

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)

2023-06-03

目次

| | |
|---|-----------|
| この文書について | 7 |
| 著者について | 7 |
| コンピュータ言語について | 7 |
| 言語について | 8 |
| PDF、ePub 版について | 8 |
| 参考 | 8 |
| 第 1 章 はじめに | 9 |
| 1.1 データサイエンスとは | 9 |
| 1.2 「データサイエンスを始めましょう」の特徴 | 15 |
| 1.3 参考 | 16 |
| 第 2 章 学ぶ内容 | 17 |
| 2.1 データサイエンス入門 | 17 |
| 2.2 第一部 パブリックデータ | 17 |
| 2.3 第二部 基本 | 17 |
| 2.4 第三部 国際機関のデータの活用 | 17 |
| 2.5 第四部 探索的データ分析 Exploratory Data Analysis | 17 |
| 2.6 第五部 分析例 | 17 |
| 2.7 付録 | 18 |
| 第 I 部 PART I PUBLIC DATA | 19 |
| 第 3 章 Public Data | 21 |
| 3.1 オープンデータ | 21 |
| 3.2 日本から世界を見る | 22 |
| 3.3 世界銀行 (World Bank) | 22 |
| 3.4 世界のさまざまな課題から見る | 24 |
| 3.5 OECD | 24 |
| 3.6 UN Data | 24 |
| 3.7 Our World in Data | 24 |
| 3.8 Eurostat | 24 |

| | | |
|----------------|---|-----------|
| 第 II 部 | PART II BASICS | 25 |
| 第 4 章 | R Studio で R | 27 |
| 4.1 | はじめに | 27 |
| 4.2 | R と R Studio | 27 |
| 4.3 | R と R Studio のインストール | 28 |
| 4.4 | クラウド - Posit Cloud | 34 |
| 4.5 | R のその他の利用方法 | 36 |
| 第 5 章 | R Markdown | 39 |
| 5.1 | Reproducible and Literate Programming | 39 |
| 5.2 | 準備：パッケージのインストール | 40 |
| 5.3 | R Notebook | 40 |
| 5.4 | 日本語のテンプレート | 41 |
| 5.5 | R Markdown いくつかの Output | 41 |
| 5.6 | YouTube Video - rmarkdown | 42 |
| 第 6 章 | R Basics | 45 |
| 6.1 | プロジェクト - Project | 45 |
| 6.2 | コンソールで実行 - Run in Console | 46 |
| 6.3 | RStudio について | 49 |
| 6.4 | R Script 実行記録 | 50 |
| 6.5 | パッケージ - Packages | 58 |
| 6.6 | 練習問題 Posit Primers | 59 |
| 6.7 | 参考文献 References | 60 |
| 6.8 | YouTube Video - getstarted | 60 |
| 第 III 部 | PART III INSTITUTIONAL DATA | 61 |
| 第 7 章 | World Bank | 63 |
| 7.1 | World Development Indicator (WDI) | 63 |
| 7.2 | WDI パッケージ | 68 |
| 7.3 | 可視化 Visualization | 78 |
| 7.4 | 課題 Assignment | 82 |
| 第 IV 部 | PART IV EDA | 83 |
| 第 8 章 | 探索的データ解析 | 85 |
| 8.1 | 探索的データ解析 (EDA) とは | 85 |
| 8.2 | 探索的データ解析 (EDA) の一例 | 85 |

| | | |
|-------|-----------------------------------|-----|
| 第 V 部 | PART V EXAMPLES | 95 |
| 第 9 章 | Example 1 | 97 |
| 付録 A | 日本語の扱いについて | 99 |
| A.1 | 日本語・中国語・韓国語 | 99 |
| A.2 | Base R でタイトルに日本語 | 99 |
| A.3 | 列名や、データに日本語 | 100 |
| A.4 | kable で表示 | 100 |
| A.5 | ggplot でグラフを作成 | 101 |
| A.6 | 備考： | 101 |
| A.7 | 参考：日本語の表示について | 102 |
| 付録 B | IT ツール | 105 |
| B.1 | Git と GitHub | 105 |
| 付録 C | Bookdown | 117 |
| C.1 | About | 117 |
| C.2 | Hello bookdown | 118 |
| C.3 | Cross-references | 118 |
| C.4 | Parts | 119 |
| C.5 | Footnotes and citations | 120 |
| C.6 | Blocks | 121 |
| C.7 | Sharing your book | 122 |

この文書について

データサイエンスを始めてみませんか。

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。

みなさんも一緒にデータサイエンスを学んでみませんか。

著者について

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

幸運にも、2019 年 9 月の日本数学会教育委員会主催教育シンポジウムで、「文理共通して行う数理・データサイエンス教育」という題で、話す機会が与えられ、その後、あることが契機となり、2020 年度から、毎年、冬学期（12 月から 2 月）に、大学院一般向け（分野の指定なし）の授業、「研究者のためのデータ分析（Data Analysis for Researchers）」を担当しています。複数の教員で担当しますが、基本的な部分は、わたしが教えています。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。

コンピュータ言語について

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。特に、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、簡単なものであれば、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 によって、可能になっています。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もできることも、R をお勧めする理由です。

ただし、日本語のものは、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。

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、日本語の扱いは、負担になることが多いからです。最近では、コードの中で日本語を使用しても、ほとんど、問題は起きないように思います。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。

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できていますから、日本語の利用についても、追って説明していきます。APPENDIX A を参照してください。

最初は、みなさんも、変数（variable）や、オブジェクト（object）に名前をつけるときは、半角英数を使い、日本語は、使わないようにすることをお勧めします。

PDF、ePub 版について

実は、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、ほとんど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

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。

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

ChatGPT (2023.6.3)

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

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

データサイエンスは、さまざまな分野で活用されています。例えば、ビジネス分野では、マーケティングキャンペーンの最適化、顧客セグメンテーション、需要予測などに利用されます。医療分野では、病気の診断や治療計画の最適化、新薬の開発

などに応用されます。また、交通や都市計画、環境保護、金融など、さまざまな領域でデータサイエンスの手法が活用されています。

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

ちょっと日本語がごちゃごちゃい感じもするので、英語での問いと答えも書いておきましょう。

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.

