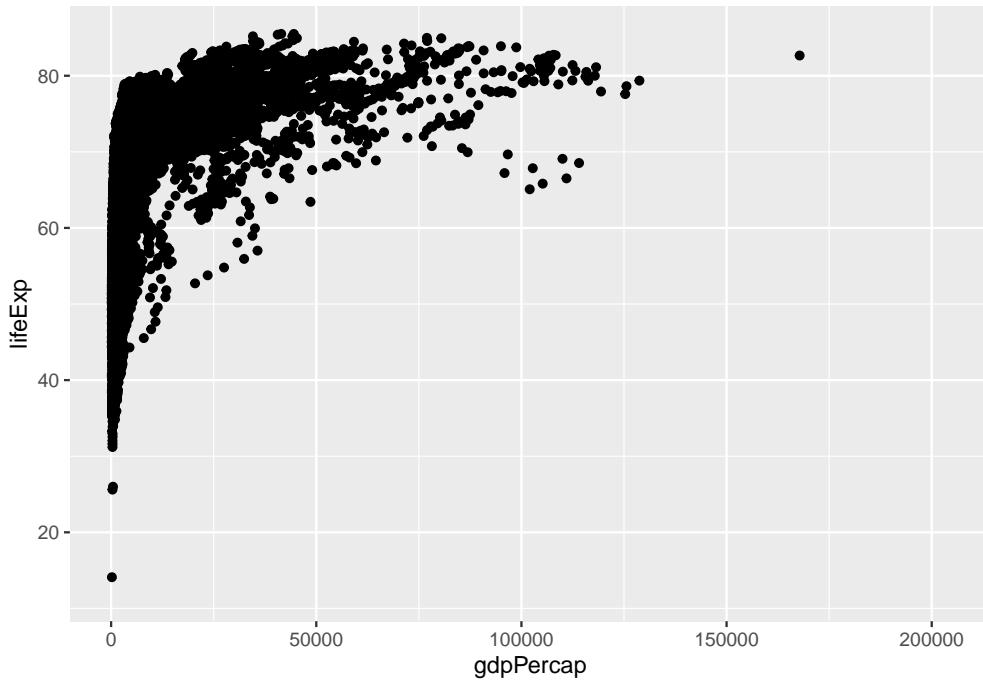
```
#> # A tibble: 1 x 1
#>   country
#>   <chr>
#> 1 Venezuela, RB
```

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。

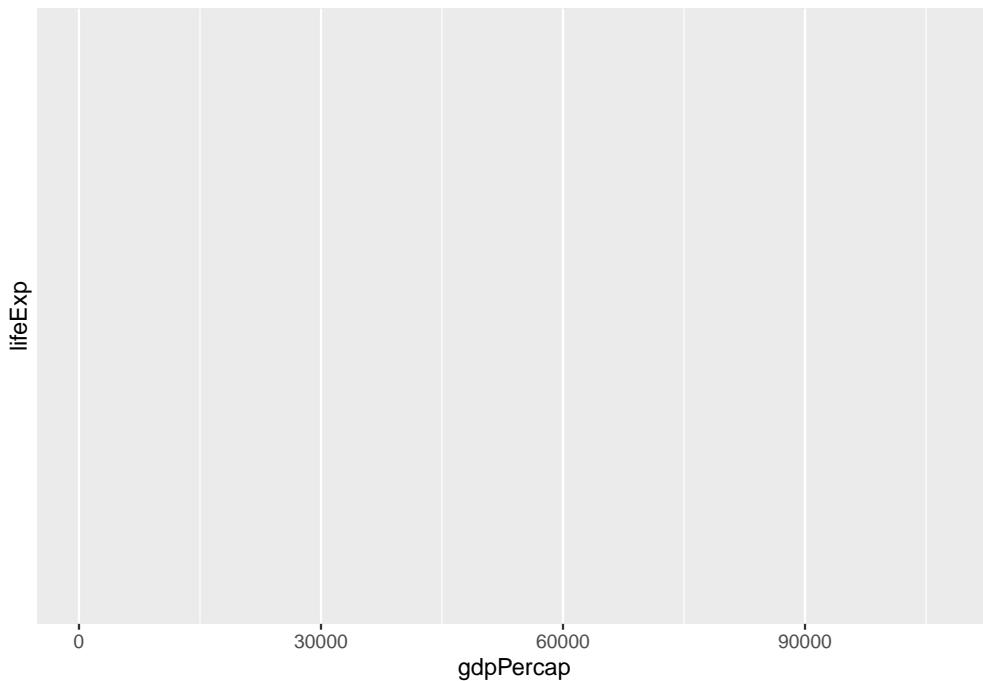
```
df_wdi3c |> ggplot(aes(gdpPercap, lifeExp)) + geom_point()
#> Warning: Removed 3981 rows containing missing values
#> (`geom_point()`).
```



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。

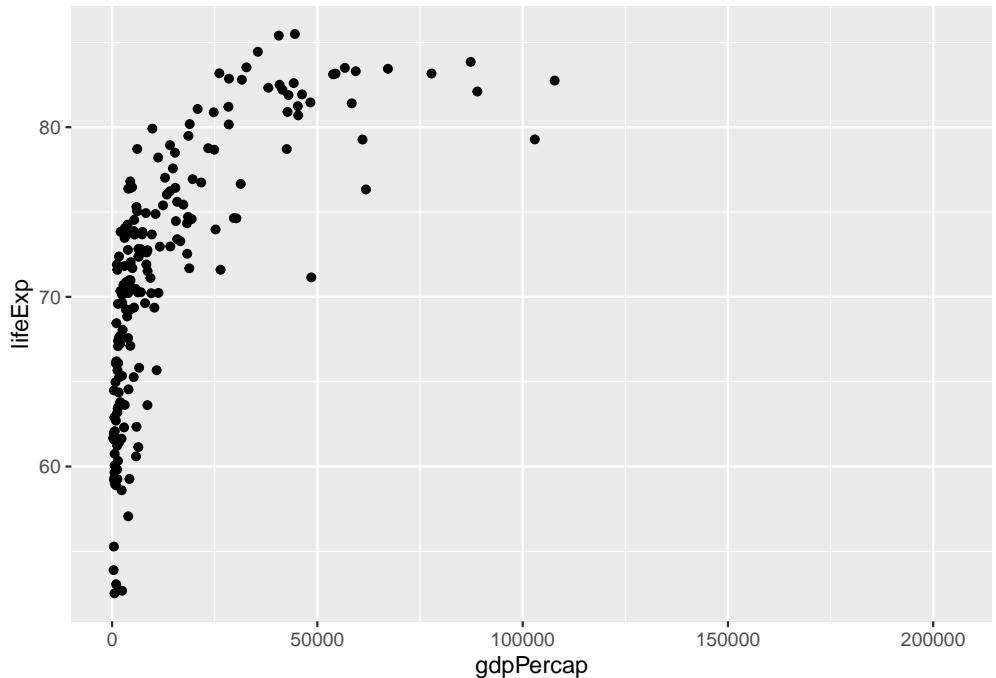
まずは、すべての年について、値を使っていますから、2022 年に限ってみましょう。

```
df_wdi3c |> filter(year == "2022") |>
  ggplot(aes(gdpPercap, lifeExp)) + geom_point()
#> Warning: Removed 216 rows containing missing values
#> (`geom_point()`).
```



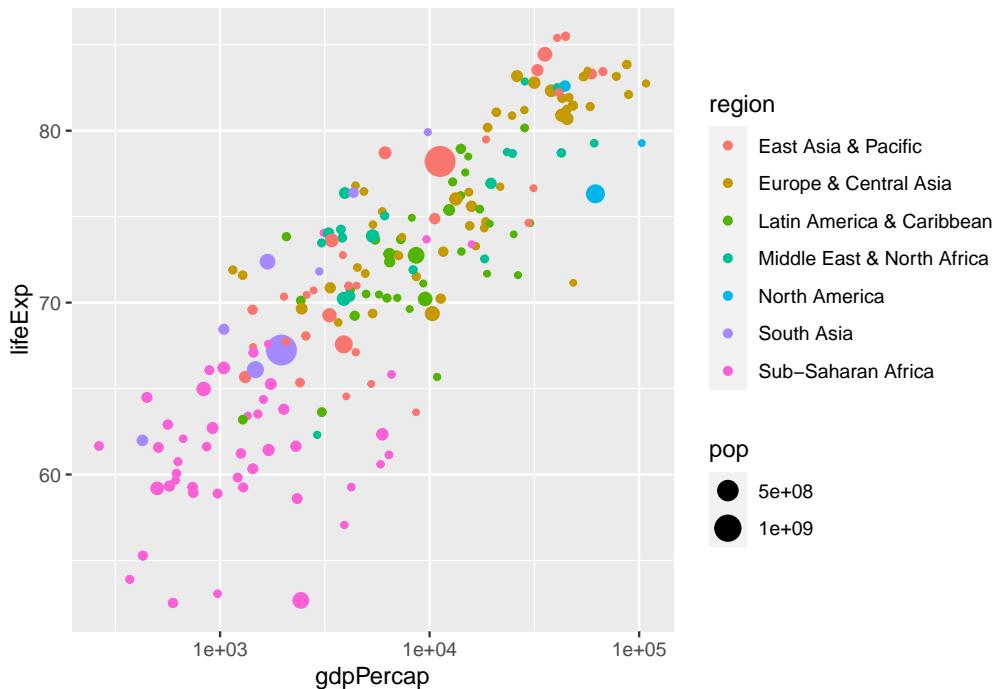
どうも、何も現れません。2021年ならどうでしょうか。

```
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、点の大きさで表したり、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、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 States 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 Kingdom 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ぐ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) |>
  summarise(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、a 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。

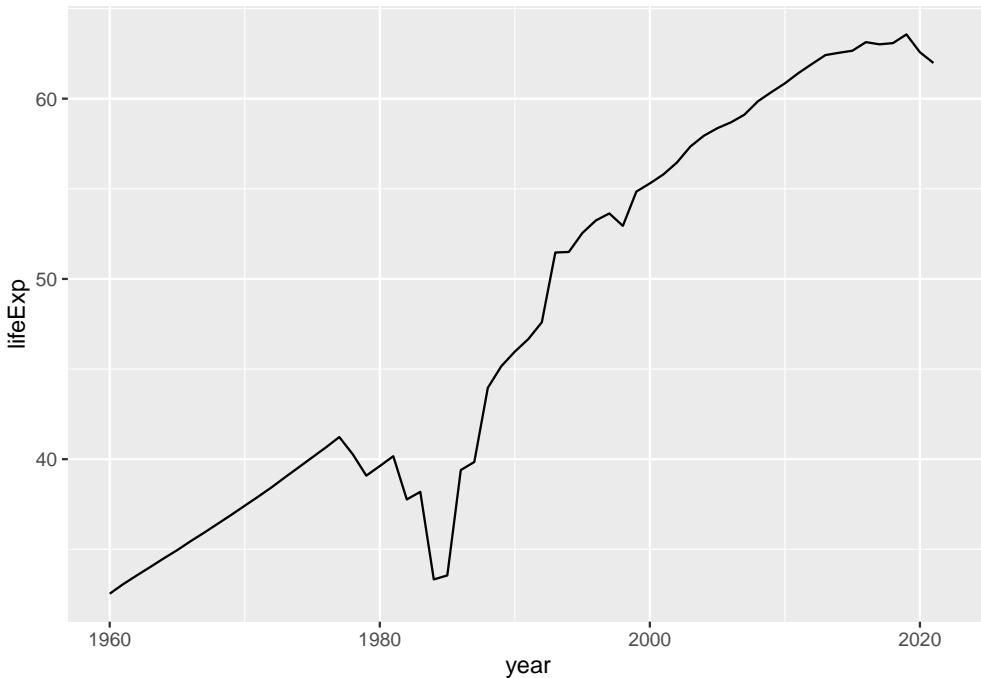
```
df_wdi |> filter(country == "Afghanistan")
#> # A tibble: 63 x 7
#>   country iso2c iso3c year lifeExp      pop gdpPercap
#>   <chr>    <chr> <chr> <dbl>     <dbl>   <dbl>      <dbl>
#> 1 Afghanistan AF    AFG    1960    32.5  8622466      NA
#> 2 Afghanistan AF    AFG    1961    33.1  8790140      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。

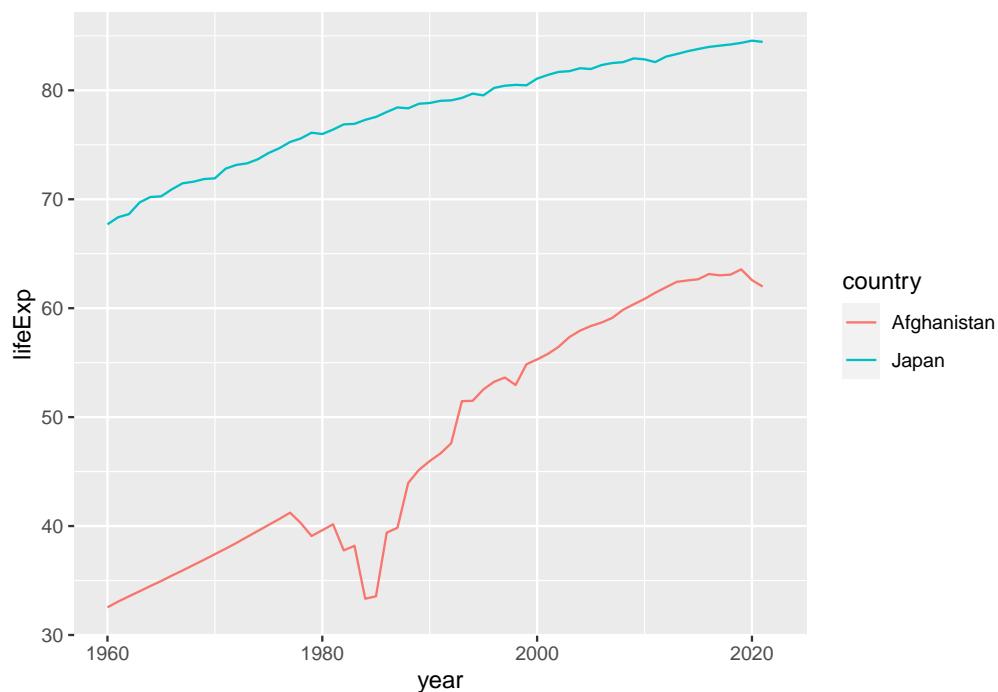
```
df_wdi %>% filter(country %in% c("Afghanistan", "Japan"))

#> # 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"
```

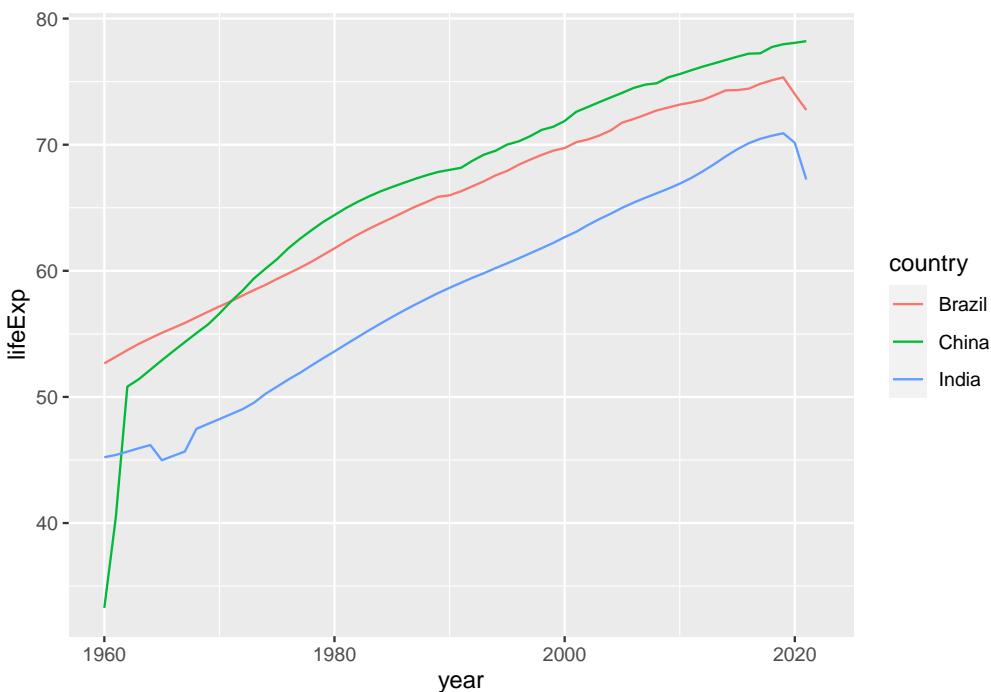
```
#> [212] "Sierra Leone"
#> [213] "Singapore"
#> [214] "Sint Maarten (Dutch part)"
#> [215] "Slovak Republic"
#> [216] "Slovenia"
#> [217] "Small states"
#> [218] "Solomon Islands"
#> [219] "Somalia"
#> [220] "South Africa"
#> [221] "South Asia"
#> [222] "South Asia (IDA & IBRD)"
#> [223] "South Sudan"
#> [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。上の国のリストで見てかもしれませんね。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。幾の大陸があり、それぞれの大陸のいくつ国データがこのデータには入っているでしょうか。

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_lifeExp = max(lifeExp), min_lifeExp = min(lifeExp))
#> # A tibble: 7 x 5
#> # Groups:   income [7]
#>   income mean_lifeExp median_lifeExp max_lifeExp min_lifeExp
#>   <chr>     <dbl>        <dbl>       <dbl>        <dbl>
#> 1 Aggregat  67.6         69.2       80.1        53.3
#> 2 High income  NA          NA          NA          NA
#> 3 Low income  56.9         55.4       73.7        47.4
#> 4 Lower income  64.4         65.9       77.1        43.1
#> 5 Not categorized  73.1         73.1       73.1        73.1
#> 6 Upper income  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 練習

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

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

18.7 まとめ

18.7.1 関数

18.7.1.1 base

- colnames()
- range()

18.7.1.2 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 tidyverse  1.3.0
#> v purrrr     1.0.2
#> -- 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 cor
library(showtext)
#> Loading required package: sysfonts
#> Loading required package: showtextdb
showtext_auto()

```

19.1.2.2 復習

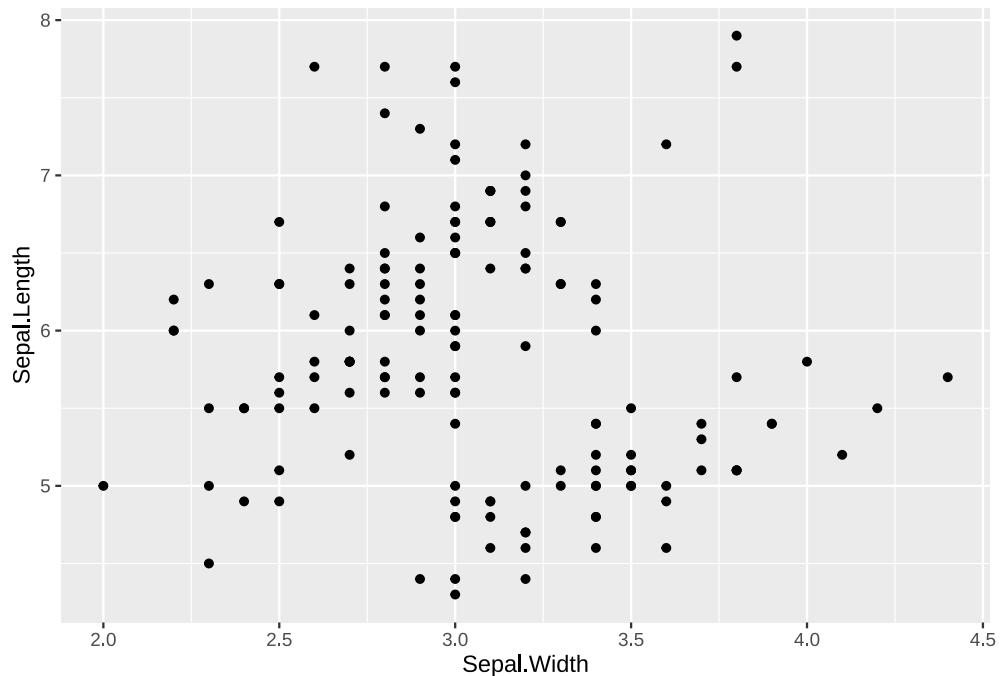
Tidyverse の章で、紹介した例の復習から始めましょう。

```

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()
```