領域 (multidisciplinary field) という言葉も使われていますから、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。みなさんが、少しずつ学びながら、考えていただくのが一番よいと思いますが、いくつか理由があると思います。

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

付け加えておかないといけないと考えていることがもう一つあります。それは、問題解決と書きましたが、課題と向き合うという表現でもよいと思います。これまでは、因果関係が単純な課題については、根拠をしめして、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必要があると、その中で、さまざまな指標に関して評価値が高いものを選択することが可能になって来たと言うことです。

そのことが、もう一歩進んで、AI（Artificial Intelligence 人工知能）を支える技術に使われています。ここで使った、ChatGPT の GPT は、Generative Pre-trained Transformer の略ですが、LLM（Large Language Model）によって、自然なこと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、応答とすることが可能になっているのです。

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

ChatGPT (2023.6.3)

データサイエンスが最近注目されている理由は、以下の要素によるものです。

1. ビッグデータの増加: 近年、デジタル技術の進歩により、膨大な量のデータが生成されるようになりました。ウェブサイトのアクセスログ、ソーシャルメディアの投稿、センサーデータ、トランザクションデータなど、多様なソースから大量のデータが利用可能です。データサイエンスは、これらのビッグデータを活用して洞察を得ることができるため、需要が高まっています。
2. 技術の進歩: データサイエンスを支える技術の進歩も、その注目を高めています。クラウドコンピューティングの発展やオープンソースのツールとライブラリの普及により、データの収集、処理、分析が容易になりました。さらに、機

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。
9. 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。

10. 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。
11. 組織内のデータ文化の普及: データサイエンスの注目度の高まりに伴い、企業や組織においてデータ文化の重要性が認識されています。データサイエンスの手法やアプローチを組織全体で活用し、データに基づいた意思決定や問題解決を行うことが求められています。組織内でのデータの収集、管理、分析の能力向上に取り組むことで、競争力を高めることができます。
12. 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、今後もその重要性和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、パブリックデータとか、オープンデータと言われる、誰でも利用できるデータが、使いやすい形で、提供されているということです。

ChatGPT (2023.6.3)

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

1. データリテラシーの重要性: データは現代社会においてますます重要な役割を

果たしています。データに基づく意思決定や問題解決は、個人の日常生活から組織や社会の活動まで広がっています。データサイエンスを学ぶことで、データを扱うスキルや知識を身につけることができます。データを理解し、適切に分析して活用する能力は、個人の能力向上や自己啓発にも繋がります。

2. 個人の成長とキャリアの展望: データサイエンスは求人市場で高い需要があり、競争力のあるキャリアの一つとなっています。データサイエンスのスキルを持つことで、自身のキャリアの成長や就業機会の拡大が期待できます。データサイエンスの知識やスキルは、さまざまな職業や業界で役立つだけでなく、起業や独立の道も開ける可能性があります。
3. 自己啓発と問題解決能力の向上: データサイエンスは、問題解決や意思決定において非常に有用なツールです。データサイエンスを学ぶことで、情報を分析し洞察を得る力や、パターンを見つける力、予測や最適化の能力を養うことができます。これにより、個人の問題解決能力や意思決定力が向上し、より効果的な選択や行動ができるようになります。

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

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

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

1.1.3 AI の活用

この電子ブックでは、AI も活用していきたいと思います。クリティカルシンキングのことも、書かれていますが、どうしても、人は、バイアスのかかった考えかたをしてしまいます。それは、経験も知識も限られていますし、単純な答えを受け入れたい傾向ということもあるかもしれません。

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

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

1.2 「データサイエンスを始めましょう」の特徴

1.2.1 学習者として想定しているのは

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。

1.2.2 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、それが可能になって来たこと。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。

1.2.3 世界のデータをみること

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

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

1.2.4 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ればと願っています。

1.3 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

世界銀行のデータで、どのようなことができるか、外観してみます。

2.2 第一部 パブリックデータ

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。

2.3 第二部 基本

R の基本、および、`tidyverse` パッケージの基本を、いくつかの、基本的なパブリックデータを使いながら、学びます。

2.4 第三部 国際機関のデータの活用

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。

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

一つ一つのステップについて、より詳しく学びます。

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。

第 I 部

PART I PUBLIC DATA

第3章

Public Data

まずは、パブリックデータを見てみましょう。大きな機関のパブリックデータには、ダッシュボード（dashboard）と呼ばれている、パラメタを変更して、そのグラフを描くなどの機能が付いているものもあります。

3.1 オープンデータ

3.1.1 Open Government Data Toolkit: Open Data Defined

The term **Open Data** has a very precise meaning. Data or content is open if anyone is free to use, re-use or redistribute it, subject at most to measures that preserve provenance and openness.

1. The data must be *legally open*, which means they must be placed in the public domain or under liberal terms of use with minimal restrictions.
2. The data must be *technically open*, which means they must be published in electronic formats that are machine readable and non-proprietary, so that anyone can access and use the data using common, freely available software tools. Data must also be publicly available and accessible on a public server, without password or firewall restrictions. To make Open Data easier to find, most organizations create and manage Open Data catalogs.

オープンデータの定義

1. オープンデータという言葉は、非常に正確な意味を持っています。データまたはコンテンツは、誰でも自由に使用、再利用、再配布でき、せいぜい出所とオープン性を維持するための措置に従うだけであればオープンです。
2. 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るか、最小限の制限で自由な使用条件のもとに置かれなければなりません。データは技術的にオープンでなければならない。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利用できるように、機械可読で

非専有の電子フォーマットで公開されていなければならない。また、データは一般に公開され、パスワードやファイアウォールによる制限を受けずに、公共のサーバーでアクセスできなければなりません。オープンデータを見つけやすくするために、ほとんどの組織がオープンデータカタログを作成し管理しています。

3.2 日本から世界を見る

以下では、世界銀行の、世界開発指標を利用するが、他にも、UNdata、OECD data、WID、Eurostat、Our World in Data など、同様の、ダッシュボードを備えており、データの提供もしている。日本では、e-Stat、ダッシュボード

- 日本のデータはどうでしょうか。

3.3 世界銀行 (World Bank)

- World Bank: <https://www.worldbank.org>
- Who we are:
 - To end extreme poverty: By reducing the share of the global population that lives in extreme poverty to 3 percent by 2030.
 - To promote shared prosperity: By increasing the incomes of the poorest 40 percent of people in every country.
- World Bank Open Data: <https://data.worldbank.org>
 - Data Bank, World Development Indicators, etc.

3.3.1 世界開発指数 (World Development Indicator (WDI))

- World Development Indicators (WDI) : the World Bank's premier compilation of cross-country comparable data on development; 1400 time series indicators
 - Themes: Poverty and Inequality, People, Environment, Economy, States and Markets, Global Links
 - Open Data & DataBank: Explore data, Query database
 - Bulk Download: Excel, CSV
 - API Documentation

3.3.2 World Bank: WDI - World Development Indicators

- 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>

3.3.3 世界開発指標 (World Development Indicator)

- 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
 - Bulk Download: Excel, CSV
 - API Documentation

3.3.4 例

3.3.4.1 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 -

3.3.4.2 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

3.3.4.3 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

3.3.4.4 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

3.3.4.5 Military expenditure (current USD)

2021 年の軍事費 (現在の米ドル)

軍事費の年次変化

3.3.4.6 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

3.4 世界のさまざまな課題から見る

3.5 OECD

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

3.6 UN Data

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

3.7 Our World in Data

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

3.8 Eurostat

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

第 II 部

PART II BASICS

第 4 章

R Studio で R

4.1 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、確認することを目指します。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。

4.2 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 で利用することが可能です。

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

RStudio は、R と Python のための、総合開発環境 (IDE) です。RStudio には、

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

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

4.3 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。

4.3.1 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。

4.3.2 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。

4.3.3 動作確認

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

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

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

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。

*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。

4.3.4 トラブル・シューティング（1）

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

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

*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。

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、問題が起こります*⁴。解決方法は上に書いたように、一つの解決方法として、TinyTeX を、Home とは異なるところにインストールしています*⁵。

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

4.3.4.1 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。

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のが大切ですが、簡単には理解できない場合が多いかと思います。そこで、まず、試してみる方法を以下に書きます。

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

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

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

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

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

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

備考：環境変数の Path について

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

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

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

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

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

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

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

3. つぎに、アカウントを作成しましょうとでますから、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。

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

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

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

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

*8 Microsoft アカウントを設定すると

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

詳細は、下のリンクなどを参照してください。

参照リンク

4.4 クラウド - Posit Cloud

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

4.4.1 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、いくつかアカウントを持って、活用しています。

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

特徴：制限など (Key Features)

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

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

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

^{*9} 右の i マークを押すと詳細な条件を確認できます。

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

TIPs :

プロジェクトをダウンロード :

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

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

他のアカウントとの共有 :

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

4.4.2 PositCloud Shared Project

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

1. まず、自分の、PositCloud アカウントにログインする。
2. リンク をクリック
3. 上に、Temporary Copy と出ています。このままでの、右下の、窓枠から、Files を選び、開いて中身を見たり、Code Chunk を実行したり、Preview や、knit をすることも可能です。Try Again と表示されますが、それをクリックすると、表示さ

れます。R Markdown などの詳細は、R Markdown の章をみてください。

4. 上に、Save a Permanent Copy とあります。これを、クリックすると、自分の、プロジェクトとすることができ、編集も可能になります。

4.4.3 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。

4.5 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。

4.5.1 Google colab で R

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

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

4.5.1.1 参考

- How to use R in Google Colab:

4.5.2 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、SinUp 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` と最初に入力すれば、そのあと、R のコマンドを実行できます。
 - New から、RMarkdown を選択すると、RMarkdown 文書を作成することができます。
 - 他にも、TeX や、Python Notebook (Jupyter Notebook) など、さまざまな道具が提供されています。
 - Maxima と呼ばれる、Maathematica の Free のバージョンから改良した、SageMath からスタートしたサイトです。

第 5 章

R Markdown

5.1 Reproducible and Literate Programming

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

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

5.1.1 目的、問いなど

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

5.1.2 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。

5.1.3 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。

5.1.4 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。

5.1.5 まとめ：R Markdown の目的

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

5.2 準備：パッケージのインストール

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

- tidyverse
- rmarkdown
- tinytex

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

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

5.3 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 of 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.

5.4 日本語のテンプレート

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

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

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

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>

5.5 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
```

5.6 YouTube Video - rmarkdown

5.6.1 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が、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。

5.6.1.1 tinytex によるインストール方法

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

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

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

注：矢内勇生さんのサイトには、詳細な説明があります。

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

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

第 6 章

R Basics

6.1 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。

6.2 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。）

6.2.1 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 ...
```

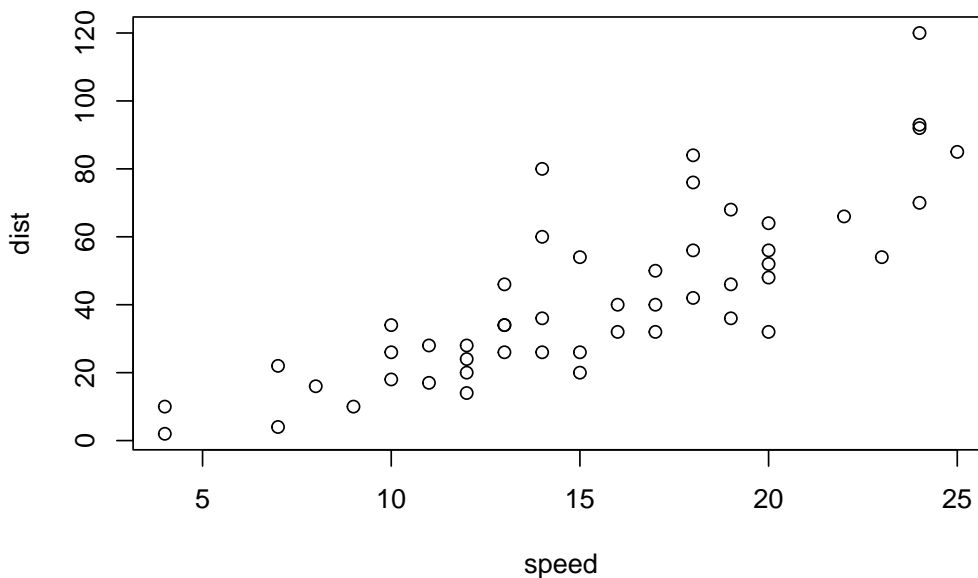
`str(cars)` は、`cars` という、R に付属している、データの構造（structure）を表示します。`data.frame` とありますが、これは、矩形になったデータ（各列の長さがおなじ）の一番簡単なクラスの名前で、2 変数、それぞれが、50 個の数値データ（numerical data）からなっていることがわかります。

`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 Quadrant)、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 Quadrant)、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。

6.2.2 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` と入れても、同じ結果が得られます。内容を確認してください。

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

- `?datasets`

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

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

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

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

- `library(help = "datasets")`

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

`cars` も見つかりましたか。

6.2.3 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。

日本語に戻りたいときは、次のようにします。

言語を日本語に設定：`Sys.setenv(LANG = "ja")`

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

6.2.4 練習

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

6.3 RStudio について

RStudio は多くの機能を持っています。

6.3.1 四つの窓枠とタブ Four Panes and Tabs

- 左上 (Top Left) : スクリプトや文書、データなどの編集 (Source Editor)
- 右上 (Top Right) : 環境変数 (Environment) , 履歴 (History) など (etc.)
- 左下 (Bottom Left) : コードの実行・実行結果などを表示するコンソール (Console) , コンピュータシステムの端末 (Terminal) , 文書の機械語翻訳 (Render) , 背後での作業 (Background Jobs)
- 右下 (Bottom Right) : ファイル (Files) , 描画 (Plots) , パッケージ (Packages) , ヘルプ (Help) , 文書などの表示窓 (Viewer) , R Markdown の HTML, PDF 表示 (Presentation^{*1})

^{*1} 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

6.4 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。

6.4.1 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。
 - 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。

6.4.2 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。

6.4.2.1 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: <-, (=  
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, _ and .