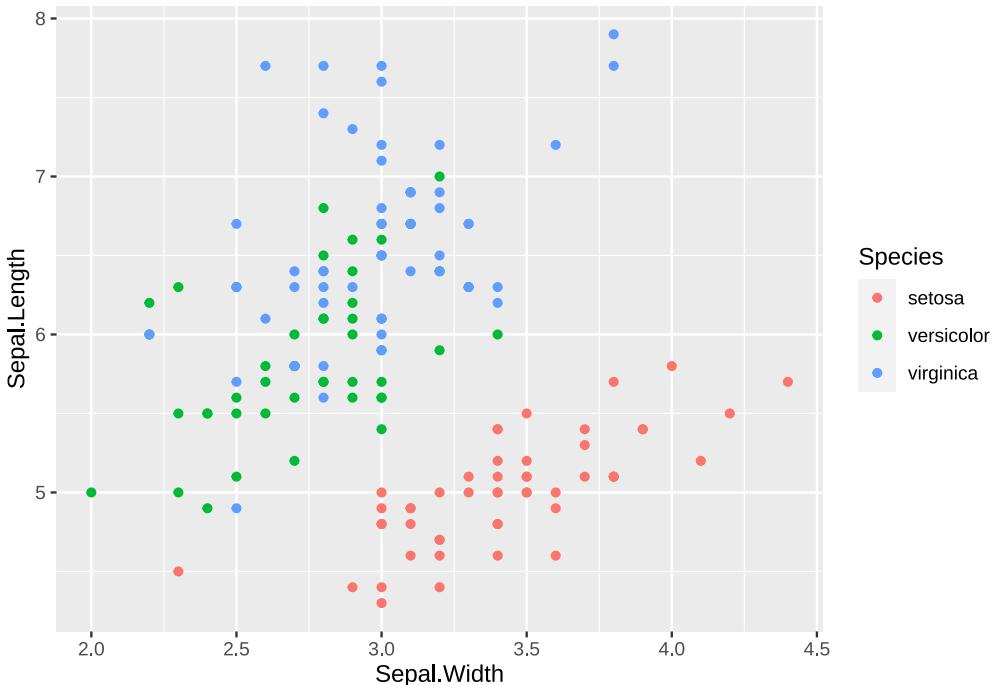
や、さらに、詳しく

```
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。

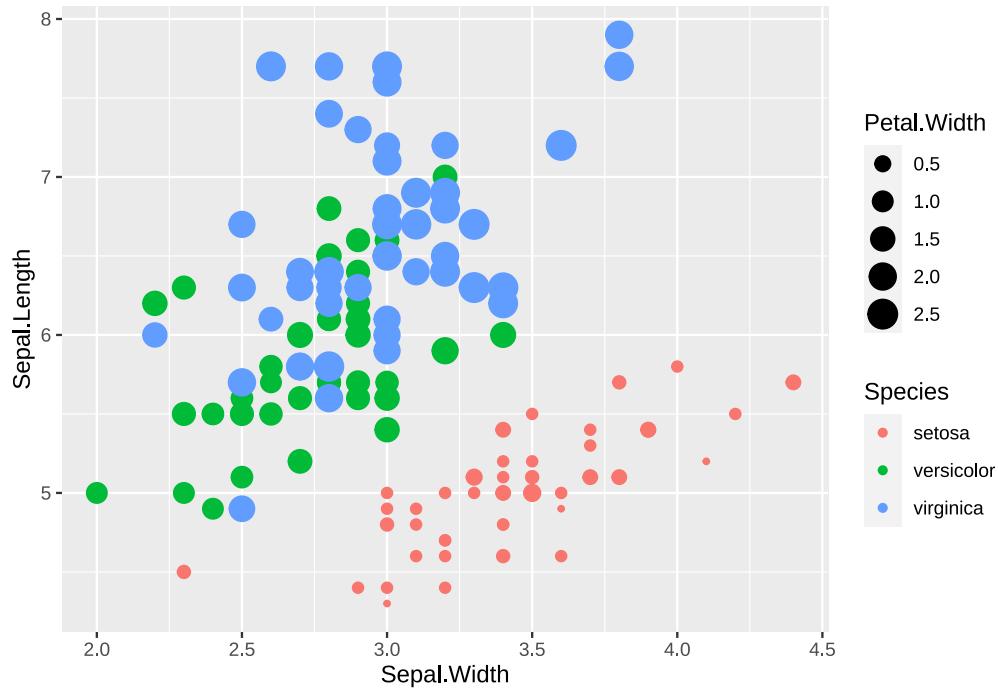
種類 (Species) ごとに色を変える場合には、color = Species とします。

```
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)

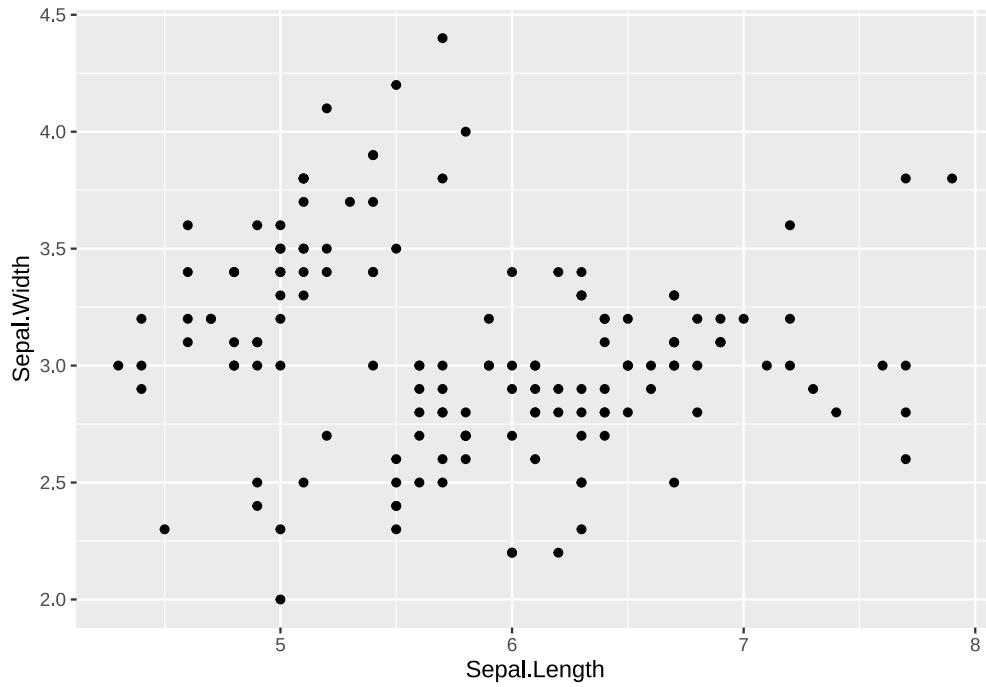
散布図は、データの二つの変数（列）を `x` と `y` に対応させる、最も基本的なグラフです。最初に試すべきグラフだともいいうことができます。`mapping =` は省略することができます。

```
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。

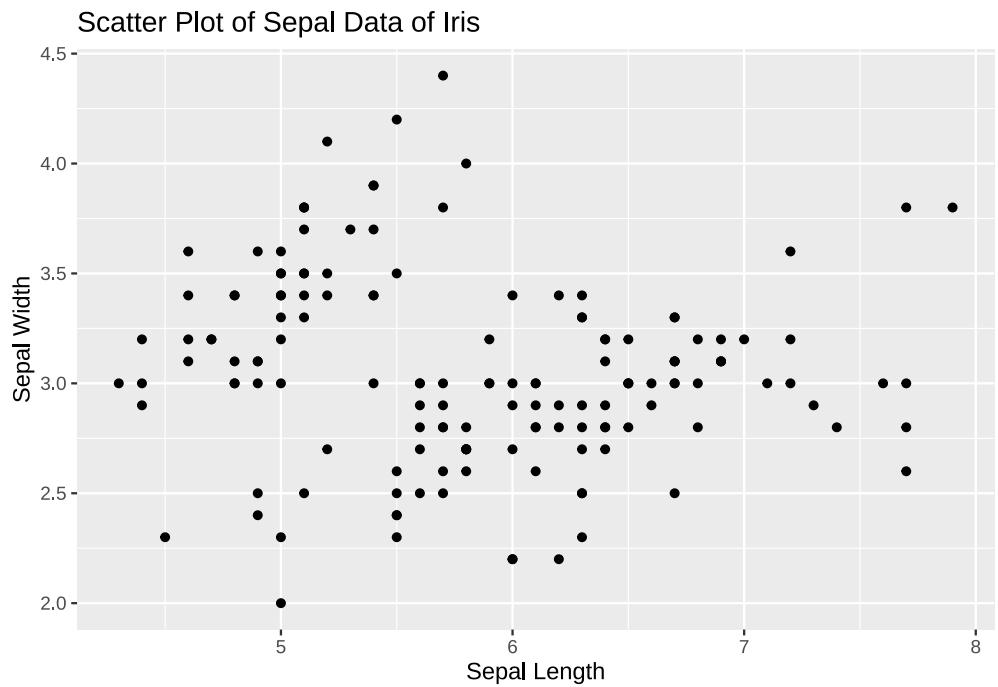
```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width)) +
  geom_point()
```



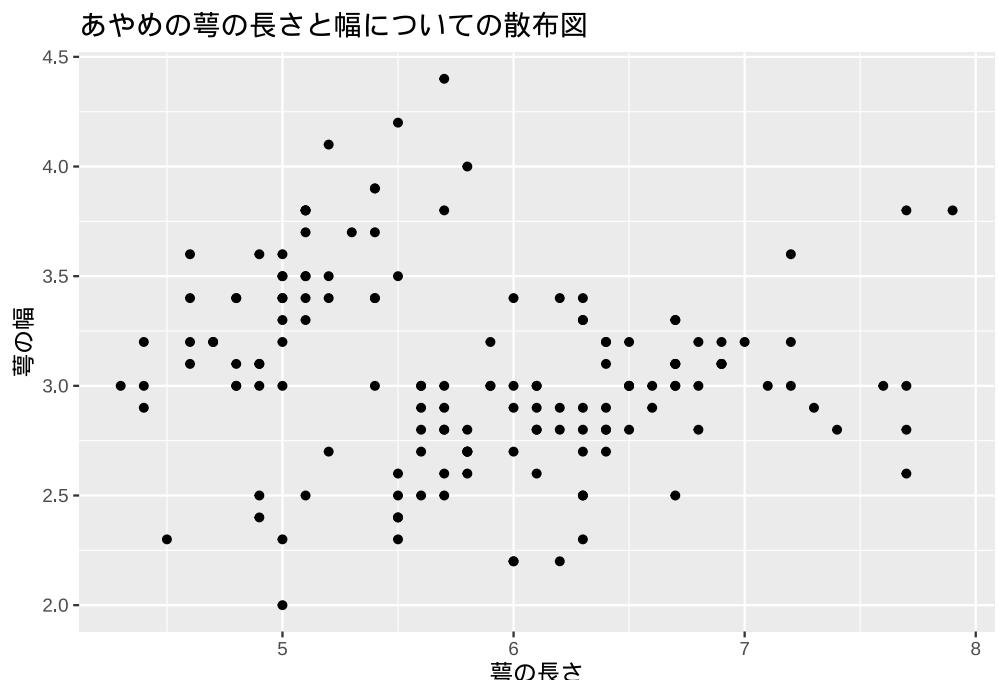
19.2.1 ラベル Labels

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")
```

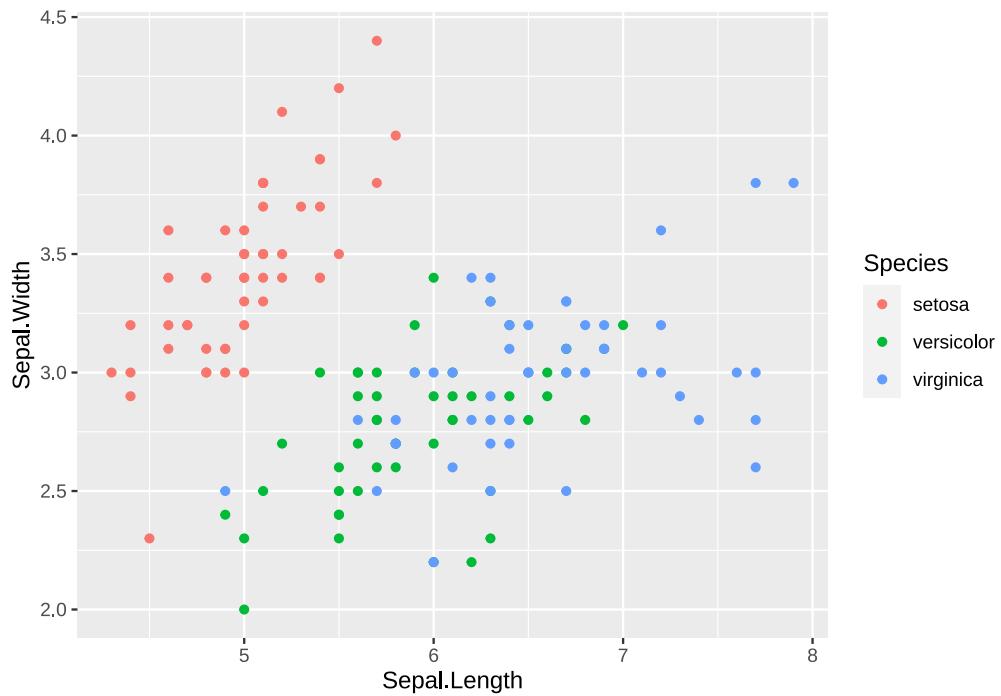


このように、日本語をタイトルや、ラベルに使うことも可能ですが、以後は、データに日本語が含まれていない場合には、そのまま表示します。

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。

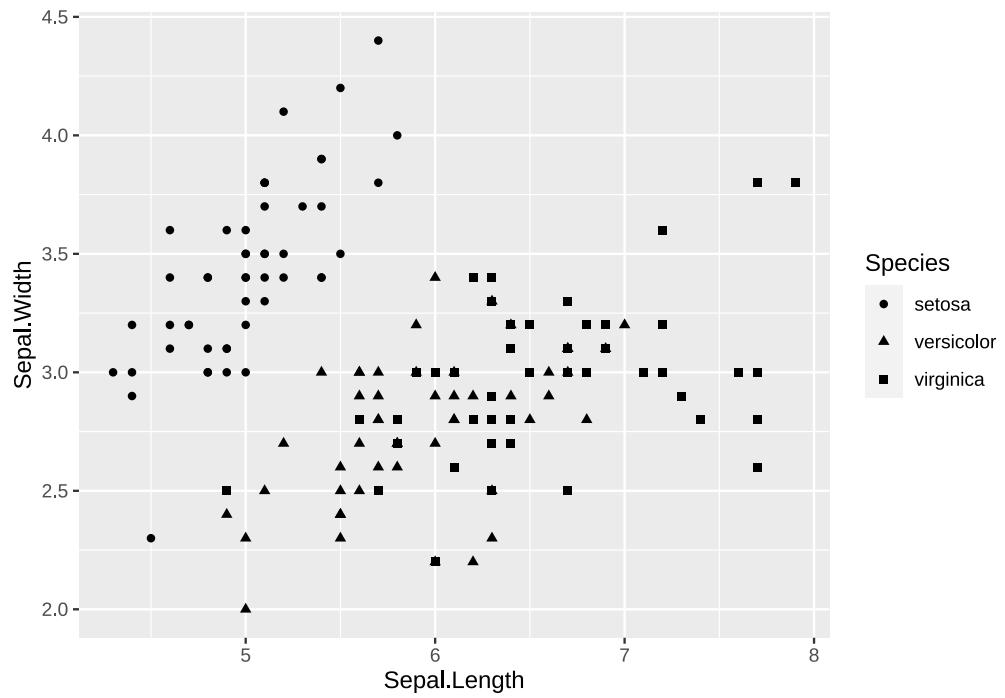
```
ggplot(data = df_iris, aes(x = Sepal.Length, y = Sepal.Width, color = Species)) +  
  geom_point()
```