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))
```

6.4.2.2 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/csv/coronavirus.csv")
# 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 recovery

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("Confirmed Cases in", COUNTRY),
     xlab = "Date", ylab = "Number of Cases")
```

6.4.3 練習

上の、`coronavirus.R` について

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

6.4.4 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。

6.5 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。

6.5.1 パッケージのインストール

いずれ使いますので、まずは、三つのパッケージをインストールしてみましょう。

- `tidyverse`
- `rmarkdown`
- `tinytex`

インストール方法はいくつかあります。

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

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

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

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

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

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

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

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

6.5.2 備考

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

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

6.6 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。

6.6.1 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けません。

6.7 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、たくさんの方が、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、できていませんが、参考になる部分もあるかと思うので、紹介しておきます。

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

6.8 YouTube Video - getstarted

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

第 III 部

PART III INSTITUTIONAL DATA

第 7 章

World Bank

7.1 World Development Indicator (WDI)

パッケージと tidyverse と WDI を使いますから、下のコードによって、ロードします。

```
library(tidyverse)
#> -- Attaching core tidyverse packages ---- tidyverse 2.0.0 --
#> v dplyr      1.1.2      v readr      2.1.4
#> v forcats    1.0.0      v stringr    1.5.0
#> v ggplot2     3.4.2      v tibble     3.2.1
#> v lubridate  1.9.2      v tidyr      1.3.0
#> v purrr      1.0.1
#> -- Conflicts ----- tidyverse_conflicts() --
#> x dplyr::filter() masks stats::filter()
#> x dplyr::lag()    masks stats::lag()
#> i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
library(WDI)
```

まず、三つの例を見てみましょう。なにをしているかわかりますか。考えてみてください。

```
WDI(country = "all", indicator = c(gdp = "NY.GDP.MKTP.CD"),
     extra=TRUE) %>% drop_na(gdp) %>%
  filter(year==max(year), income != "Aggregates") %>%
  drop_na(region) %>% arrange(desc(gdp))
```

```
#> Rows: 16492 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr   (7): country, iso2c, iso3c, region, capital, income...
#> dbl   (4): year, gdp, longitude, latitude
#> lgl   (1): status
#> date  (1): lastupdated
```

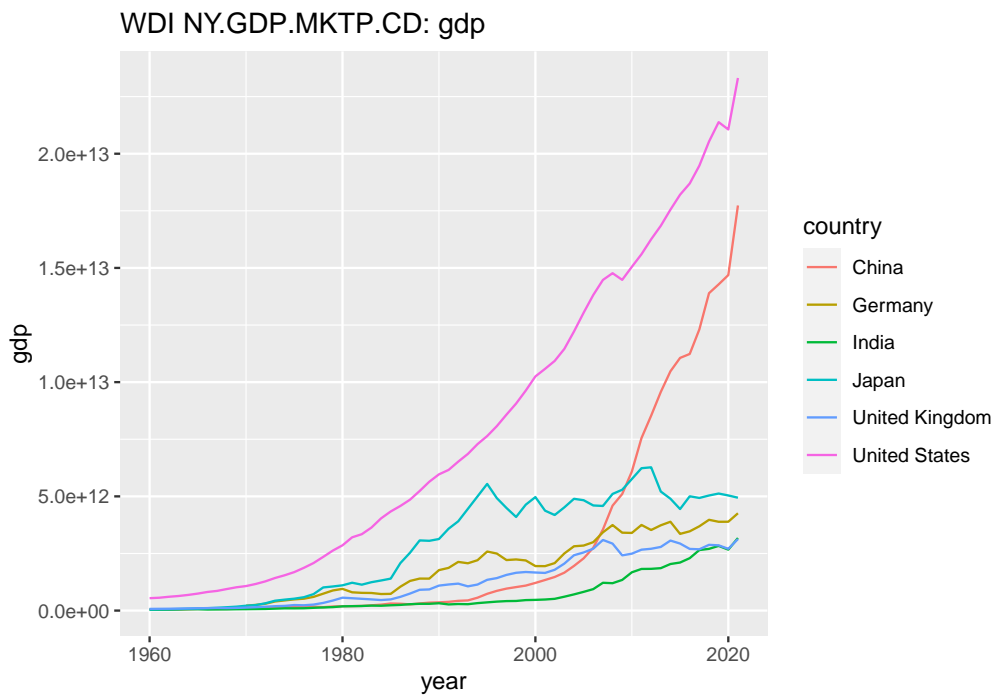
```

#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa
#> # A tibble: 196 x 13
#>   country      iso2c iso3c  year      gdp status lastupdated
#>   <chr>        <chr> <chr> <dbl>    <dbl> <lgl>   <date>
#> 1 United Stat~ US     USA   2021  2.33e13 NA     2022-12-22
#> 2 China       CN     CHN   2021  1.77e13 NA     2022-12-22
#> 3 Japan       JP     JPN   2021  4.94e12 NA     2022-12-22
#> 4 Germany     DE     DEU   2021  4.26e12 NA     2022-12-22
#> 5 India       IN     IND   2021  3.18e12 NA     2022-12-22
#> 6 United King~ GB     GBR   2021  3.13e12 NA     2022-12-22
#> 7 France      FR     FRA   2021  2.96e12 NA     2022-12-22
#> 8 Italy       IT     ITA   2021  2.11e12 NA     2022-12-22
#> 9 Canada      CA     CAN   2021  1.99e12 NA     2022-12-22
#> 10 Korea, Rep. KR     KOR   2021  1.81e12 NA     2022-12-22
#> # i 186 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> #   longitude <dbl>, latitude <dbl>, income <chr>,
#> #   lending <chr>

WDI(country = c("CN", "GB", "JP", "IN", "US", "DE"), indicator = c(gdp = "NY.GDP.MKTP.
  ggplot(aes(year, gdp, col = country)) + geom_line() +
  labs(title = "WDI NY.GDP.MKTP.CD: gdp")

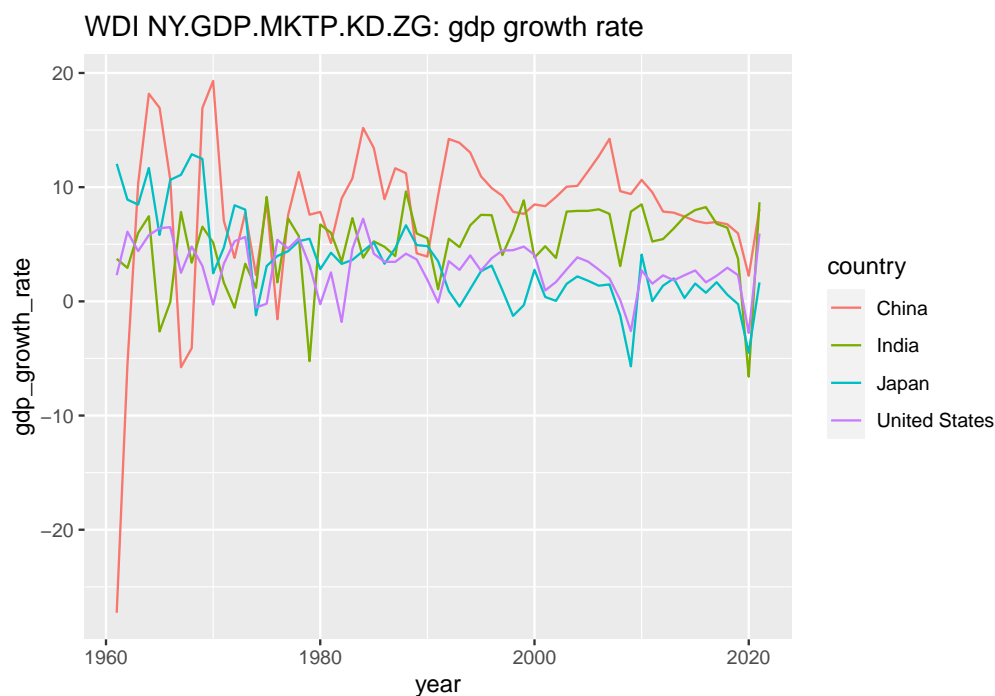
#> Rows: 372 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr  (7): country, iso2c, iso3c, region, capital, income...
#> dbl  (4): year, gdp, longitude, latitude
#> lgl  (1): status
#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa

```

```
WDI(country = c("CN", "IN", "JP", "US"),
     indicator = c(gdp_growth_rate = "NY.GDP.MKTP.KD.ZG"), extra=TRUE) %>%
  drop_na(gdp_growth_rate) %>%
  ggplot(aes(year, gdp_growth_rate, col = country)) + geom_line() +
  labs(title = paste("WDI NY.GDP.MKTP.KD.ZG: gdp growth rate"))
```

```
#> Rows: 248 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (4): year, gdp_growth_rate, longitude, latitude
#> lgl (1): status
#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```



まず、世界の国々の、GDP (gross domestic product 国内総生産) のデータを、取得して、2021 年の GDP を大きな順に並べています。

値は、たとえば、 $2.331508e + 13$ のように書かれていますが、これは、科学的記法と呼ばれるもので、 2.331508×10^{13} を意味しています。約 23 兆ドルです。

次に、3 兆ドル以上の、6 カ国を選択し、その、iso2c と呼ばれるコードを使って、それらの国のデータをもう一度取得し、年次変化をあらわすグラフを描いています。

さらにその中から、4 カ国を選んで、今度は、GDP の年次変化率を描いています。単位は、パーセントです。

これは、ひとつの例ですが、ここで使われているのが、WDI World Development Indicator というもので、世界銀行が、いくつかの指標を定めて、編纂しているものです。

7.1.1 指標 Indicators (WDI)

上の例では、次の二つの指標のコード Indicator Code (WDI Code) が使われました。

- NY.GDP.MKTP.CD: GDP (current US\$)
- NY.GDP.MKTP.KD.ZG: GDP growth (annual %)

7.1.2 指標 WDI (World Development Indicators)

The World Development Indicators is a compilation of relevant, high-quality, and internationally comparable statistics about global development and the fight against poverty. The database contains 1,400 time series indicators for

217 economies and more than 40 country groups, with data for many indicators going back more than 50 years.

WDI は、世界の開発状況と、貧困との戦いに関する、適切で上質、かつ、国際的に比較可能な時系列の統計データを編纂したものです。このデータベースは、217 の経済と 40 以上の国グループについて 1,400 の時系列指標を含み、指標のデータの多くは 50 年以上前に遡ることができます。

- 世界銀行 (World Bank) : <https://www.worldbank.org>
- World Bank Open Data: <https://data.worldbank.org>
 - Country / Indicator > Featured & All > Details
- World Development Indicators (WDI) :
 - Themes: Poverty and Inequality, People, Environment, Economy, States and Markets, Global Links
 - Open Data & DataBank: Explore data, Query database

7.1.3 指標のコード、WDI code を探してみよう

いくつかの探し方があります。まず、ここでは、World Bank のサイトから探す方法を説明しましょう。

ふた通りあります。

1. World Bank Open Data にいくと、表題の下を検索窓の下に、Country / Indicator とありますから、Indicator を選択します。すると、そこに、項目のリストが、Featured と All という二つのタブに分かれて出ています。かなり膨大です。それを選択すると、その項目のサイトに行きます。それが、指標のサイトです。図などの、右上に、Details とありますから、それを選択すると、その中に、Indicator が書かれています。実は、指標のサイトのアドレス (URL) を見ると、そこにも、この Indicator が書かれていることがわかります。
2. World Development Indicators (WDI) にいくと、下のようなテーマに分かれています。

Themes: Poverty and Inequality, People, Environment, Economy, States and Markets, Global Links

その中から、選択して、スクロールすると、そこに、指標が書かれています。

Indicator, Code, Time coverage, Region coverage, Get data

とあり、Code が、指標のコードです。実は、すべての年や、すべての地域のデータが揃っているわけではないので、この情報を見しておくことはとても重要です。ほとんど、データがない場合もあります。

一番右端の Get data からは、CSV や、データバンク (Data Bank) へのリンクがあり

ます。

それぞれの方法で、上で使った、二つの指標およびそのコードは見つかりましたか。

1の方法の途中に出てきた、検索窓から検索することも可能です。

7.1.4 指標 WDI の例

このあとの、例で使う指標を書いておきます。

- NY.GDP.MKTP.CD: GDP (current US\$)
- NY.GDP.DEFL.KD.ZG: Inflation, GDP deflator (annual %)
- SL.UEM.TOTL.NE.ZS: Unemployment, total (% of total labor force) (national estimate)
- CPTOTNSXN: CPI Price, nominal
- SL.TLF.CACT.MA.NE.ZS: Labor force participation rate, male (% of male population ages 15+) (national estimate)
- SL.TLF.CACT.FE.NE.ZS: Labor force participation rate, female (% of male population ages 15+) (national estimate)

7.1.5 練習 1. - 調べてみたい WDI 指標とそのコード

いくつか、リストしてみましょう。

7.2 WDI パッケージ

WDI パッケージの使い方を紹介します。

WDI パッケージで、データをダウンロードしたり、探したり、詳細情報を得たりできます。

7.2.1 指標 WDI 検索

まず、検索です。上で、サイトから調べる方法を紹介しましたが、WDI パッケージの、WDIsearch でも探すことができます。詳細は、右下の窓枠の Help タブの検索窓に、WDIsearch といれて調べて見てください。ここでは、二種類の検索方法を紹介します。

7.2.1.1 検索例 1 (WDI 名)

WDI 名に、ある文字列が含まれているものを検索します。検索文字列は、大文字・小文字は関係ありません。

```
WDIsearch(string = "gdp", field = "name", short = TRUE, cache = NULL) %>%
  as_tibble()
#> # A tibble: 540 x 2
```

```
#>   indicator      name
#>   <chr>         <chr>
#> 1 5.51.01.10.gdp  "Per capita GDP growth"
#> 2 6.0.GDP_current "GDP (current $)"
#> 3 6.0.GDP_growth  "GDP growth (annual %)"
#> 4 6.0.GDP_usd     "GDP (constant 2005 $)"
#> 5 6.0.GDPpc_constant "GDP per capita, PPP (constant 2011~
#> 6 BG.GSR.NFSV.GD.ZS "Trade in services (% of GDP)"
#> 7 BG.KAC.FNEI.GD.PP.ZS "Gross private capital flows (% of ~
#> 8 BG.KAC.FNEI.GD.ZS   "Gross private capital flows (% of ~
#> 9 BG.KLT.DINV.GD.PP.ZS "Gross foreign direct investment (%~
#> 10 BG.KLT.DINV.GD.ZS  "Gross foreign direct investment (%~
#> # ... with 530 more rows
```

なんと、500 件以上出てきました。Indicator（指標コード）と、Name（指標名）が列挙されます。すべてに、GDP という文字列が入っていることを確認できると思います。

7.2.1.2 検索例 2 (WDI)

Indicator（指標コード）から、Name（指標名）を検索します。