19.2.3 形状 Shapes

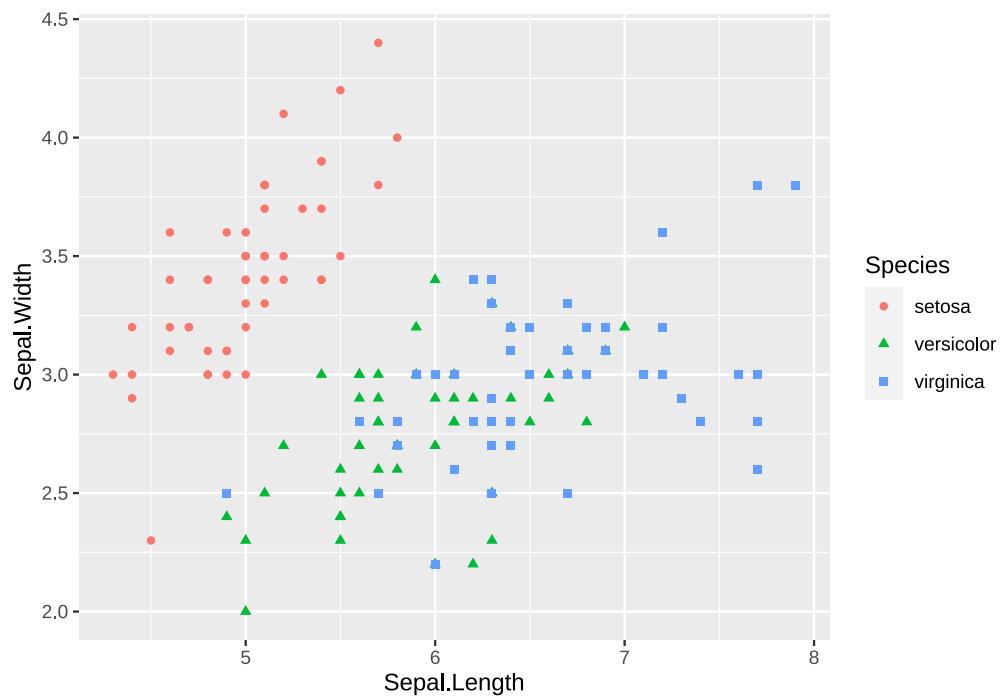
色ではなく、形で Species を区別することも可能です。

```
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ます。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>

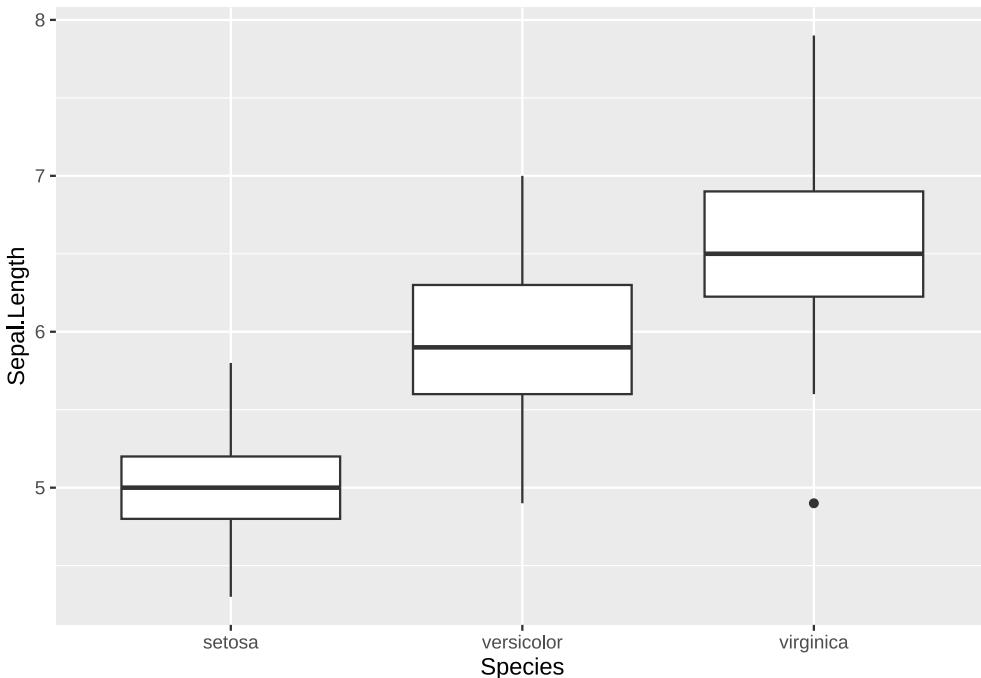
日本語では、下のサイトの箱ひげ図の項目を見ていただければと思います。外れ値についての説明もあります。

箱ひげ図と幹葉表示：<https://bellcurve.jp/statistics/course/5219.html>

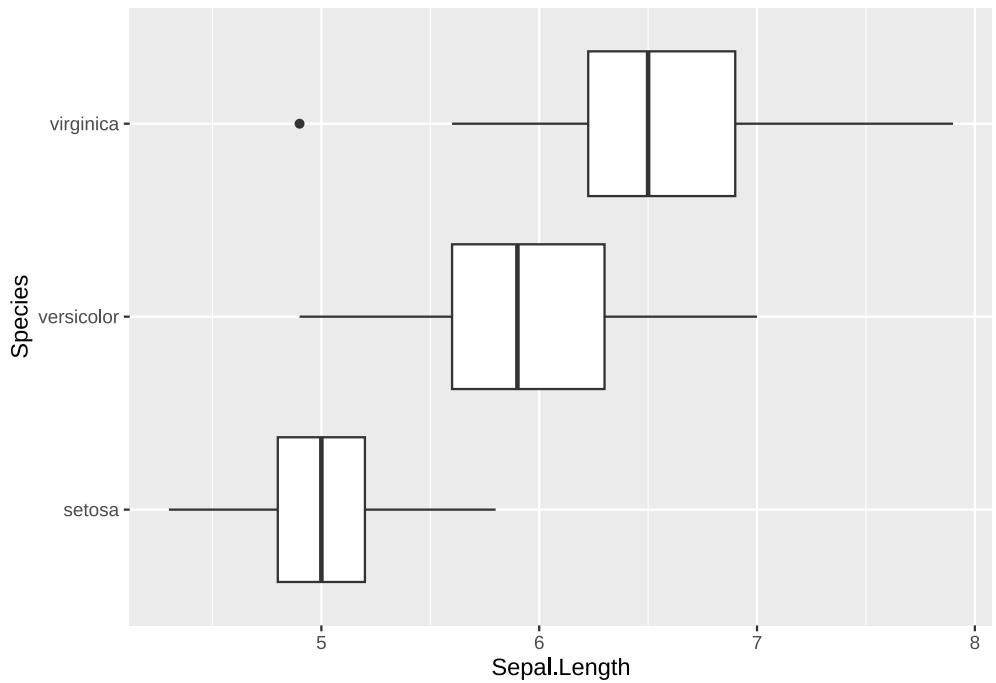
Transcript ボタンから、スクリプトを表示することもできます。

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ができますが、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()
```



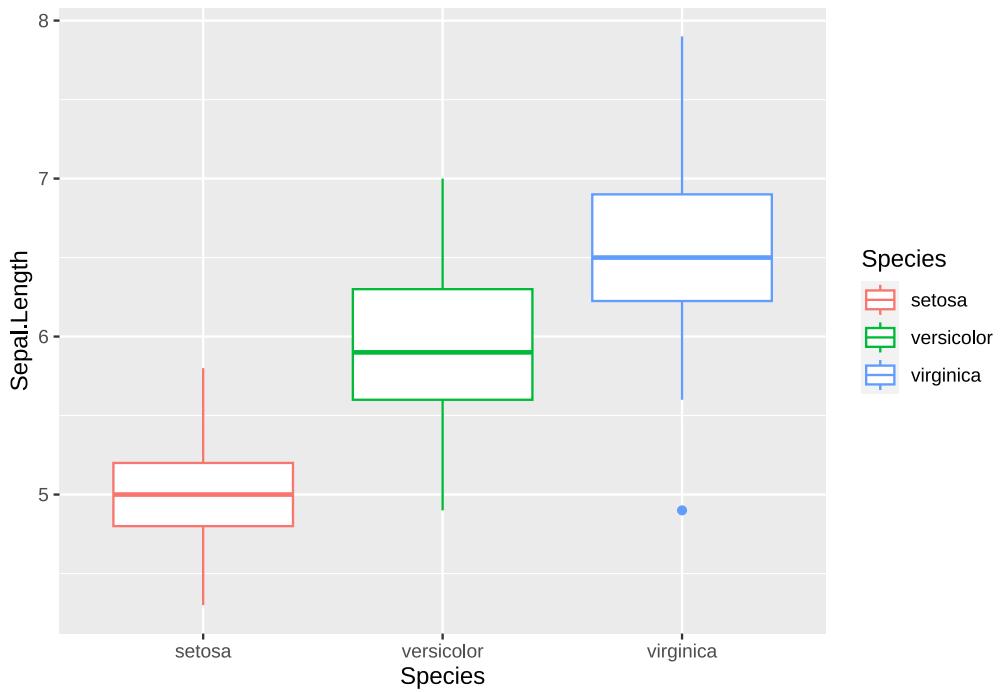
```
ggplot(data = df_iris, aes(y = Species, x = Sepal.Length)) +  
  geom_boxplot()
```



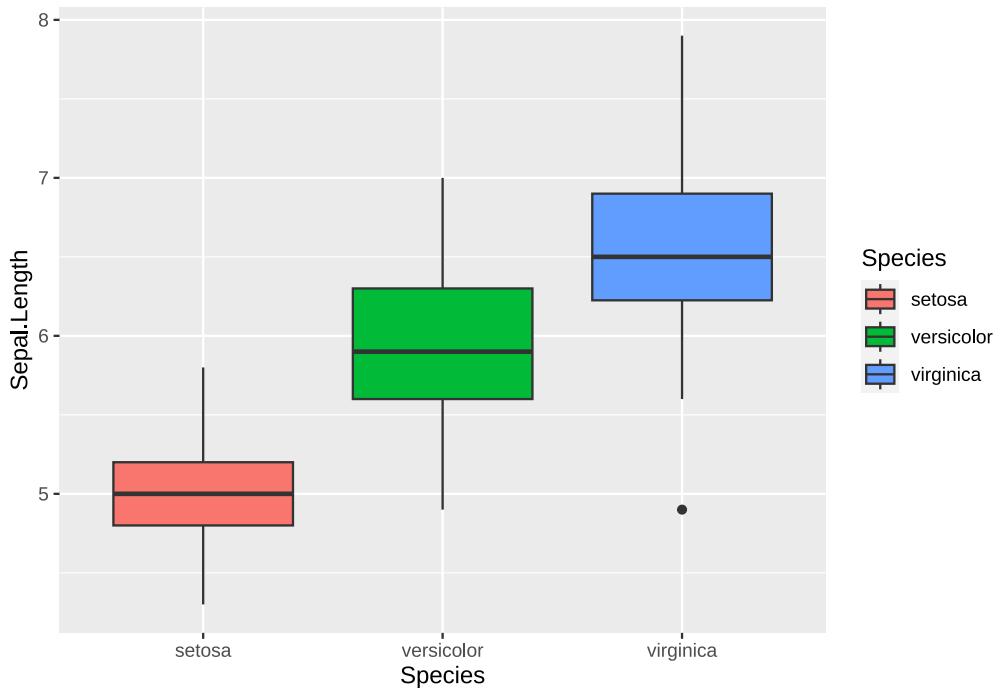
各、種類 (Species) ごとに、Sepal.Width (萼 (がく) 幅) が、どのように分布しているかを示しています。真ん中の太い線が、中央値 (median)、箱が、第一四分位 (Q1) から第三四分位 (Q3)、線と点で表される外れ値も、どのような基準か定められています。(IQR = Q3-Q1, 線は、 $Q3+1.5 \times IQR$ 以下に入っている実際の値までと、 $Q1-1.5 \times IQR$ 以上に入っている実際の値まで。それらに入っていないものが外れ値)。

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

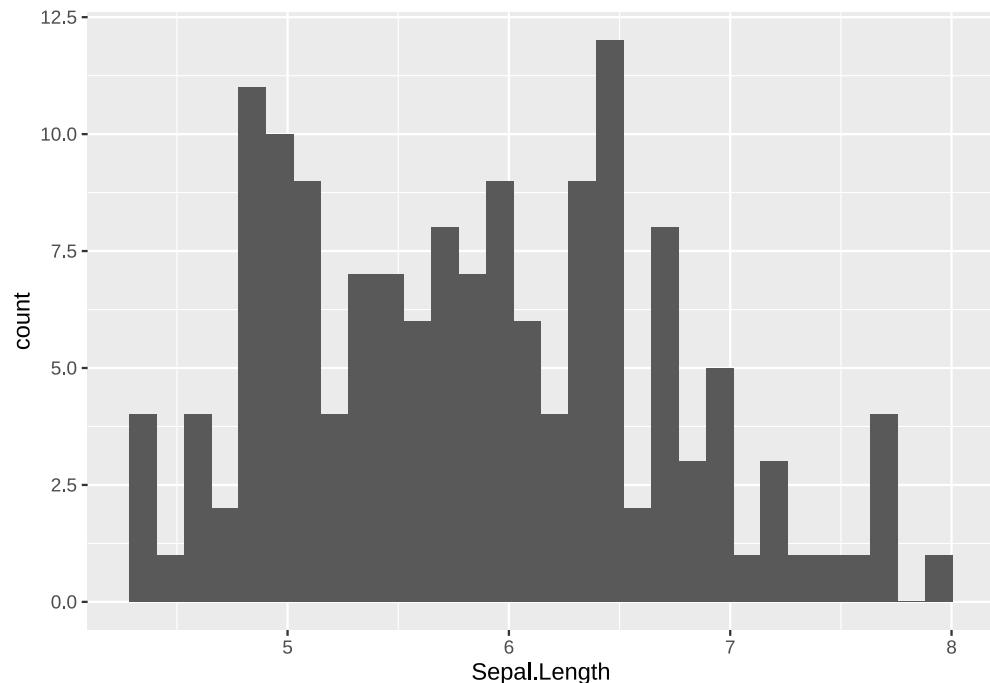
Visualize the distribution of a single continuous variable by dividing the x axis into bins and counting the number of observations in each bin. Histograms (geom_histogram()) display the counts with bars; frequency polygons

(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 bin に分割し、各 bin 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`.
```

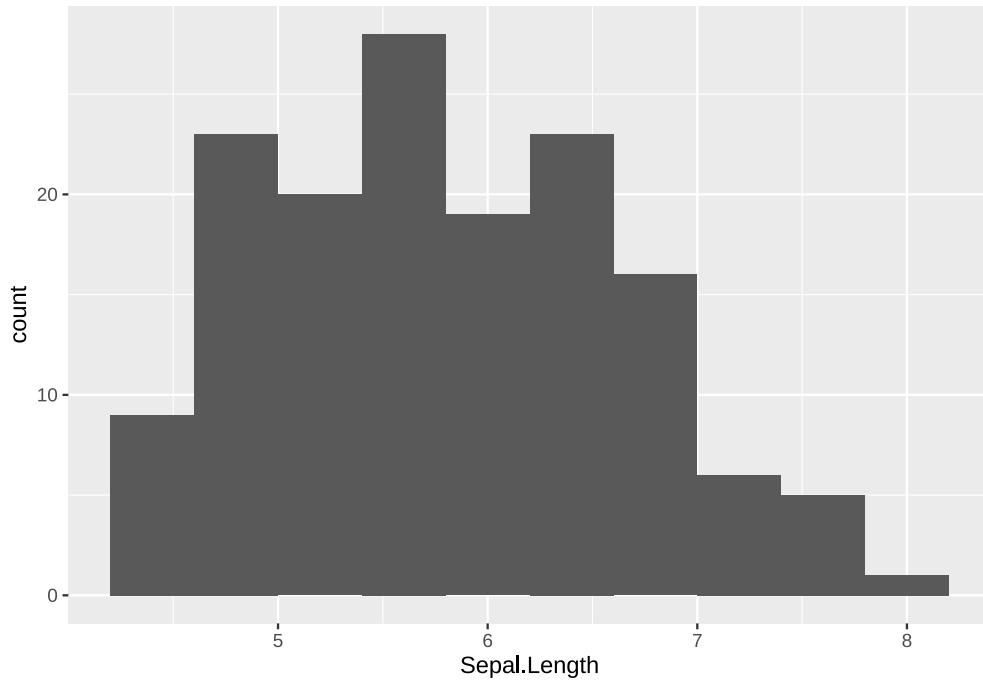


枠 (bins) を幾つに分けるか、または枠の幅を指定するようにとのメッセージが表示されます。

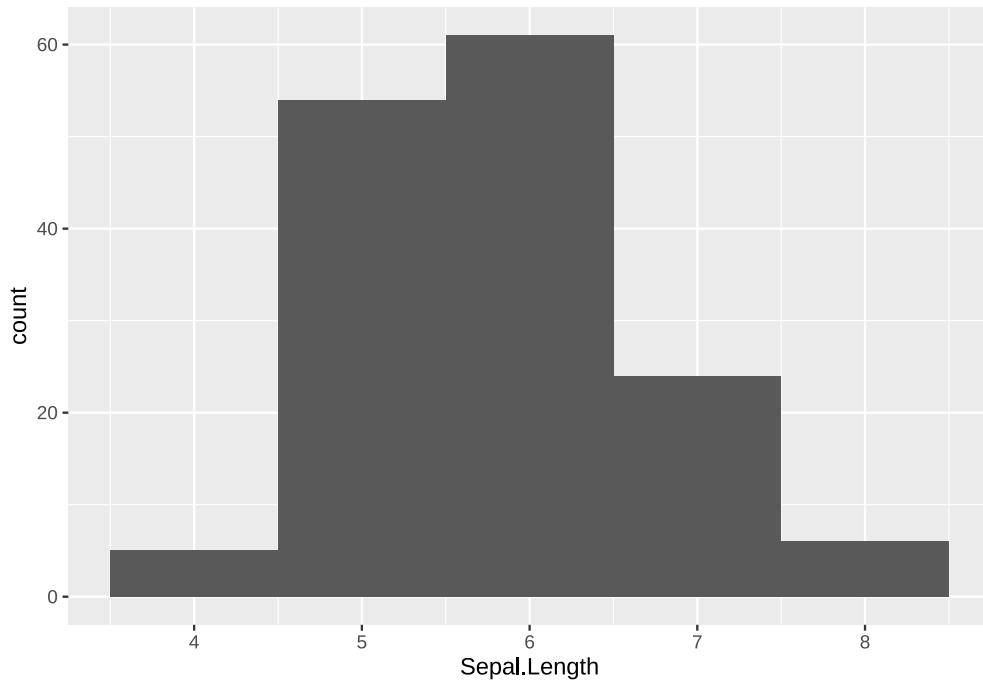
枠 (bins) の数を変更するには `bins = <number>` を使用します。幅を指定するときは、`binwidth = <number>` とします。

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 しています。

```
ggplot(data = df_iris, aes(x = Sepal.Length)) +
  geom_histogram(bins = 10)
```



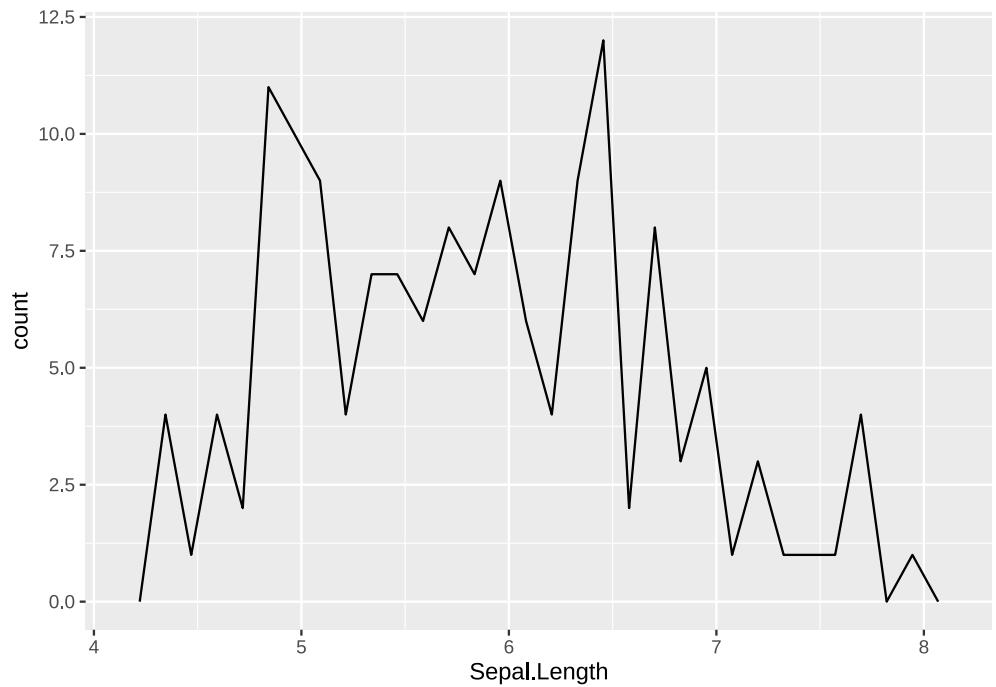
```
ggplot(data = df_iris, aes(x = Sepal.Length)) +  
  geom_histogram(binwidth = 1)
```



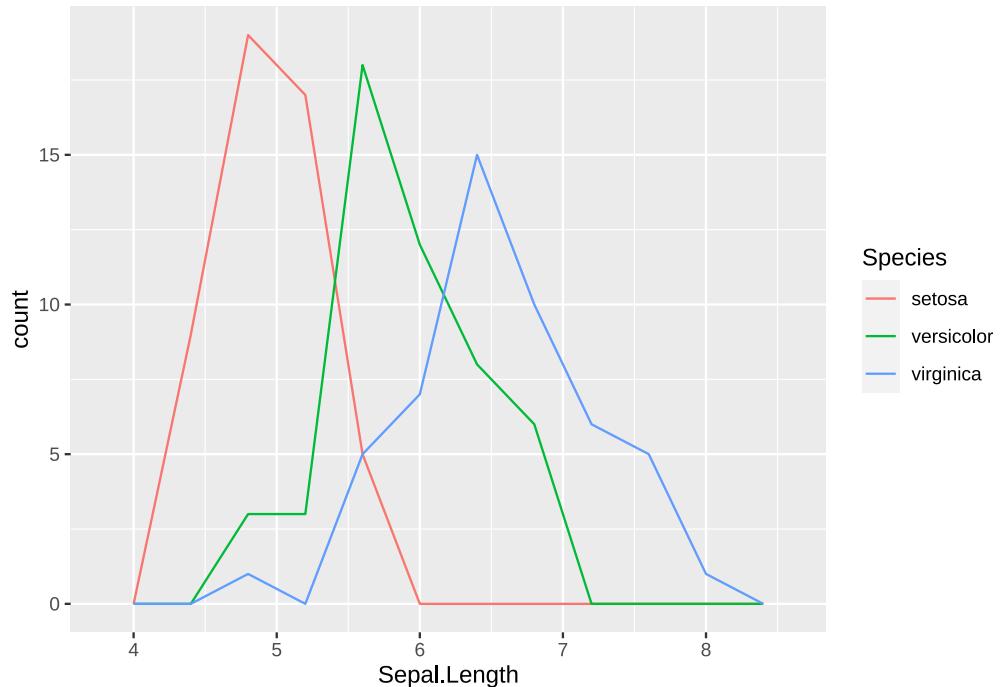
頻度多角形 (geom_freqpoly()) を使うと以下のようになります。Species ごとに比べたり、色をつけたりもできます。