```
WDIsearch(string = "NY.GDP.MKTP.CD", field = "indicator", short = TRUE, cache = NULL)
#>           indicator
#> 11410    NY.GDP.MKTP.CD
#> 11411 NY.GDP.MKTP.CD.XD
#>
#>                                     name
#> 11410                                GDP (current US$)
#> 11411 GDP deflator, index (2000=100; US$ series)
```

二件出てきました。

7.2.1.3 練習 2. - 検索 (short)

名前で検索（“ ” の間に、（なるべく簡単な）検索文字列を入れてください。）

```
WDIsearch(string = "", field = "name", short = TRUE, cache = NULL)
```

Indicator で検索（“ ” の間に、調べたい indicator を入れてください。）

```
WDIsearch(string = "", field = "indicator", short = TRUE, cache = NULL)
```

7.2.1.4 詳しい情報を得るには

上では、Indicator（指標コード）と、Name（指標名）だけでしたが、Description（説明）なども得ることができます。

それには、`short = FALSE` とします。

一回一回、World Bank にアクセスして調べるのは、時間もかかりますから、Indicator と、名前などの情報をもったファイルを手元に持っておくことにします。それには、次のようにします。

```
wdi_cache <- WDIcache()
```

これは、`series` と、`country` の二つのデータ・フレームからなっているリストです。

右上の窓枠 (pane) から、`wdi_cache` を探して、中身を見てみましょう。三角印や、右から二番目の巻物のようなアイコンをクリックすると中身が見えます。

`series` には、すべての指標がリストされ、その情報が書かれています。

また、`country` には、それぞれについて、さまざまな情報が書かれています。これは、とても、たいせつな情報です。国名と、`iso2c`, `iso3c` のようなコードもありますし、地域 (region) や、その国が、どの income level (収入の階級) に入るかも書かれています。また、国だけではなく、地域など、グループの名称も含まれています。

今後、さまざまに利用していきたいと思います。

7.2.1.5 検索例 3 (WDI 名)

`short = FALSE` として、検索してみましょう。文字列が入っている、指標名を検索します。

```
WDIsearch(string = "CPI Price", field = "name", short = FALSE, cache = wdi_cache)

#>      indicator
#> 2586    CPTOTNSXN
#> 2587 CPTOTSAXMZGY
#> 2588    CPTOTSAXN
#> 2589 CPTOTSAXNZGY
#>
#>                                     name
#> 2586                               CPI Price, nominal
#> 2587 CPI Price, % y-o-y, median weighted, seas. adj.
#> 2588                               CPI Price, nominal, seas. adj.
#> 2589 CPI Price, % y-o-y, nominal, seas. adj.
#>
#> 2586                                     The consum
#> 2587                                     Median inj
#> 2588                                     The consumer price index re
#> 2589 The consumer price index reflects the change in prices for the average co
#>
#>      sourceDatabase
#> 2586 Global Economic Monitor
#> 2587 Global Economic Monitor
#> 2588 Global Economic Monitor
```

```
#> 2589 Global Economic Monitor
#>                                     sourceOrganization
#> 2586 World Bank staff calculations based on Datastream data.
#> 2587 World Bank staff calculations based on Datastream data.
#> 2588 World Bank staff calculations based on Datastream data.
#> 2589 World Bank staff calculations based on Datastream data.
```

- CPTOTNSXN: CPI Price, nominal
 - The consumer price index reflects the change in prices for the average consumer of a constant basket of consumer goods. Data is not seasonally adjusted.

7.2.1.6 検索例 4 (WDI)

指標コードから、詳細情報を得ます。

```
WDIsearch(string = "NY.GDP.MKTP.KD.ZG", field = "indicator", short = FALSE, cache = wdi_cache)
#>               indicator               name
#> 12114 NY.GDP.MKTP.KD.ZG GDP growth (annual %)
#>
#> 12114 Annual percentage growth rate of GDP at market prices based on constant local currency. Aggrega
#>               sourceDatabase
#> 12114 World Development Indicators
#>
#>                                     sourceOrganization
#> 12114 World Bank national accounts data, and OECD National Accounts data files.
```

7.2.1.7 練習 2 - 検索 (long w/ cache)

string と、field を、ふたつとも入れてください。

```
WDIsearch(string = "", field = "", short = FALSE, cache = wdi_cache)
```

7.2.2 指標 WDI データのダウンロード

Indicator が決まったら、ダウンロードします。右下の窓枠の Help タブの検索枠に、WDI と入れて確認しましょう。

```
WDI(
  country = "all",
  indicator = "NY.GDP.PCAP.KD",
  start = 1960,
  end = NULL,
  extra = FALSE,
  cache = NULL,
```

```

    latest = NULL,
    language = "en"
)

```

上が基本的な用法ですが、`start` 以下は、Default（初期値）が書かれていますから、たいせつなのは、最初の二つ、`country` と、`indicator` です。

7.2.2.1 ダウンロード例 1-1

`country` は、初期値も、“all” となっていますから、最も簡単なのは、`indicator` に、指標コードを入れることです。引用符を忘れずに。

```
df_gdp1 <- WDI(country = "all", indicator = "NY.GDP.MKTP.CD")
df_gdp1
```

```

#> Rows: 16492 Columns: 5
#> -- Column specification -----
#> Delimiter: ","
#> chr (3): country, iso2c, iso3c
#> dbl (2): year, NY.GDP.MKTP.CD
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message
#> # A tibble: 16,492 x 5
#>   country          iso2c iso3c  year NY.GDP.MKTP.CD
#>   <chr>          <chr> <chr> <dbl>         <dbl>
#> 1 Africa Eastern and Sout~ ZH   AFE   2021      1.08e12
#> 2 Africa Eastern and Sout~ ZH   AFE   2020      9.27e11
#> 3 Africa Eastern and Sout~ ZH   AFE   2019      1.00e12
#> 4 Africa Eastern and Sout~ ZH   AFE   2018      1.01e12
#> 5 Africa Eastern and Sout~ ZH   AFE   2017      1.02e12
#> 6 Africa Eastern and Sout~ ZH   AFE   2016      8.83e11
#> 7 Africa Eastern and Sout~ ZH   AFE   2015      9.25e11
#> 8 Africa Eastern and Sout~ ZH   AFE   2014      1.00e12
#> 9 Africa Eastern and Sout~ ZH   AFE   2013      9.83e11
#> 10 Africa Eastern and Sout~ ZH   AFE   2012      9.73e11
#> # i 16,482 more rows

```

これでも良いのですが、利用するには、指標コードではわかりにくいので、それを簡単な名前に置き換えて、データを読み込むことができます。

7.2.2.2 ダウンロード例 1-2


```
df_gdp2 <- WDI(country = "all", indicator = c(gdp = "NY.GDP.MKTP.CD"))
df_gdp2

#> Rows: 16492 Columns: 5
#> -- Column specification -----
#> Delimiter: ","
#> chr (3): country, iso2c, iso3c
#> dbl (2): year, gdp
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
#> # A tibble: 16,492 x 5
#>   country          iso2c iso3c  year    gdp
#>   <chr>          <chr> <chr> <dbl>  <dbl>
#> 1 Africa Eastern and Southern ZH    AFE    2021 1.08e12
#> 2 Africa Eastern and Southern ZH    AFE    2020 9.27e11
#> 3 Africa Eastern and Southern ZH    AFE    2019 1.00e12
#> 4 Africa Eastern and Southern ZH    AFE    2018 1.01e12
#> 5 Africa Eastern and Southern ZH    AFE    2017 1.02e12
#> 6 Africa Eastern and Southern ZH    AFE    2016 8.83e11
#> 7 Africa Eastern and Southern ZH    AFE    2015 9.25e11
#> 8 Africa Eastern and Southern ZH    AFE    2014 1.00e12
#> 9 Africa Eastern and Southern ZH    AFE    2013 9.83e11
#> 10 Africa Eastern and Southern ZH    AFE    2012 9.73e11
#> # i 16,482 more rows
```

7.2.2.3 ダウンロード例 1-3

今度は、`extra = TRUE` として、読み込みましょう。先ほど、読み込んだである、`wdi_cache` を使います。

```
df_gdp3 <- WDI(country = "all", indicator = c(gdp = "NY.GDP.MKTP.CD"),
              extra=TRUE, cache=wdi_cache)
df_gdp3
```

```
#> Rows: 16492 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr (7): country, iso2c, iso3c, region, capital, income...
#> dbl (4): year, gdp, longitude, latitude
#> lgl (1): status
```

```

#> date (1): lastupdated
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa
#> # A tibble: 16,492 x 13
#>   country      iso2c iso3c  year      gdp status lastupdated
#>   <chr>      <chr> <chr> <dbl>    <dbl> <lg1> <date>
#> 1 Afghanistan AF    AFG   2021  1.48e10 NA    2022-12-22
#> 2 Afghanistan AF    AFG   2020  2.01e10 NA    2022-12-22
#> 3 Afghanistan AF    AFG   2019  1.89e10 NA    2022-12-22
#> 4 Afghanistan AF    AFG   2018  1.84e10 NA    2022-12-22
#> 5 Afghanistan AF    AFG   2017  1.89e10 NA    2022-12-22
#> 6 Afghanistan AF    AFG   2016  1.80e10 NA    2022-12-22
#> 7 Afghanistan AF    AFG   2015  2.00e10 NA    2022-12-22
#> 8 Afghanistan AF    AFG   2014  2.06e10 NA    2022-12-22
#> 9 Afghanistan AF    AFG   2013  2.06e10 NA    2022-12-22
#> 10 Afghanistan AF    AFG   2012  2.02e10 NA    2022-12-22
#> # i 16,482 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> #   longitude <dbl>, latitude <dbl>, income <chr>,
#> #   lending <chr>

```

右上の三角印を使って、どのような詳細情報が付加されたか見てみましょう。どんなことがわかりますか。

7.2.2.4 ダウンロード例 1-4

国名を指定します。WDI では、iso2c コードを使って、国名を指定します。上で見たように、Environment から、wdi_cache を選択すると、国名と、iso2c コード両方を見ることができました。iso2c や、iso3c は、よく使われるので、web 検索でも簡単にみつかることができます。最初に紹介した例ですから、どの国かはわかりますね、

```

df_gdp4 <- WDI(country = c("CN", "GB", "JP", "IN", "US", "DE"),
               indicator = c(gdp = "NY.GDP.MKTP.CD"), extra=TRUE, cache=wdi_cache)
df_gdp4

```

```

#> Rows: 372 Columns: 13
#> -- Column specification -----
#> Delimiter: ","
#> chr  (7): country, iso2c, iso3c, region, capital, income...
#> dbl  (4): year, gdp, longitude, latitude
#> lg1  (1): status
#> date (1): lastupdated

```

```
#>
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this message.
#> # A tibble: 372 x 13
#>   country iso2c iso3c year      gdp status lastupdated
#>   <chr>   <chr> <chr> <dbl>   <dbl> <lgl>   <date>
#> 1 China   CN     CHN   2021 1.77e13 NA    2022-12-22
#> 2 China   CN     CHN   2020 1.47e13 NA    2022-12-22
#> 3 China   CN     CHN   2019 1.43e13 NA    2022-12-22
#> 4 China   CN     CHN   2018 1.39e13 NA    2022-12-22
#> 5 China   CN     CHN   2017 1.23e13 NA    2022-12-22
#> 6 China   CN     CHN   2016 1.12e13 NA    2022-12-22
#> 7 China   CN     CHN   2015 1.11e13 NA    2022-12-22
#> 8 China   CN     CHN   2014 1.05e13 NA    2022-12-22
#> 9 China   CN     CHN   2013 9.57e12 NA    2022-12-22
#> 10 China  CN     CHN   2012 8.53e12 NA    2022-12-22
#> # i 362 more rows
#> # i 6 more variables: region <chr>, capital <chr>,
#> #   longitude <dbl>, latitude <dbl>, income <chr>,
#> #   lending <chr>
```

7.2.2.5 ダウンロード例 2-1

二つの、指標コードを使って、同時に読み込むこともできます。そのときは、`c()` (combine) を使います。

- NY.GDP.DEFL.KD.ZG: Inflation, GDP deflator (annual %)
- CPTOTNSXN: CPI Price, nominal

```
df_gdp21 <- WDI(country = "all",
  indicator = c(gdp_deflator = "NY.GDP.DEFL.KD.ZG",
    cpi_price = "CPTOTNSXN"),
  extra=TRUE, cache=wdi_cache)
df_gdp21
```

```
#> Rows: 23972 Columns: 14
#> -- Column specification -----
#> Delimiter: ","
#> chr  (7): country, iso2c, iso3c, region, capital, income...
#> dbl  (5): year, gdp_deflator, cpi_price, longitude, lati...
#> lgl  (1): status
#> date (1): lastupdated
#>
```

```
#> i Use `spec()` to retrieve the full column specification for this data.
#> i Specify the column types or set `show_col_types = FALSE` to quiet this messa
#> # A tibble: 23,972 x 14
#>   country iso2c iso3c year status lastupdated gdp_deflator
#>   <chr>   <chr> <chr> <dbl> <lgl>   <date>             <dbl>
#> 1 Advanc~ AME   <NA>  1987 NA     2020-07-27         NA
#> 2 Advanc~ AME   <NA>  1988 NA     2020-07-27         NA
#> 3 Advanc~ AME   <NA>  1989 NA     2020-07-27         NA
#> 4 Advanc~ AME   <NA>  1990 NA     2020-07-27         NA
#> 5 Advanc~ AME   <NA>  1991 NA     2020-07-27         NA
#> 6 Advanc~ AME   <NA>  1992 NA     2020-07-27         NA
#> 7 Advanc~ AME   <NA>  1993 NA     2020-07-27         NA
#> 8 Advanc~ AME   <NA>  1994 NA     2020-07-27         NA
#> 9 Advanc~ AME   <NA>  1995 NA     2020-07-27         NA
#> 10 Advanc~ AME   <NA>  1996 NA     2020-07-27         NA
#> # i 23,962 more rows
#> # i 7 more variables: cpi_price <dbl>, region <chr>,
#> #   capital <chr>, longitude <dbl>, latitude <dbl>,
#> #   income <chr>, lending <chr>
```

NA (not available) つまり、データがないものが多いことがわかります。もう少し、データをよく見て見ましょう。