```
ggplot(data = df_iris, aes(x = Sepal.Length)) +  
  geom_freqpoly()  
#> `stat_bin()` using `bins = 30`. Pick better value with
```

```
#> `binwidth`.
```

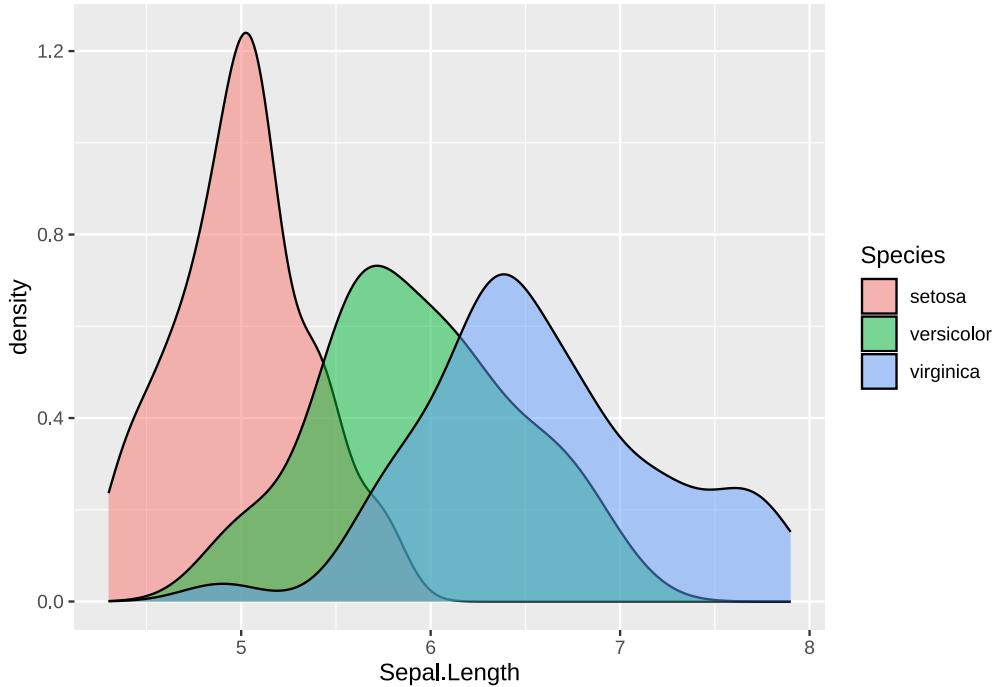


```
ggplot(data = df_iris, aes(x = Sepal.Length, color = Species)) +
  geom_freqpoly(bins = 10)
```



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。

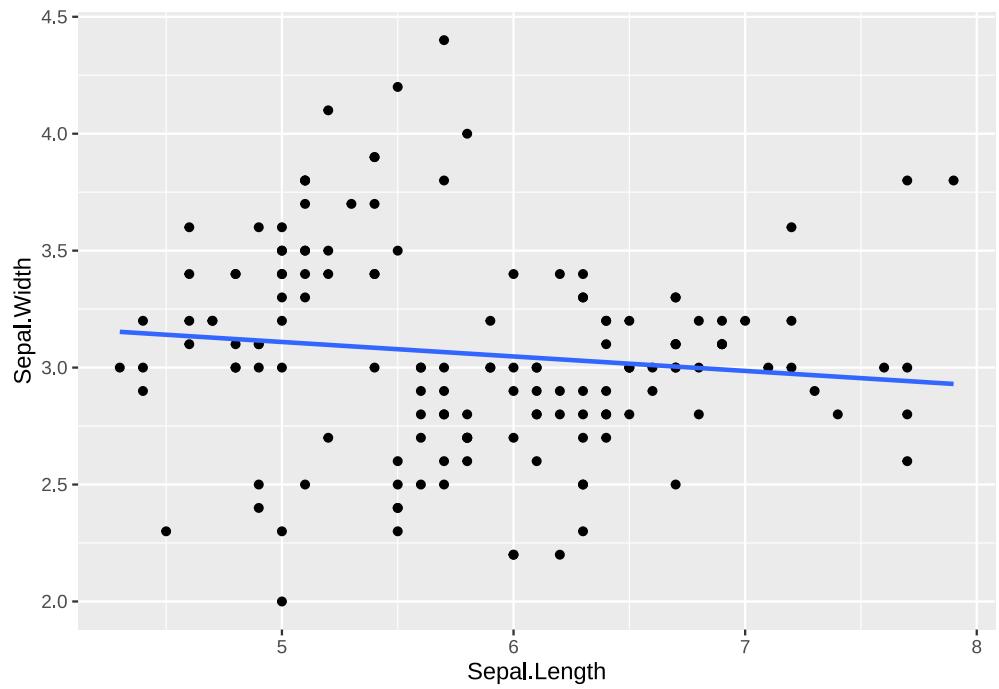
```
ggplot(data = df_iris, aes(x = Sepal.Length, fill = Species)) +  
  geom_density(alpha = 0.5)
```



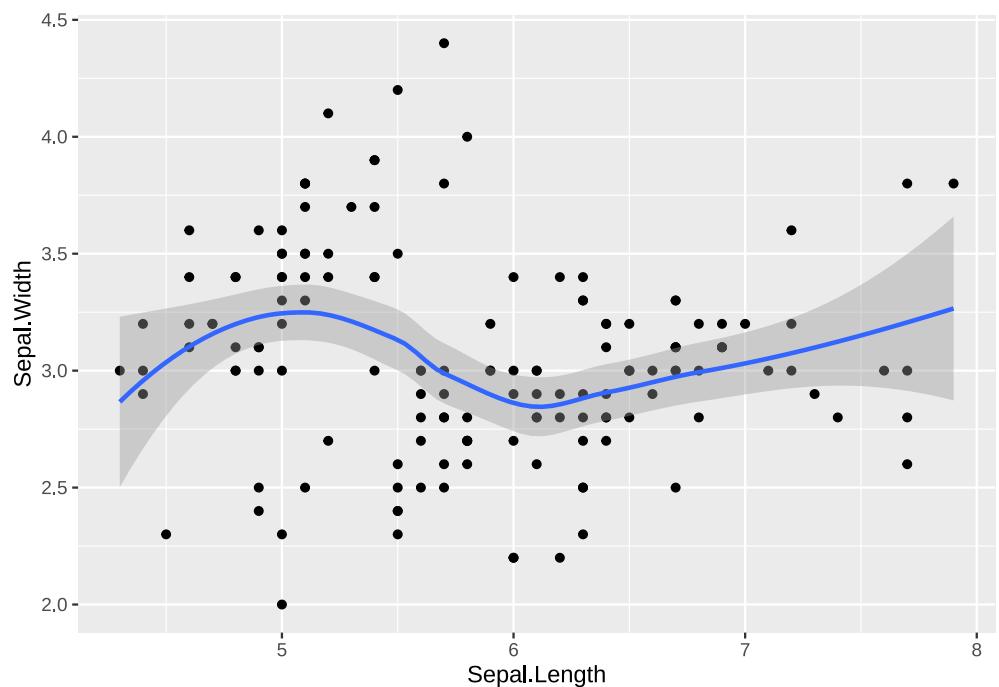
19.3.2 線形モデル Data Modeling

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が。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。

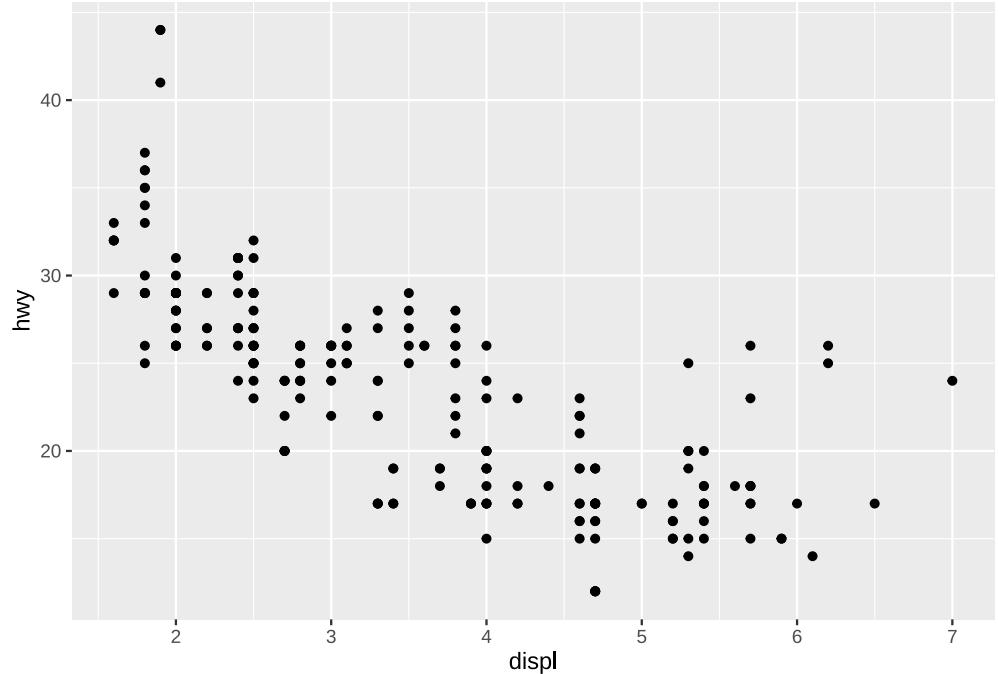
manufacturer は、メーカー名、model は、型式、disp は、排気量（単位：リットル）、year は年式、cyl は、気筒数、trans は、オートマかマニュアルか、drv は、f が前輪駆動、r は後輪駆動、4 は四輪駆動、cty は街中の燃費（1 ガロンで何マイル走るか）、hwy は高速道路での燃費（1 ガロンで何マイル走るか）、fl 燃料の種類、class タイプ

```
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>
```

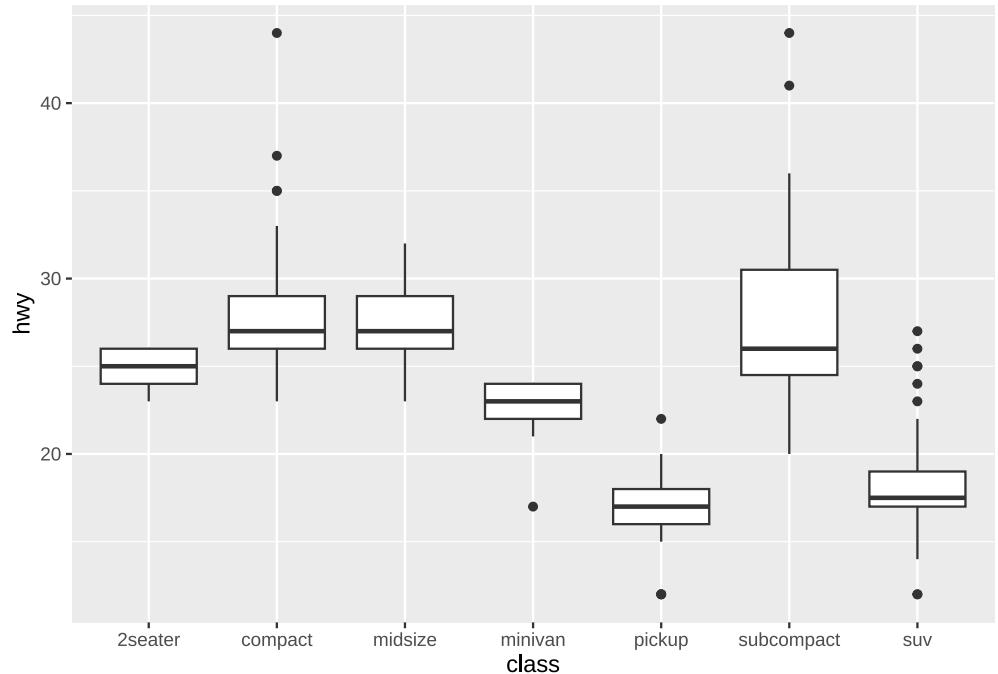
```
glimpse(df_mpg)
#> Rows: 234
#> Columns: 11
#> $ manufacturer <chr> "audi", "audi", "audi", "audi", "audi~"
#> $ model       <chr> "a4", "a4", "a4", "a4", "a4", "~"
#> $ displ       <dbl> 1.8, 1.8, 2.0, 2.0, 2.8, 2.8, 3.1, 1.~
#> $ year        <int> 1999, 1999, 2008, 2008, 1999, 1999, 2-
#> $ cyl          <int> 4, 4, 4, 4, 6, 6, 4, 4, 4, 4, 6, 6-
#> $ trans        <chr> "auto(l5)", "manual(m5)", "manual(m6)~"
#> $ drv          <chr> "f", "f", "f", "f", "f", "f", "f", "4-
#> $ cty          <int> 18, 21, 20, 21, 16, 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~  
#> $ 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、`dspl` (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
)
```

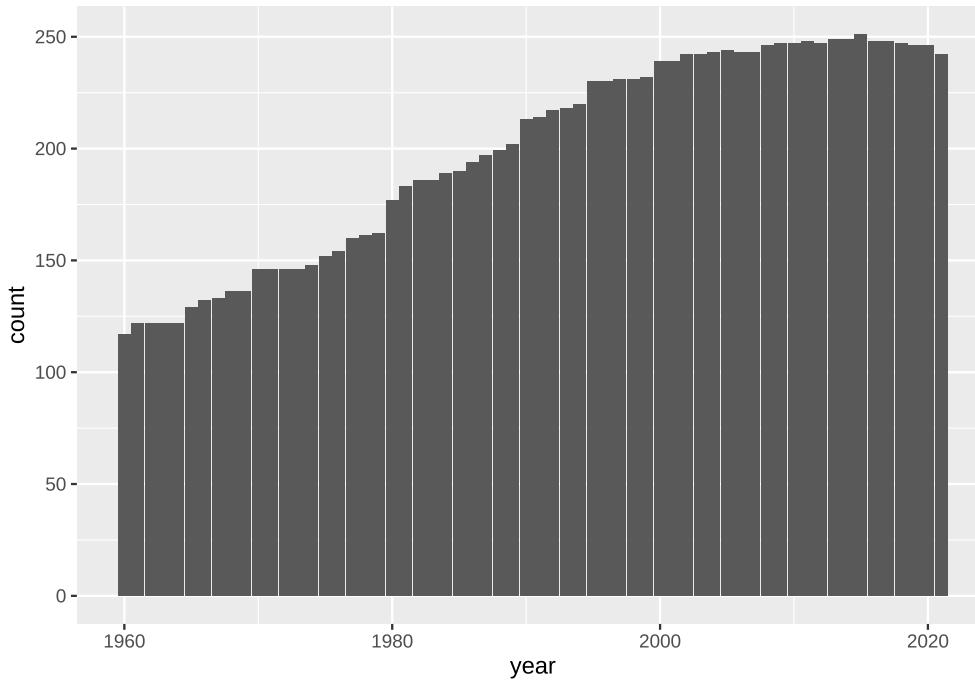
すこし、追加情報を付加したものも取得しておきます。

```
#> Rows: 16758 Columns: 15
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (6): year, lifeExp, pop, gdpPercap, longitude, 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
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))
```



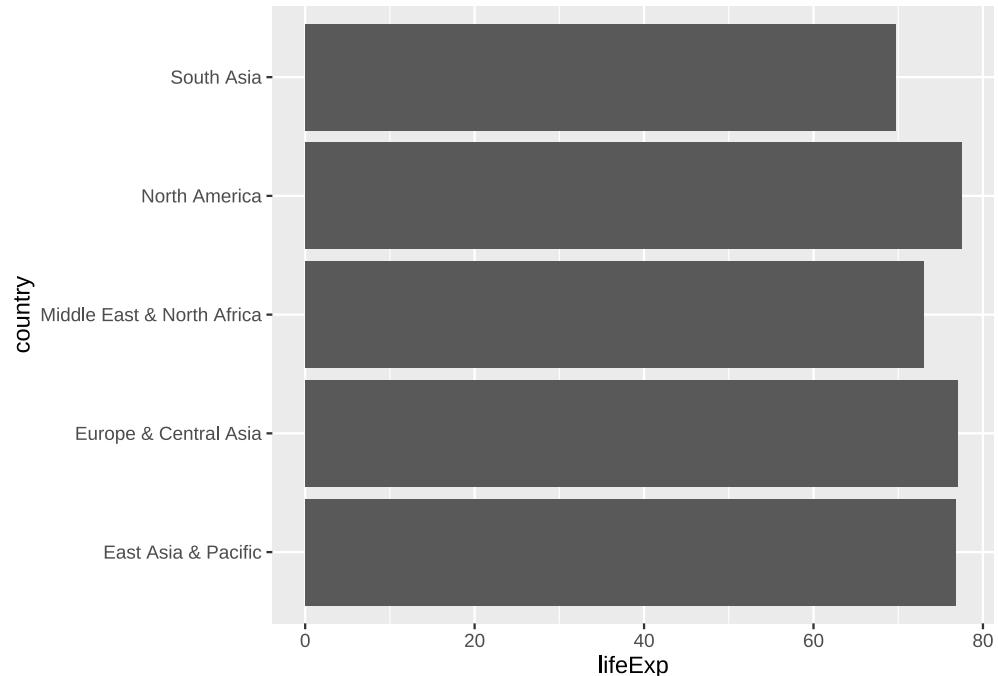
最初に、三つの indicator がどれもが、NA でない行だけを選択していますから、三つの指標がすべて値がある、データを年ごとに数えてグラフにしたものが、上のものになっています。表にすると次のようになっています。

```
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
```

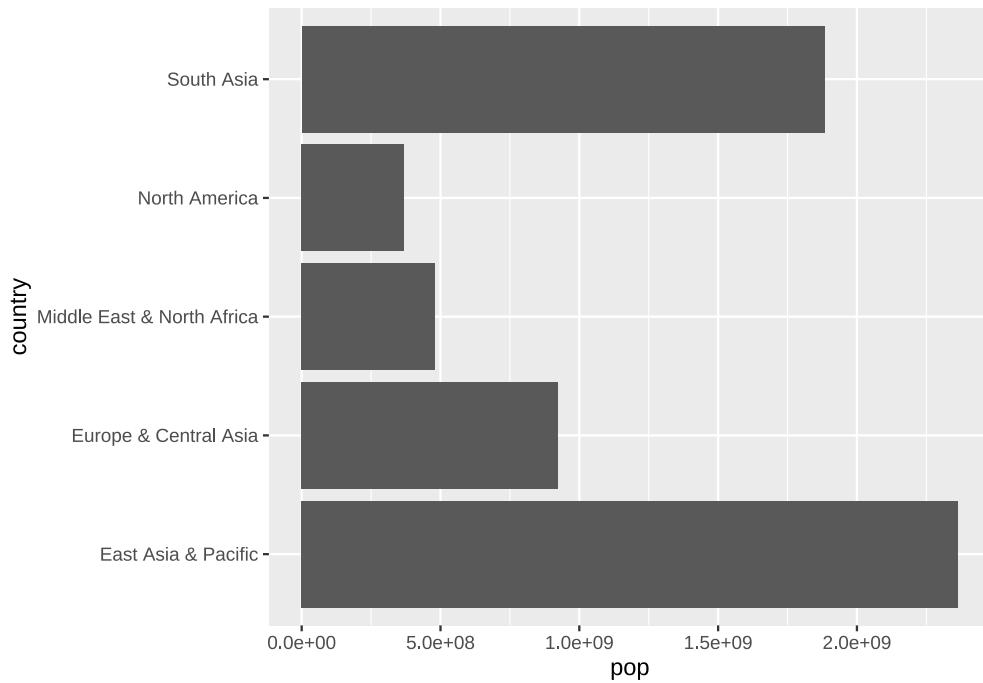
同じような棒グラフですが、それぞれの値をグラフにする場合には、`geom_col()` を使います。

```
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"

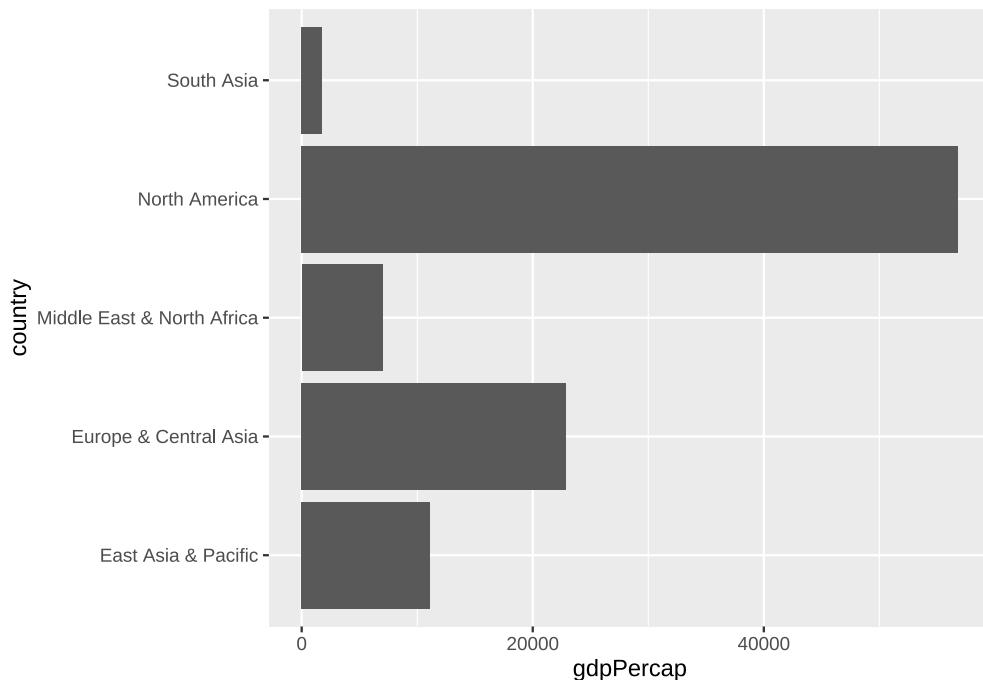
df_wdi_extra |> filter(income == "Aggregates", year == 2020) |>
  filter(country %in% regions) |>
  ggplot(aes(y = country, x = lifeExp)) + geom_col()
```



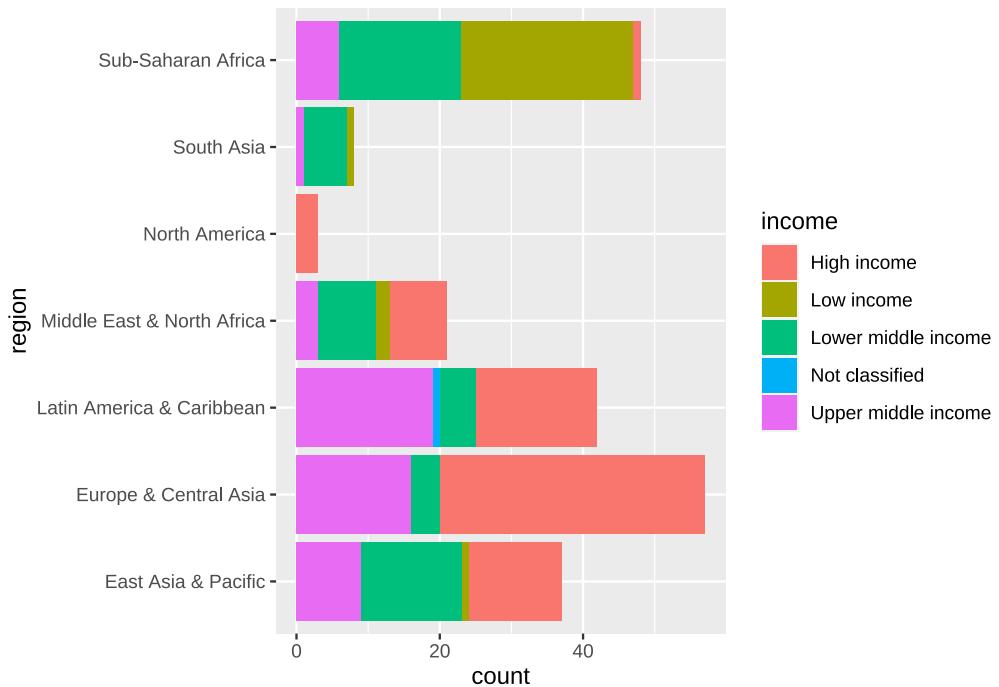
```
df_wdi_extra |> filter(income == "Aggregates", year == 2020) |>
  filter(country %in% regions) |>
  ggplot(aes(y = country, x = pop)) + geom_col()
```



```
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 棒グラフの応用

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

地域のリストを `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、地域ごとにグループ化します。

まず、地域ごとにまとめて、国がいくつあるかは次のように表にすることもできます。また、収入の多寡のグループにも分けたものをその下に加えておきます。

これらを、棒グラフで表示しています。

```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |> summarize(n = n())
#> # A tibble: 7 x 2
#>   region           n
#>   <chr>          <int>
#> 1 East Asia & Pacific      37
#> 2 Europe & Central Asia     57
#> 3 Latin America & Caribbean 42
#> 4 Middle East & North Africa 21
#> 5 North America              3
#> 6 South Asia                 8
```

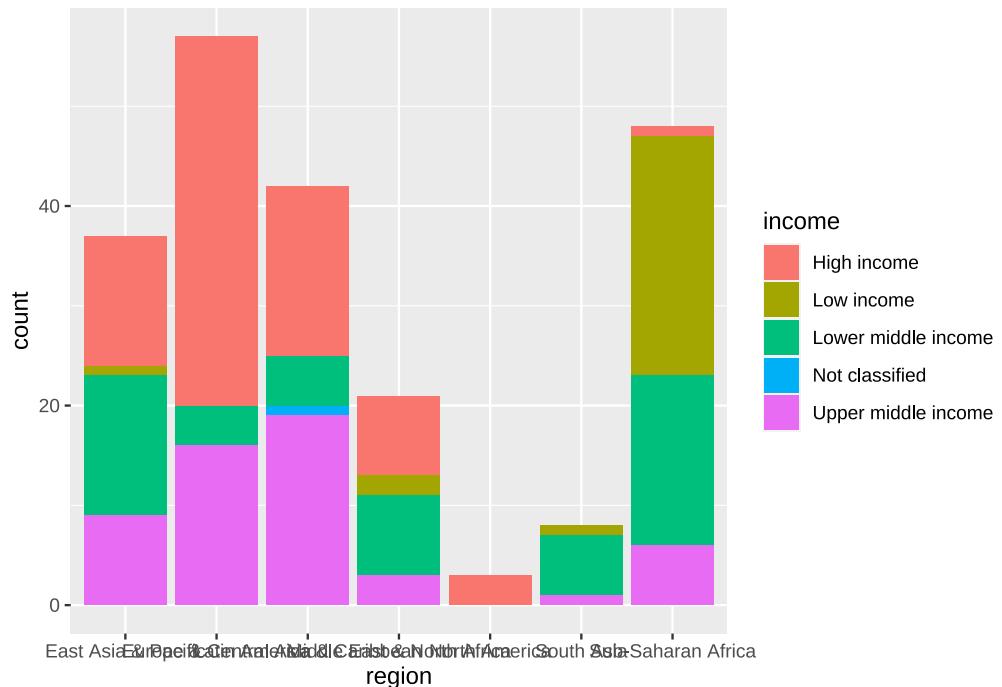
```
#> 7 Sub-Saharan Africa          48
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region, income) |> summarize(n = n())
#> `summarise()` has grouped output by 'region'. You can
#> override using the `~.groups` argument.
#> # A tibble: 23 x 3
#> # Groups:   region [7]
#>   region           income       n
#>   <chr>            <chr>     <int>
#> 1 East Asia & Pacific  High income    13
#> 2 East Asia & Pacific  Low income      1
#> 3 East Asia & Pacific Lower middle income 14
#> 4 East Asia & Pacific Upper middle income   9
#> 5 Europe & Central Asia High income    37
#> 6 Europe & Central Asia Lower middle income   4
#> 7 Europe & Central Asia Upper middle income  16
#> 8 Latin America & Caribbean High income    17
#> 9 Latin America & Caribbean Lower middle income   5
#> 10 Latin America & Caribbean Not classified    1
#> # i 13 more rows
```

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。

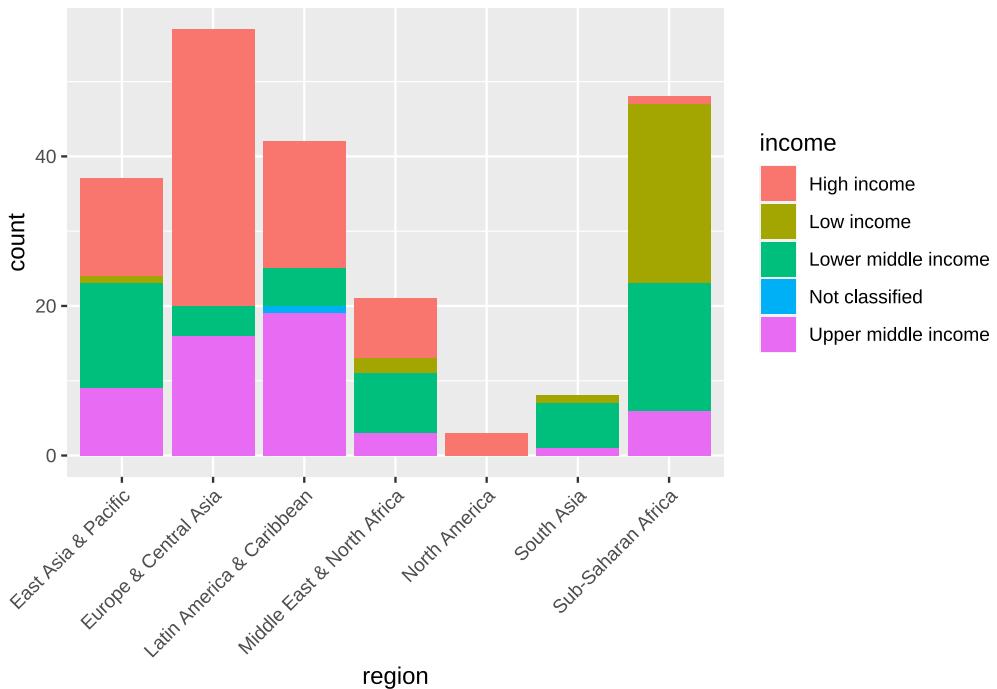
```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar()
```



19.5.3.3 地域名

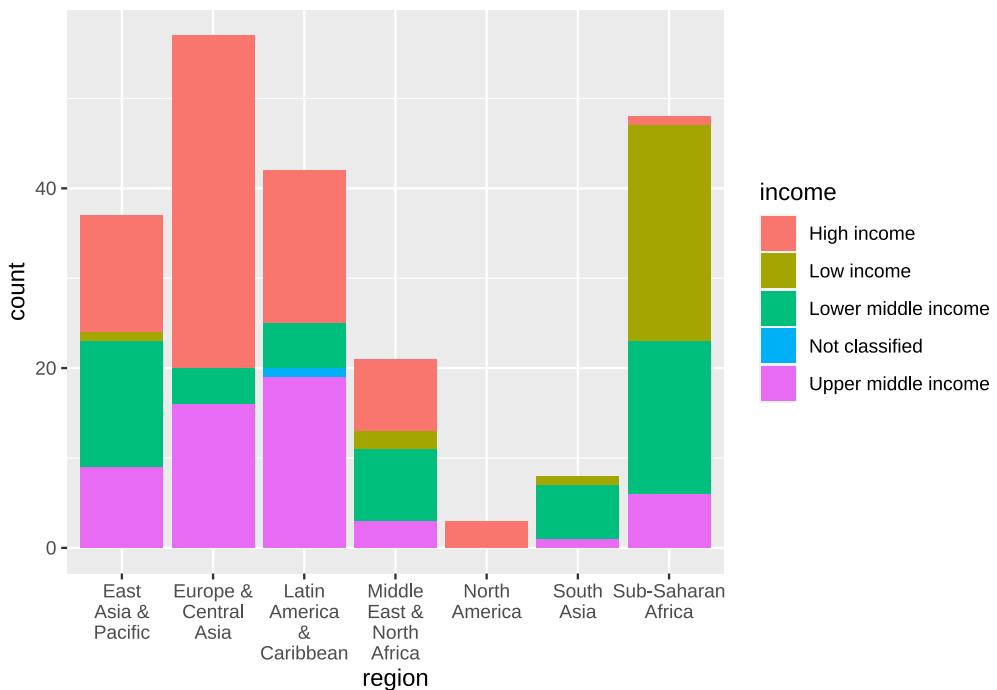
なぜ、横にしたか、見当がついたかと思います。縦にすると、地域名が長くて重なってしまうのですね。もちろん、それを、改善する手がいくつかあります。1つ目は、地域名を縦にしたり、角度を付ける方法。`theme` を使い、中に角度を書きます。実は、それだけだと、ちょっと重なってしまうので、`hjust` の変数で調節しています。垂直方向の調節は、`vjust` です。数はいろいろと変更してみてください。

```
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))
```



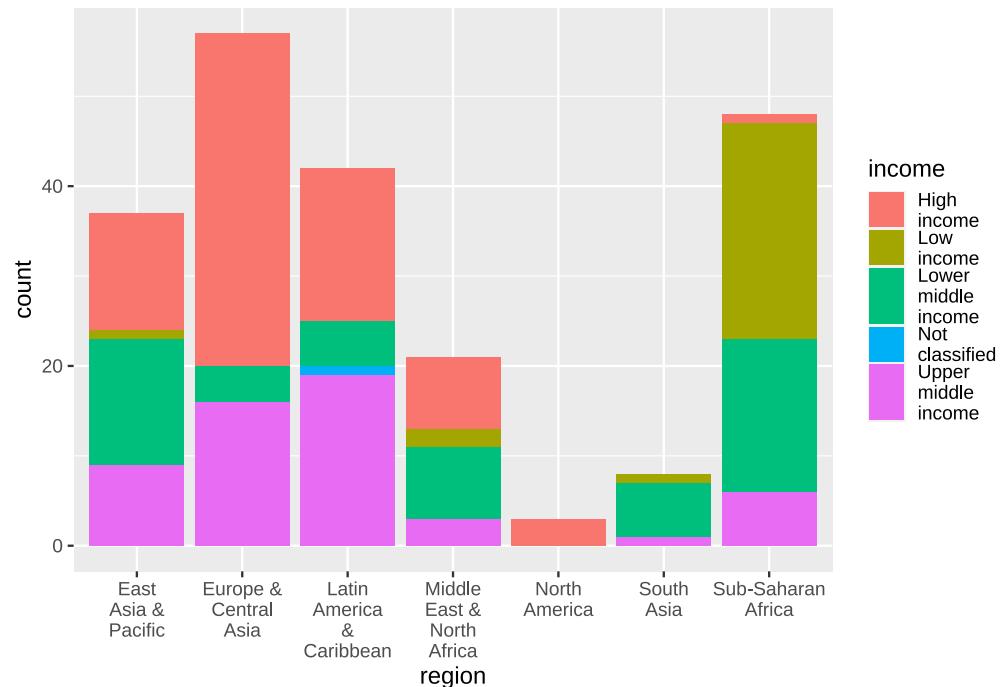
もう一つは折り返す方法です。こちらの方が、良いかもしれません。

```
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 のあとに数字は、適切な数を選んでください。

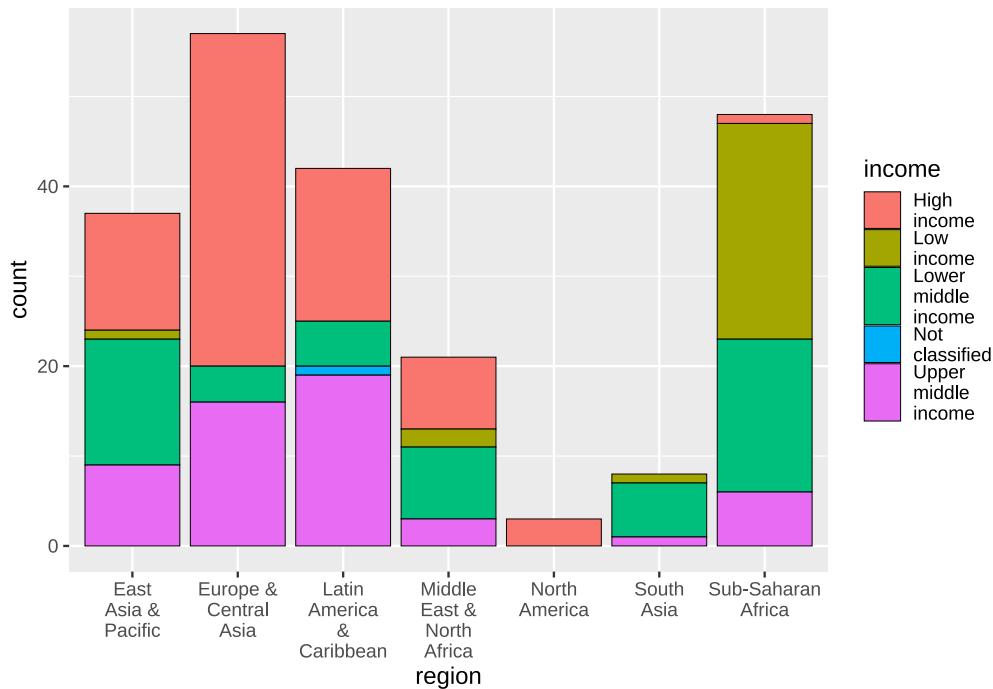
```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar() +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_fill_discrete(labels = function(x) str_wrap(x, width = 8))
```



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。

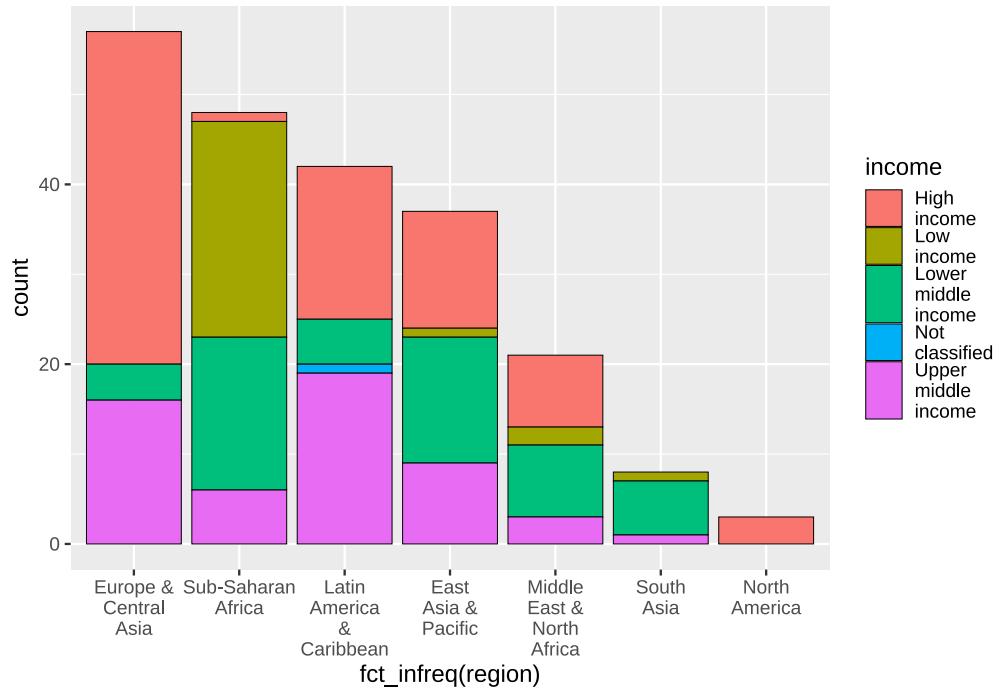
```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar(color = "black", linewidth =
  0.2) +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_fill_discrete(labels = function(x) str_wrap(x, 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。

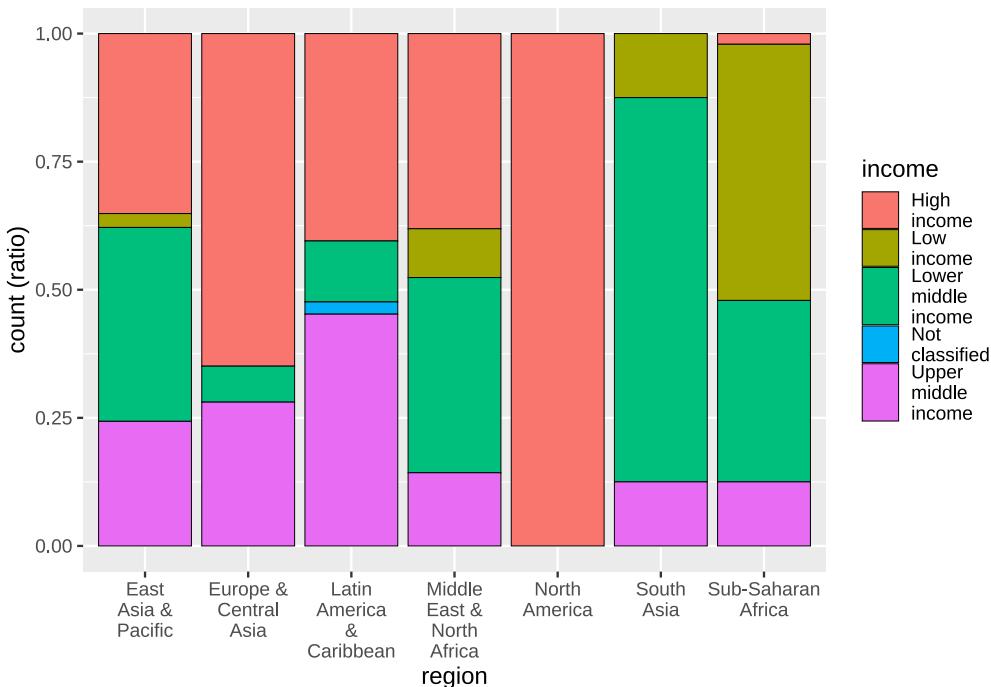
```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = fct_infreq(region), fill = income)) +
  geom_bar(color = "black", linewidth = 0.2) +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_fill_discrete(labels = function(x) 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 =
  scale_x_discrete(labels = function(x) str_wrap(x, width = 8)) +
  scale_fill_discrete(labels = function(x) str_wrap(x, width = 8)) +
  labs(y = "count (ratio)")
```



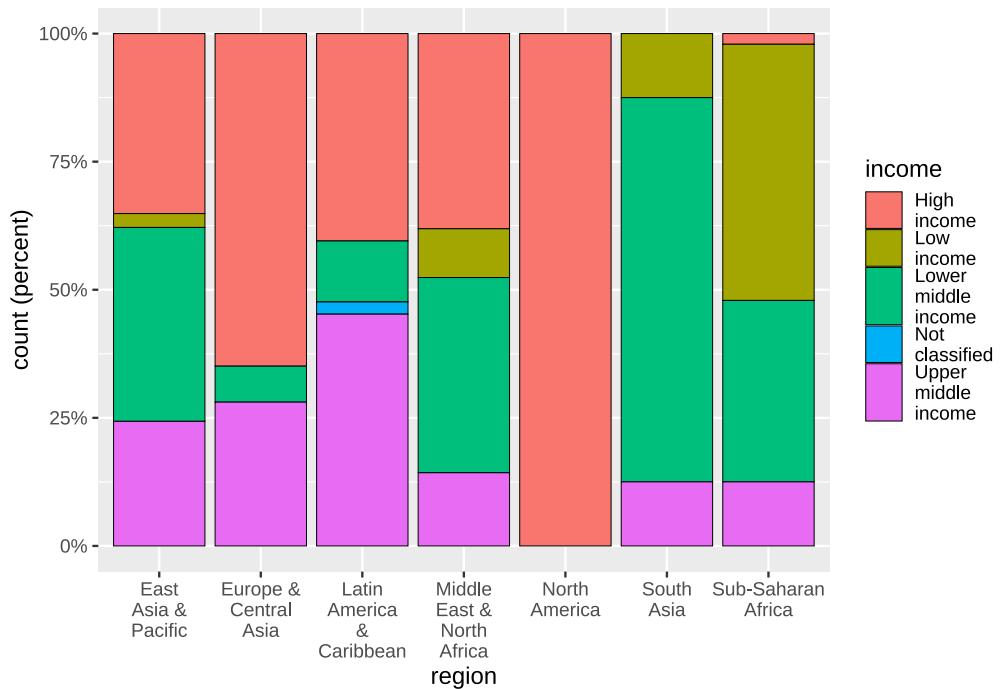
19.5.3.7 縦軸を百分率に

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)
```

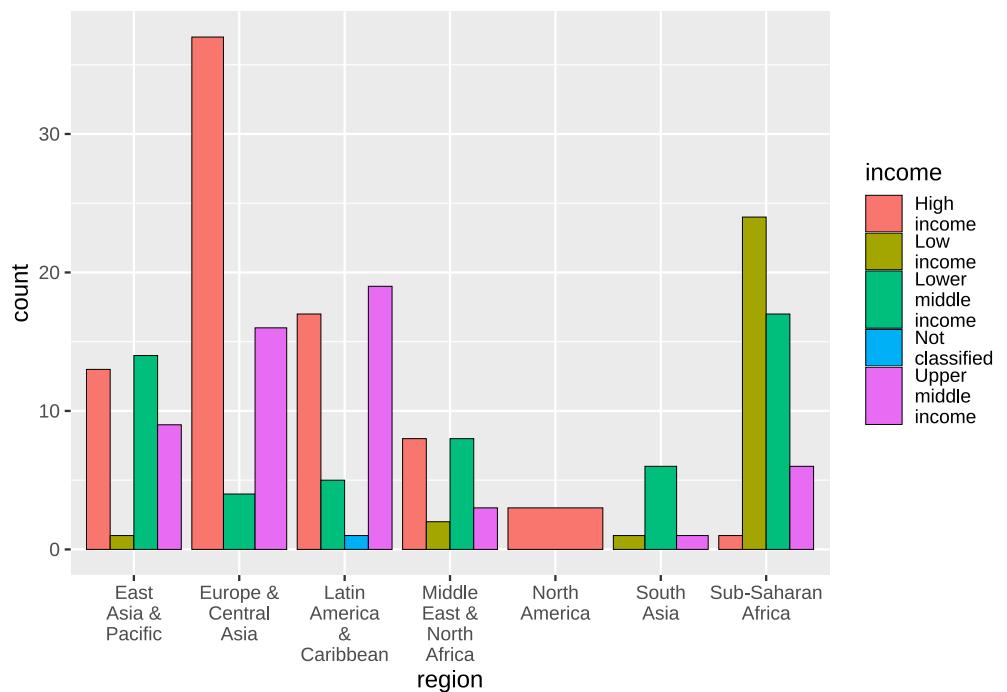
まず、scales は、Tidyverse パッケージの一部ですが、読み込まれてはいないので、加えています。library(scales) としてあれば、省略可能です。accuracy = 1 は小数点以下を省略するもので、小数点以下一位まであれば、accuracy = 0.1 とします。

```
df_wdi_extra |> filter(income != "Aggregates", year == 2020) |>
  group_by(region) |>
  ggplot(aes(x = region, fill = income)) + geom_bar(color = "black", linewidth = 0.2, position = "fill") +
  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)) +
  labs(y = "count (percent)")
```



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けれど。

```
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_fill_discrete(labels = function(x) str_wrap(x, width = 8))
```



19.5.4 折線グラフと箱ひげ図 (line graphs and boxplots)

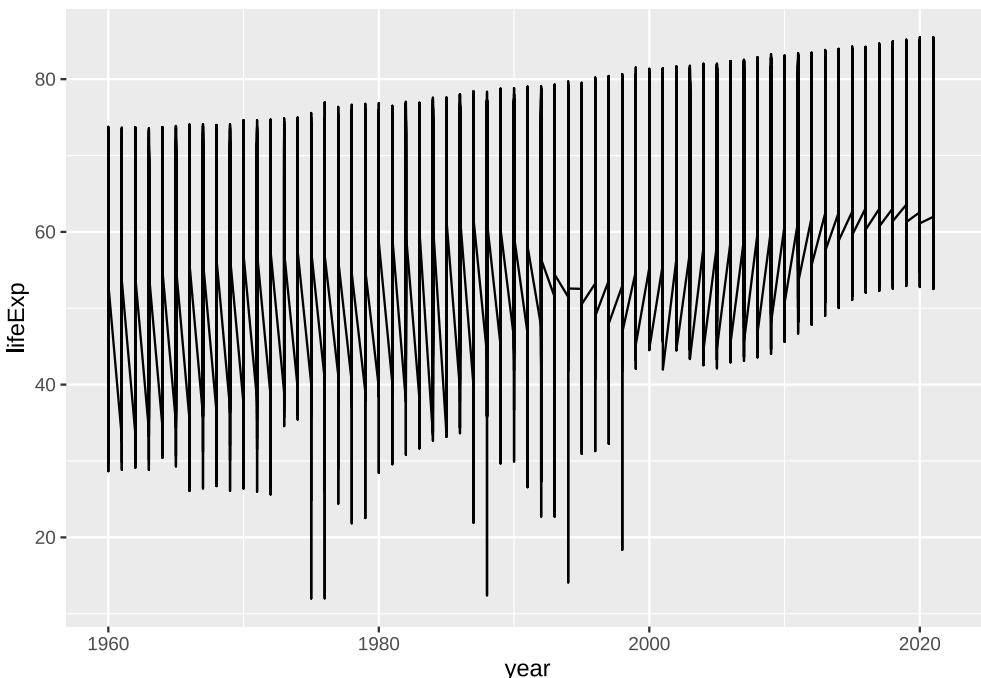
WDI は時系列データですから、折れ線グラフも使います。後ほど紹介しますが、

```
<DATA> |> ggplot(aes(year, lifeExp)) + geom_line()
```

と言った感じです。

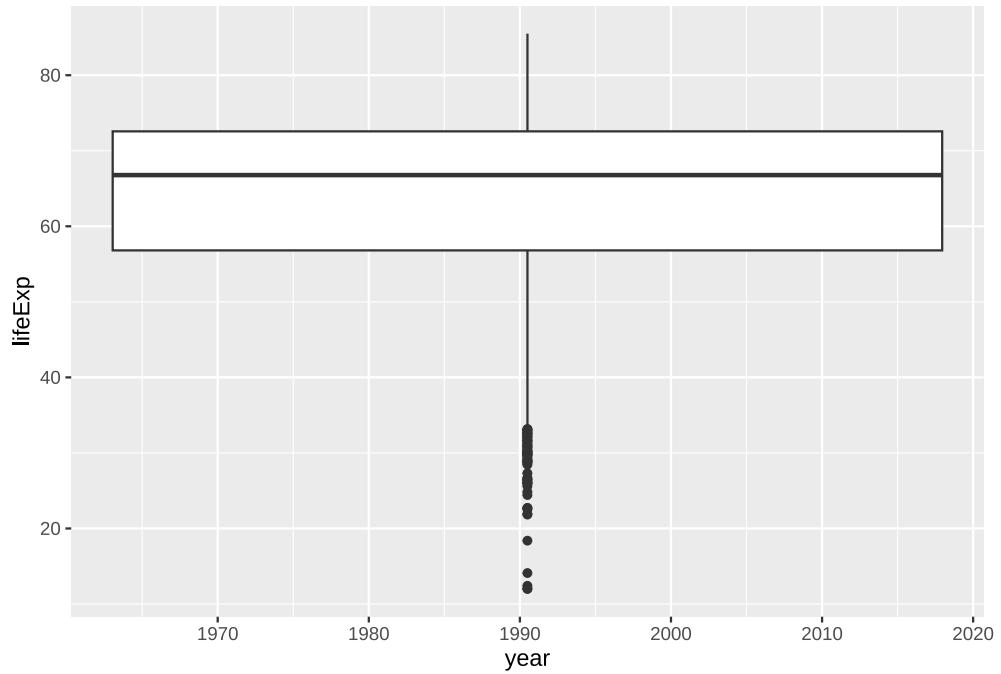
まずは、失敗例から。次のコードでグラフが描けるでしょうか。

```
df_wdi |> ggplot(aes(year, lifeExp)) + geom_line()
#> Warning: Removed 266 rows containing missing values
#> (`geom_line()`).
```



何が起こっているかわかりますか。これは、鋸の刃グラフ (saw-tooth chart) と言わ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()`).
```

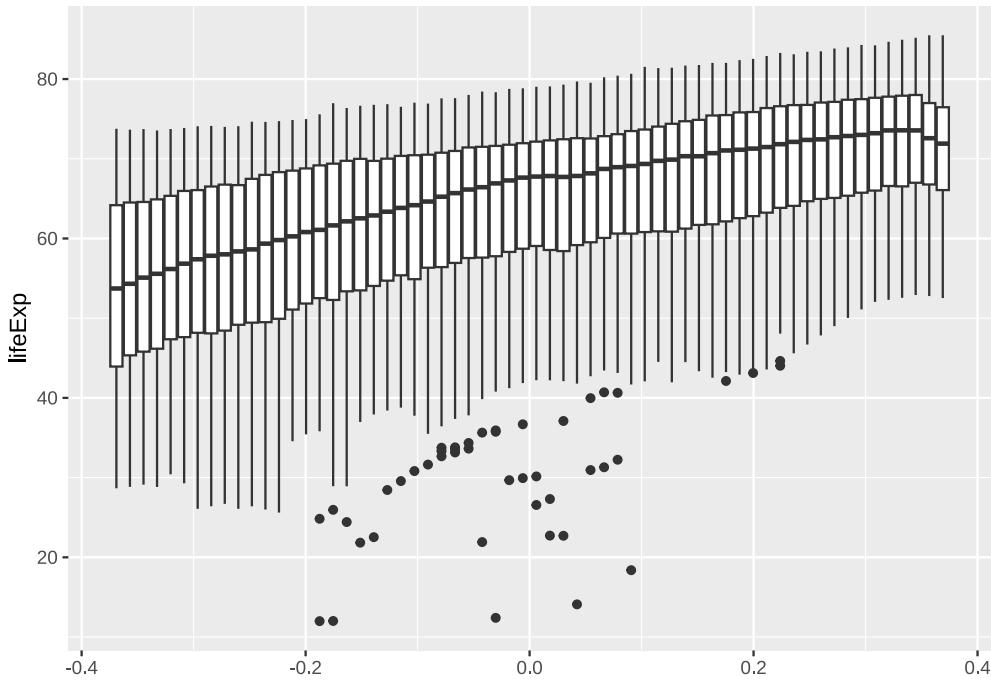


これも期待した箱ひげ図にはなっていません。年は、カテゴリーではなく、数値データですね。

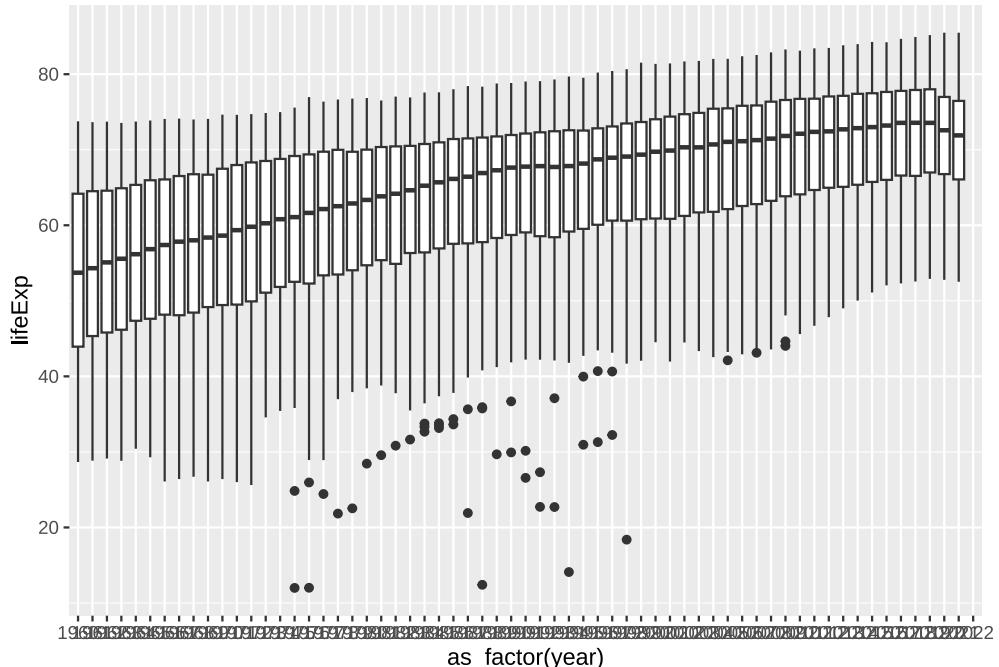
```
typeof(pull(df_wdi, year)) # same as typeof(df$year)
#> [1] "double"
```

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()` ).
```



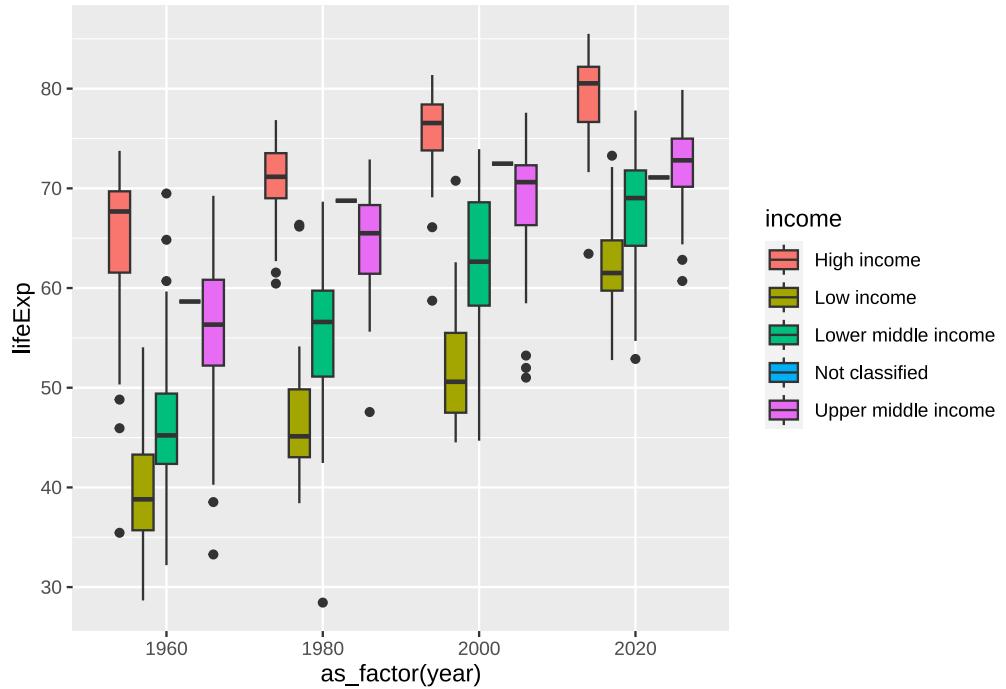
```
ggplot(df_wdi, aes(x = as_factor(year), y = lifeExp)) + geom_boxplot()
#> Warning: Removed 892 rows containing non-finite values
#> (`stat_boxplot()`).
```



■19.5.4.0.1 Box Plot

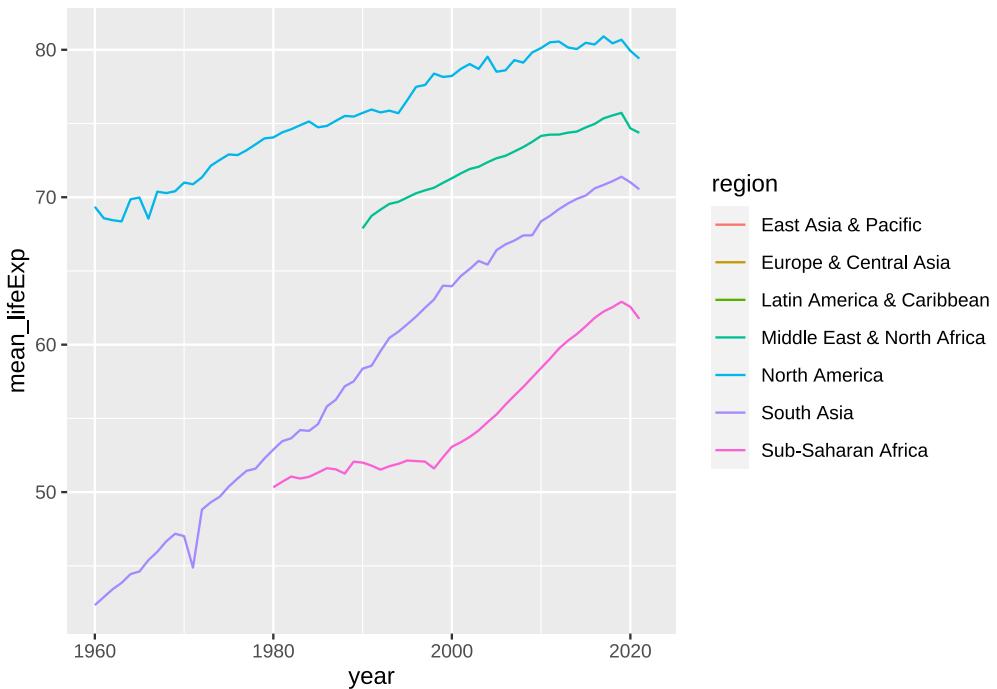
とはいって、数が多くて、色もつけてみましょう。塗りつぶしは、fill 枠の線に色をつけるのは、color ですから、ここでは、fill を使います。

```
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()`).
```



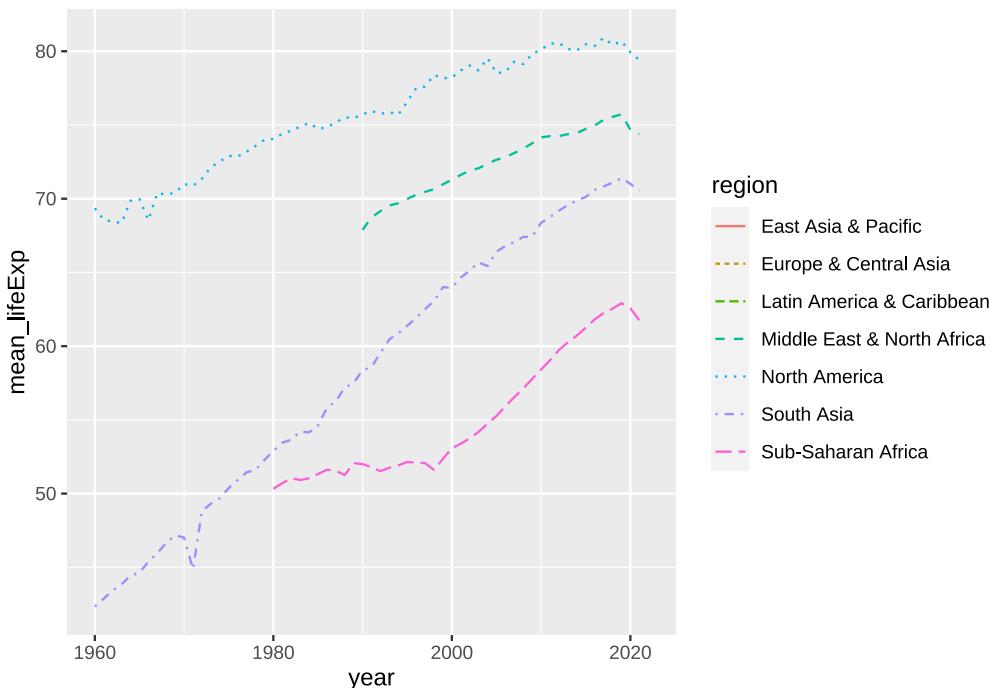
折線グラフの例としては次のようなものがあります。

```
df_lifeExp <- df_wdi_extra |> filter(region != "Aggregates") |>
  group_by(region, year) |>
  summarize(mean_lifeExp = mean(lifeExp), median_lifeExp = median(lifeExp), max_l
df_lifeExp %>% ggplot(aes(x = year, y = mean_lifeExp, color = region)) +
  geom_line()
#> Warning: Removed 243 rows containing missing values
#> (`geom_line()`).
```



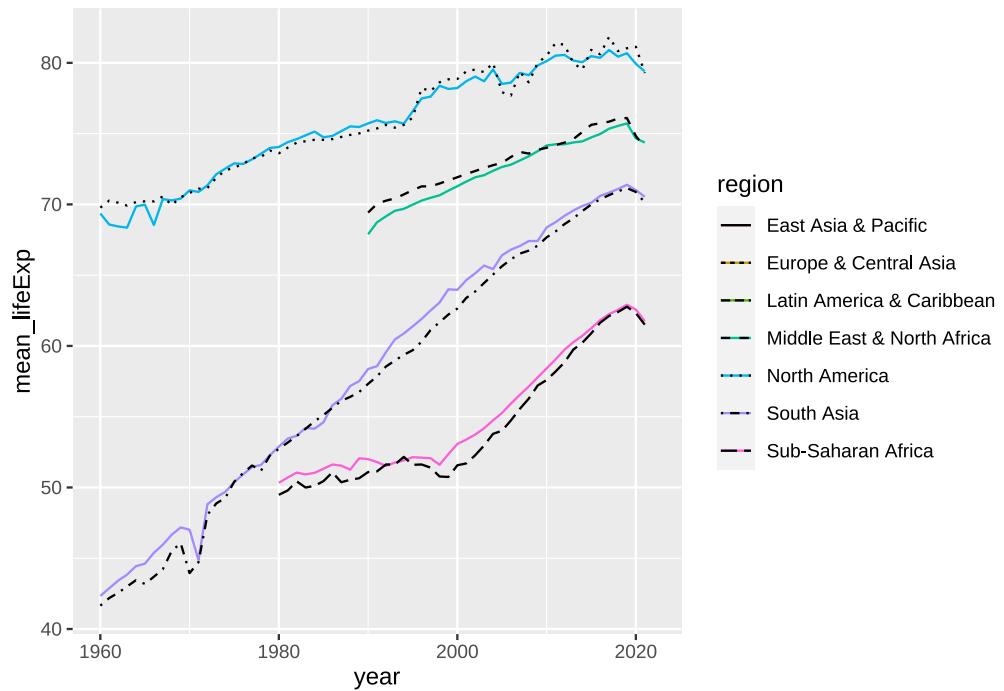
```
df_lifeExp %>% ggplot(aes(x = year, y = mean_lifeExp, color = region, linetype = region)) +
  geom_line()

#> 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()`).
```

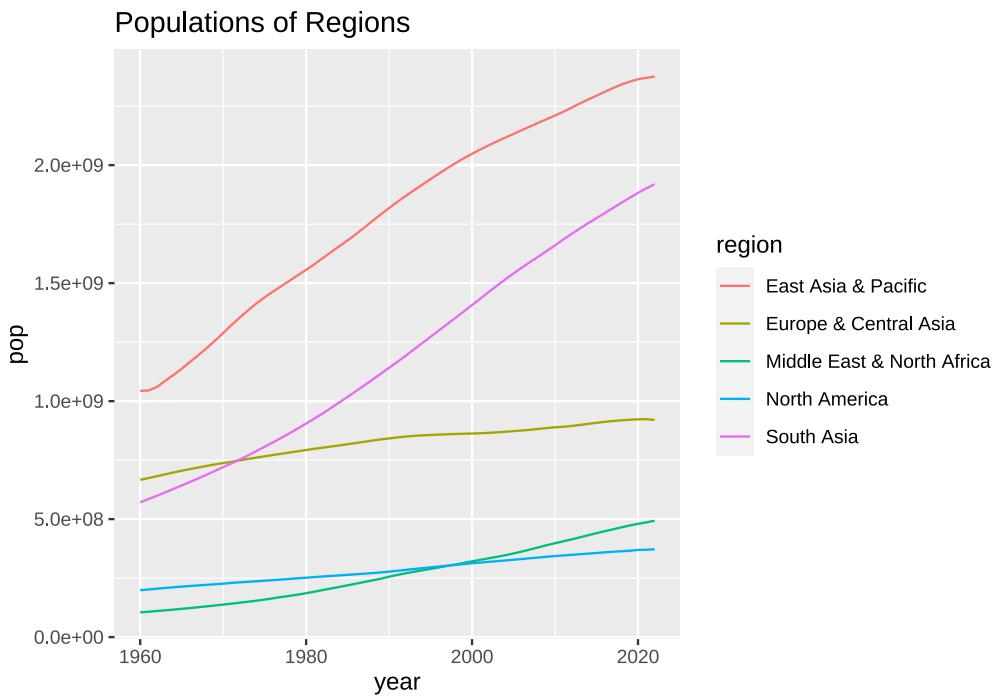


19.5.5 人口と一人当たりの国内総生産

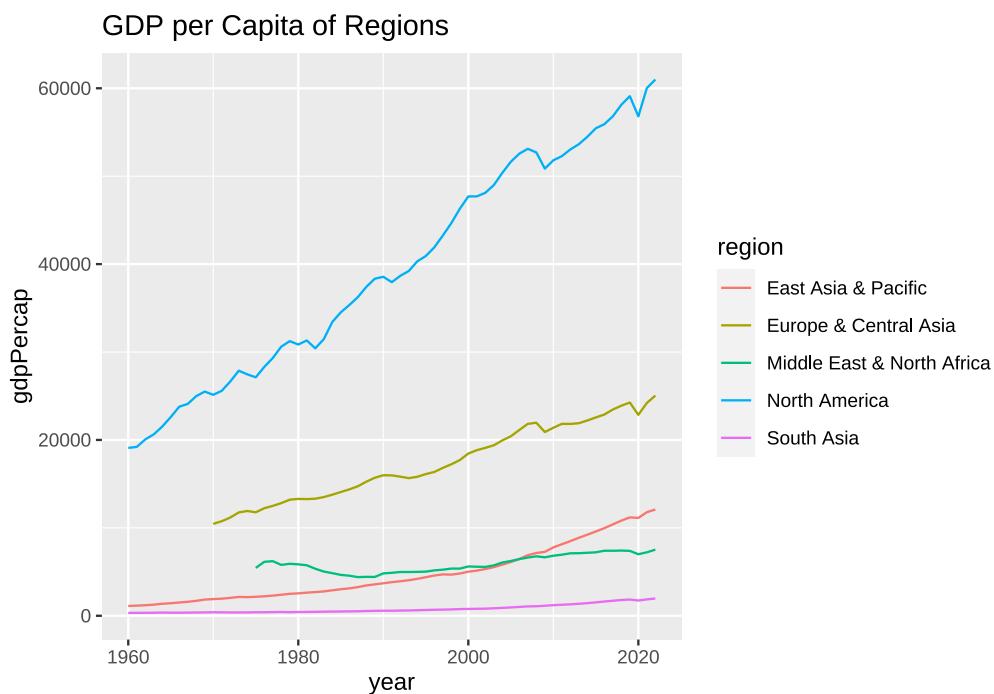
まず、出生時の平均寿命を見てみましたが、他の指標についてみてみましょう。他には、人口 (population) と、一人当たりの国内総生産 (GDP per Capita) の指標がそれぞれ、pop と、gdpPercap という変数名で含まれています。

まずは、簡単に、地域ごとの経年変化を見てみましょう。平均を計算してという方法もありますが、集計したものもデータの中にあるので、上で作った、region 名を使って、それで、グラフを書いてみましょう。

```
df_wdi_extra |> filter(region == "Aggregates") |>
  filter(country %in% regions) |> drop_na(pop) |>
  ggplot(aes(year, pop, color = country)) + geom_line() +
  labs(title = "Populations of Regions", color = "region")
```

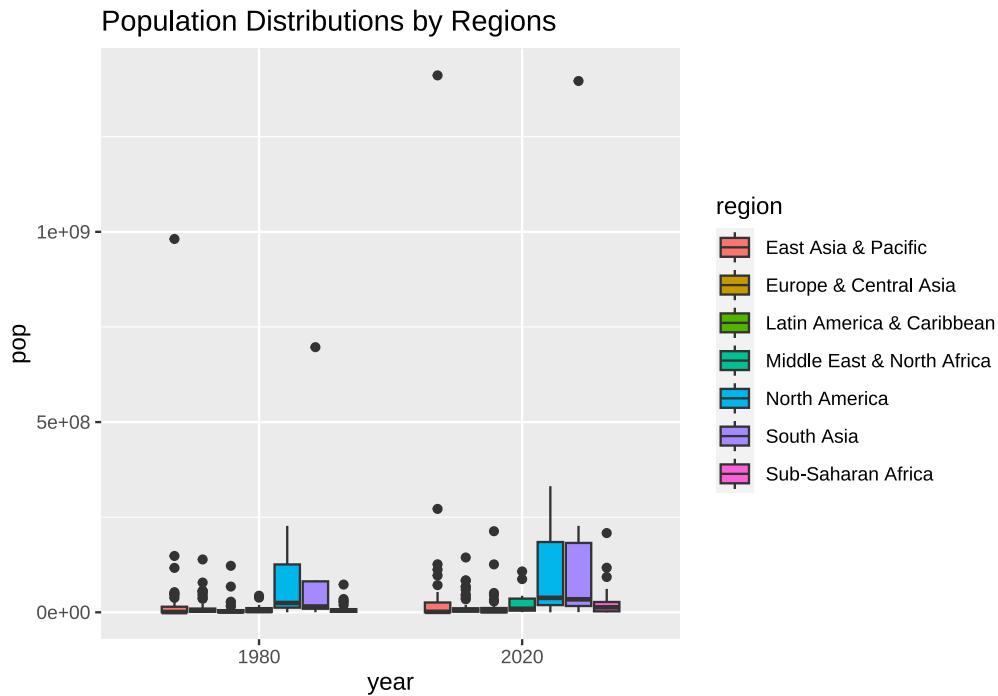


```
df_wdi_extra |> filter(region == "Aggregates") |>
  filter(country %in% regions) |> drop_na(gdpPercap) |>
  ggplot(aes(year, gdpPercap, color = country)) + geom_line() +
  labs(title = "GDP per Capita of Regions", color = "region")
```

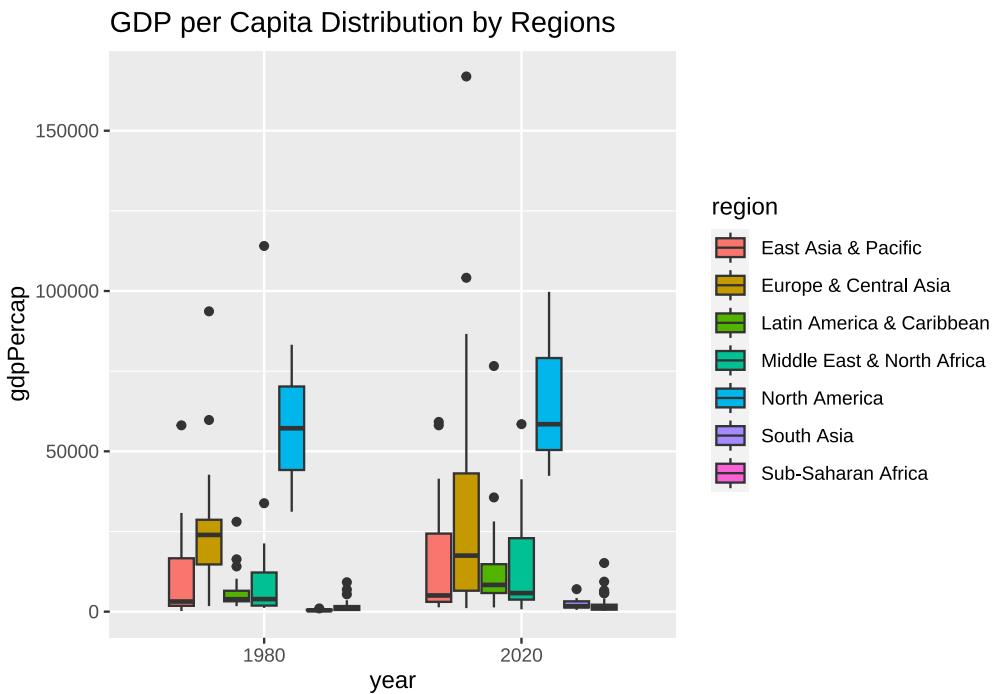


大体、期待したものが描けたかと思います。では、箱ひげ図はどうでしょうか。何年かを切り取り、年毎の地域ごとの分布を見てみましょう。古いものは地域によってはデータがなさそうですから、1980 年と、2020 年にしましょう。

```
df_wdi_extra |> filter(region != "Aggregates", year %in% c(1980, 2020)) |>
  drop_na(pop) |>
  ggplot(aes(factor(year), pop, fill = region)) + geom_boxplot() +
  labs(title = "Population Distributions by Regions", x = "year", fill = "region")
```



```
df_wdi_extra |> filter(region != "Aggregates", year %in% c(1980, 2020)) |>
  drop_na(gdpPercap) |>
  ggplot(aes(factor(year), gdpPercap, fill = region)) + geom_boxplot() +
  labs(title = "GDP per Capita Distribution by Regions", x = "year", fill = "region")
```

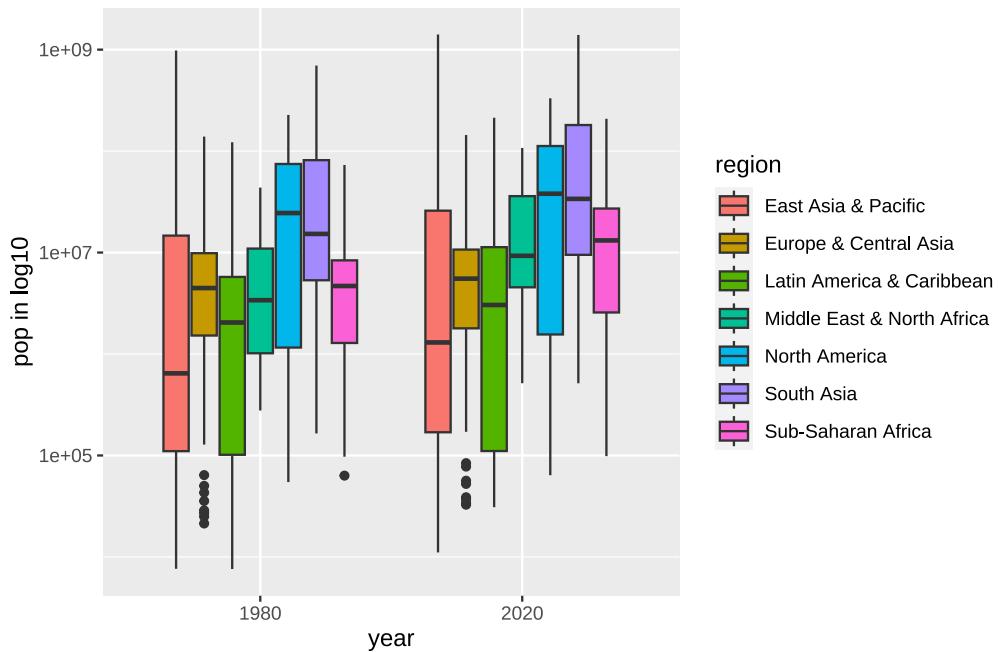


一応、期待したものが表示されたようですが、どうも、下の方が固まっていて、見にくいですね。特に人口のほうは、問題がありそうです。Y 軸でみると、指標表示になっています。小さな国から、人口の多い国までありますから、かなり、幅があるのでしょうね。

両方とも、`log10` で表示してみましょう。変形 `dplyr` のところで使いました。1,000 だと 3、10,000 だと 4、100,000 だと 5、1,000,000 だと 6、10,000,000 だと 7、100,000,000 だと 8 と、零の数が表示されるのでした。

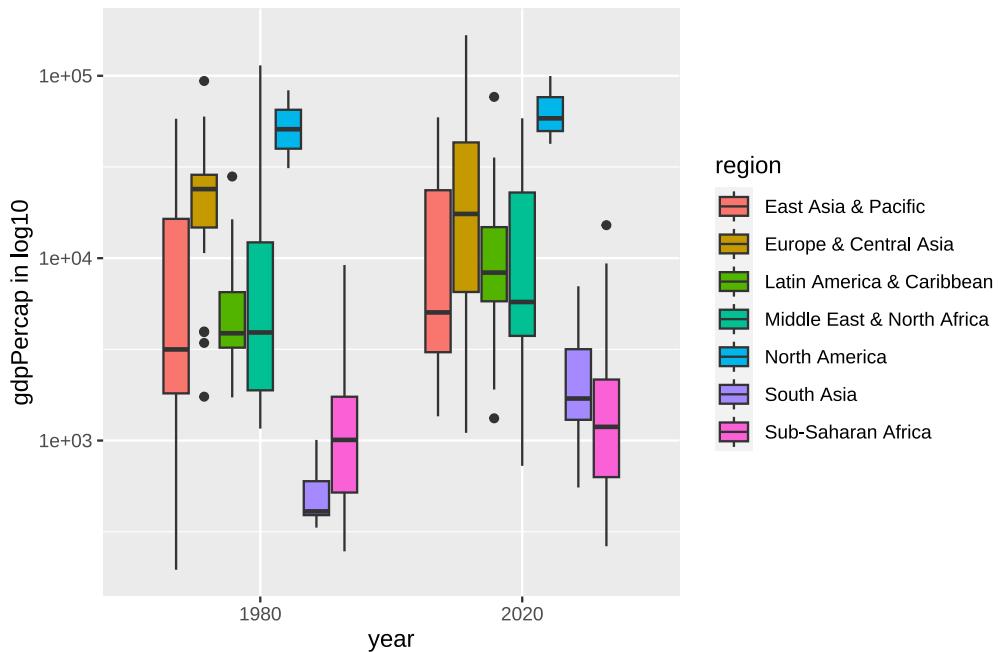
```
df_wdi_extra |> filter(region != "Aggregates", year %in% c(1980, 2020)) |>
  drop_na(pop) |>
  ggplot(aes(factor(year), pop, fill = region)) + geom_boxplot() +
  scale_y_log10() +
  labs(title = "Population Distributions by Regions",
       x = "year", y = "pop in log10", fill = "region")
```

Population Distributions by Regions



```
df_wdi_extra |> filter(region != "Aggregates", year %in% c(1980, 2020)) |>
  drop_na(gdpPercap) |>
  ggplot(aes(factor(year), gdpPercap, fill = region)) + geom_boxplot() +
  scale_y_log10() +
  labs(title = "GDP per Capita Distributions by Regions",
       x = "year", y = "gdpPercap in log10", fill = "region")
```

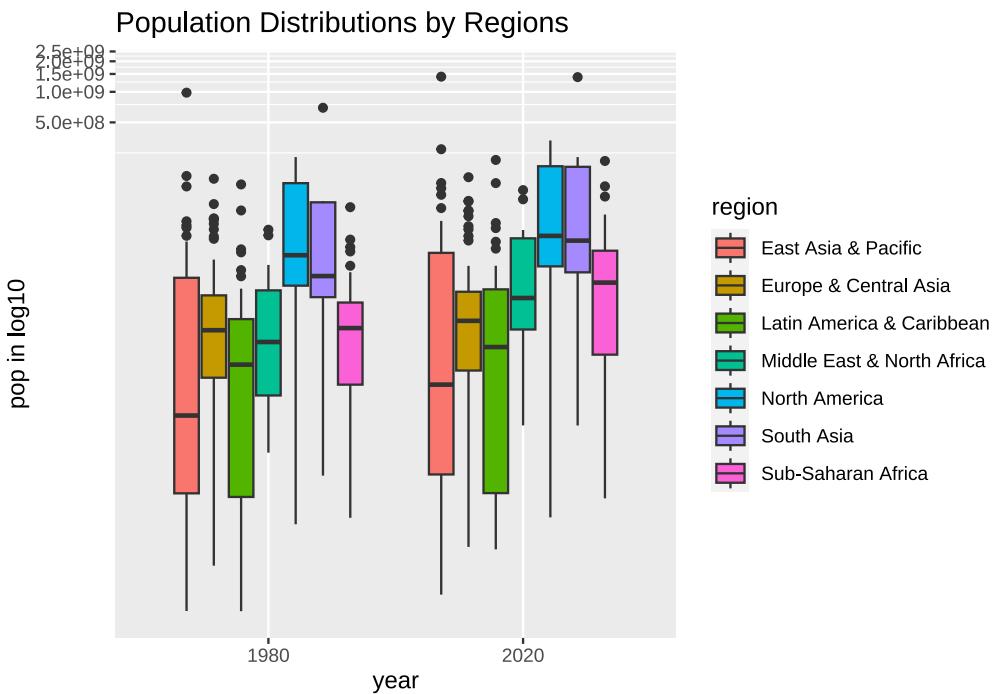
GDP per Capita Distributions by Regions



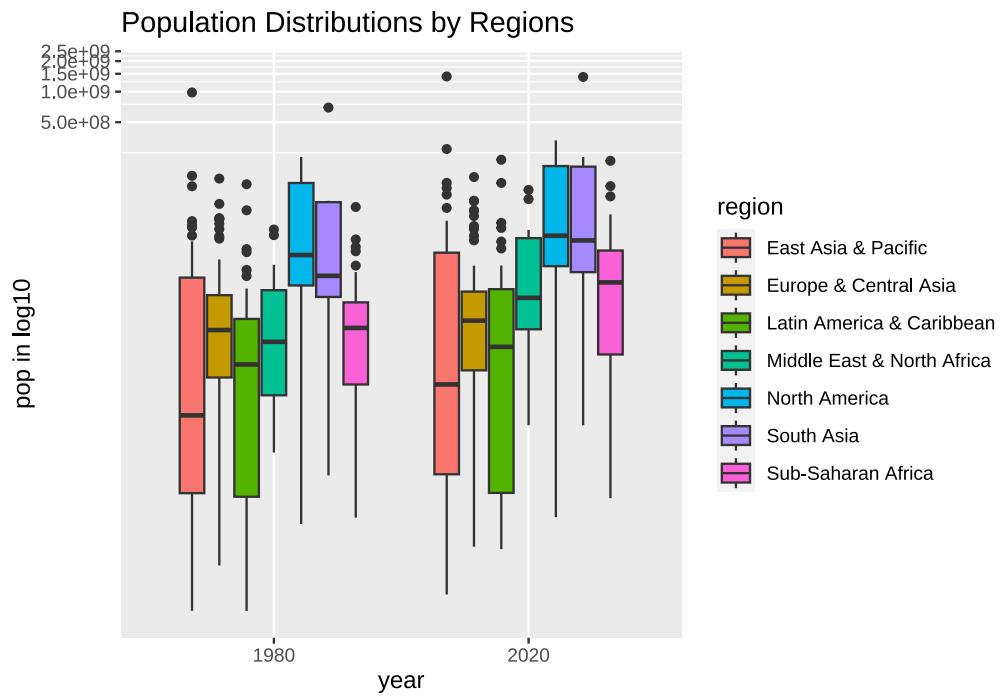
少しみやすいですかね。実は、 \log_{10} で表示する仕方はもう一つあります。座標軸を等間

隔にしないで、 \log_{10} の切り方に変える方法です。

```
df_wdi_extra |> filter(region != "Aggregates", year %in% c(1980, 2020)) |>
  drop_na(pop) |>
  ggplot(aes(factor(year), pop, fill = region)) + geom_boxplot() +
  coord_trans(y = "log10") +
  labs(title = "Population Distributions by Regions",
       x = "year", y = "pop in log10", fill = "region")
```

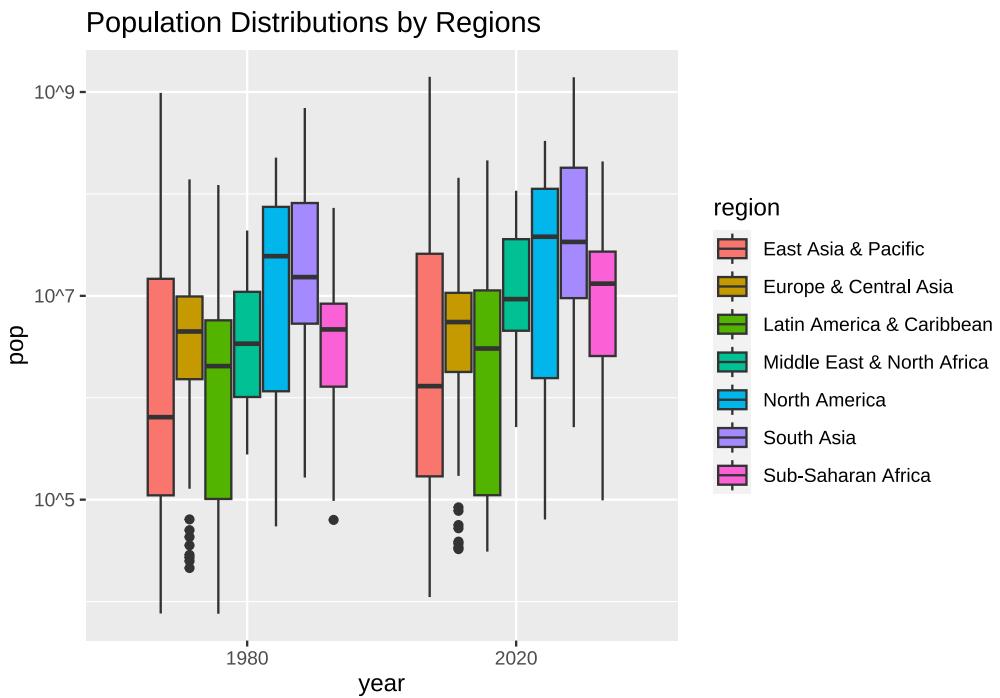


```
df_wdi_extra |> filter(region != "Aggregates", year %in% c(1980, 2020)) |>
  drop_na(pop) |>
  ggplot(aes(factor(year), pop, fill = region)) + geom_boxplot() +
  coord_trans(y = "log10") +
  labs(title = "Population Distributions by Regions",
       x = "year", y = "pop in log10", fill = "region")
```



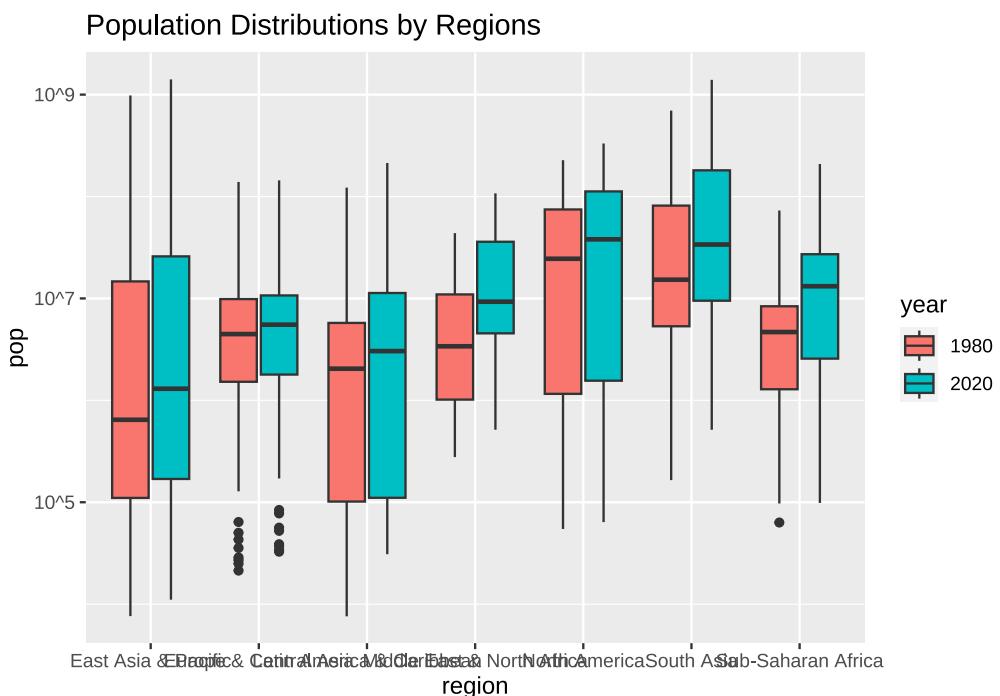
指数表示は避けたいということでしたら、次のようにすることも可能です。日本語にすることも可能です。

```
df_wdi_extra |> filter(region != "Aggregates", year %in% c(1980, 2020)) |>
  drop_na(pop) |>
  ggplot(aes(factor(year), pop, fill = region)) + geom_boxplot() +
  scale_y_log10(breaks = c(1e5, 1e7, 1e9), labels = c("10^5", "10^7", "10^9")) +
  labs(title = "Population Distributions by Regions",
       x = "year", y = "pop", fill = "region")
```



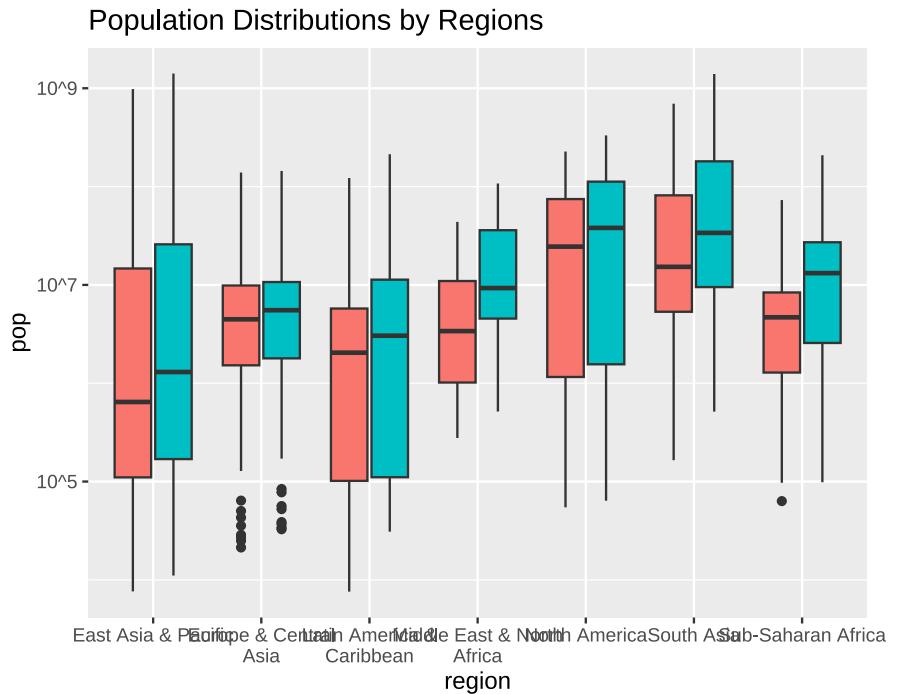
まずは、地域ごとに変化を見るにはどうしたら良いでしょうか。

```
df_wdi_extra |> filter(region != "Aggregates", year %in% c(1980, 2020)) |>
  drop_na(pop) |>
  ggplot(aes(region, pop, fill = factor(year))) + geom_boxplot() +
  scale_y_log10(breaks = c(1e5, 1e7, 1e9), labels = c("10^5", "10^7", "10^9")) +
  labs(title = "Population Distributions by Regions",
       x = "region", y = "pop", fill = "year")
```



地域名が重なっているところがありますね。

```
df_wdi_extra |> filter(region != "Aggregates", year %in% c(1980, 2020)) |>
  drop_na(pop) |>
  ggplot(aes(region, pop, fill = factor(year))) + geom_boxplot() +
  scale_x_discrete(labels = function(x) str_wrap(x, width = 20)) +
  scale_y_log10(breaks = c(1e5, 1e7, 1e9), labels = c("10^5", "10^7", "10^9")) +
  labs(title = "Population Distributions by Regions",
       x = "region", y = "pop", fill = "year")
```



19.5.6 散布図

```
df_wdi_extra |> filter(year == 2021) |> drop_na(pop, gdpPercap) |>
  ggplot(aes(pop, gdpPercap, color = region)) +
  geom_point() + scale_x_log10() + scale_y_log10()
```