```
str(df_gdp21)
#> spec_tbl_ [23,972 x 14] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
#> $ country      : chr [1:23972] "Advanced Economies" "Advanced Economies" "Advanc
#> $ iso2c         : chr [1:23972] "AME" "AME" "AME" "AME" ...
#> $ iso3c         : chr [1:23972] NA NA NA NA ...
#> $ year          : num [1:23972] 1987 1988 1989 1990 1991 ...
#> $ status        : logi [1:23972] NA NA NA NA NA NA ...
#> $ lastupdated   : Date[1:23972], format: "2020-07-27" ...
#> $ gdp_deflator: num [1:23972] NA NA NA NA NA NA NA NA NA ...
#> $ cpi_price     : num [1:23972] 58.7 60.5 63 66 69.1 ...
#> $ region        : chr [1:23972] NA NA NA NA ...
#> $ capital       : chr [1:23972] NA NA NA NA ...
#> $ longitude     : num [1:23972] NA NA NA NA NA NA NA NA NA ...
#> $ latitude      : num [1:23972] NA NA NA NA NA NA NA NA NA ...
#> $ income        : chr [1:23972] NA NA NA NA ...
#> $ lending       : chr [1:23972] NA NA NA NA ...
#> - attr(*, "spec")=
#> .. cols(
#> ..   country = col_character(),
#> ..   iso2c = col_character(),
```

```
#> .. iso3c = col_character(),
#> .. year = col_double(),
#> .. status = col_logical(),
#> .. lastupdated = col_date(format = ""),
#> .. gdp_deflator = col_double(),
#> .. cpi_price = col_double(),
#> .. region = col_character(),
#> .. capital = col_character(),
#> .. longitude = col_double(),
#> .. latitude = col_double(),
#> .. income = col_character(),
#> .. lending = col_character()
#> .. )
#> - attr(*, "problems")=<externalptr>
```

```
summary(df_gdp21)
```

```
#>      country      iso2c      iso3c
#> Length:23972      Length:23972      Length:23972
#> Class :character  Class :character  Class :character
#> Mode  :character  Mode  :character  Mode  :character
#>
#>
#>
#>      year      status      lastupdated
#> Min.   :1960      Mode:logical      Min.   :2020-07-27
#> 1st Qu.:1982      NA's:23972      1st Qu.:2020-07-27
#> Median :1996                                     Median :2022-12-22
#> Mean   :1995                                     Mean   :2022-03-23
#> 3rd Qu.:2009                                     3rd Qu.:2022-12-22
#> Max.   :2021                                     Max.   :2022-12-22
#>
#>      gdp_deflator      cpi_price      region
#> Min.   : -98.704      Min.   : 0.00      Length:23972
#> 1st Qu.:  2.317      1st Qu.: 55.95      Class :character
#> Median :  5.273      Median : 83.28      Mode  :character
#> Mean   : 25.308      Mean   : 84.18
#> 3rd Qu.: 10.411      3rd Qu.:108.75
#> Max.   :26765.858      Max.   :551.25
#> NA's   :11616      NA's   :18410
#>      capital      longitude      latitude
```

```
#> Length:23972      Min.   :-175.22   Min.   :-41.286
#> Class :character  1st Qu.: -15.18   1st Qu.:  4.174
#> Mode  :character  Median :  19.26   Median : 17.300
#>                      Mean  :  19.14   Mean   : 18.889
#>                      3rd Qu.:  50.53   3rd Qu.: 40.050
#>                      Max.   : 179.09   Max.   : 64.184
#>                      NA's   :10890    NA's   :10890
#>      income      lending
#> Length:23972    Length:23972
#> Class :character Class :character
#> Mode  :character Mode  :character
#>
#>
#>
#>
```

どんなことが分かりましたか。

右上の窓枠の、Environment でも df_gdp21 を見てみましょう。

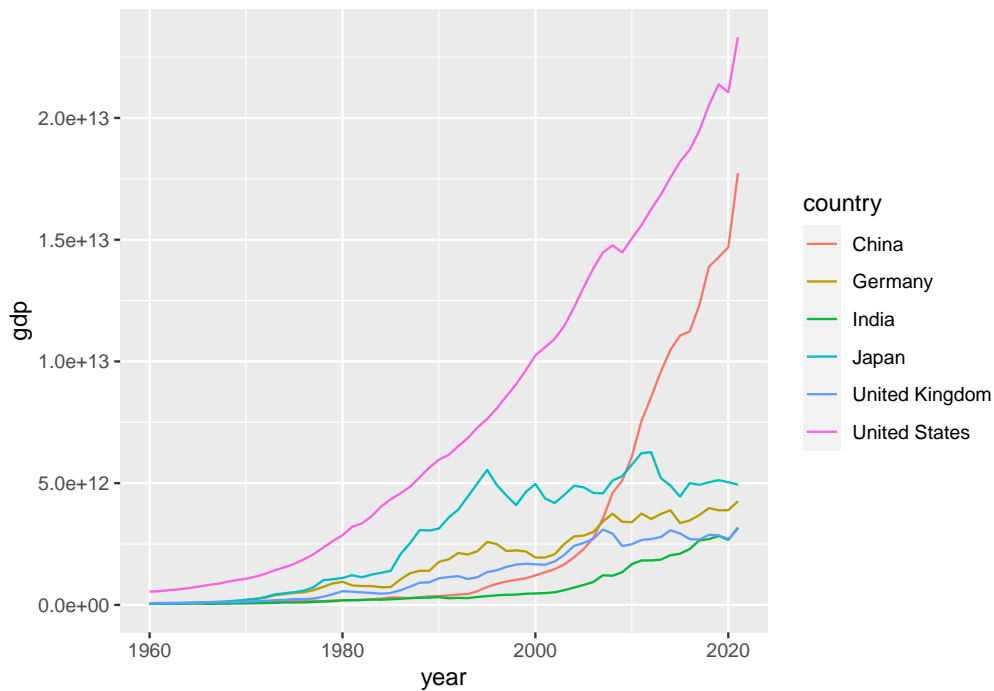
7.3 可視化 Visualization

グラフ (Chart) を描いて視覚化します。ここでは、年ごとの変化をみる、折れ線グラフだけを描いて見ます。

7.3.1 グラフ 1

x = year, y = gdp の、x=, y= は省略してあります。col=country は、国ごとに、グループにして、色分けをします。col は、color としても colour としても、問題ありません。‘

```
df_gdp4 %>% ggplot(aes(year, gdp, col=country)) + geom_line()
#> Warning: Removed 10 rows containing missing values
#> (`geom_line()`).
```

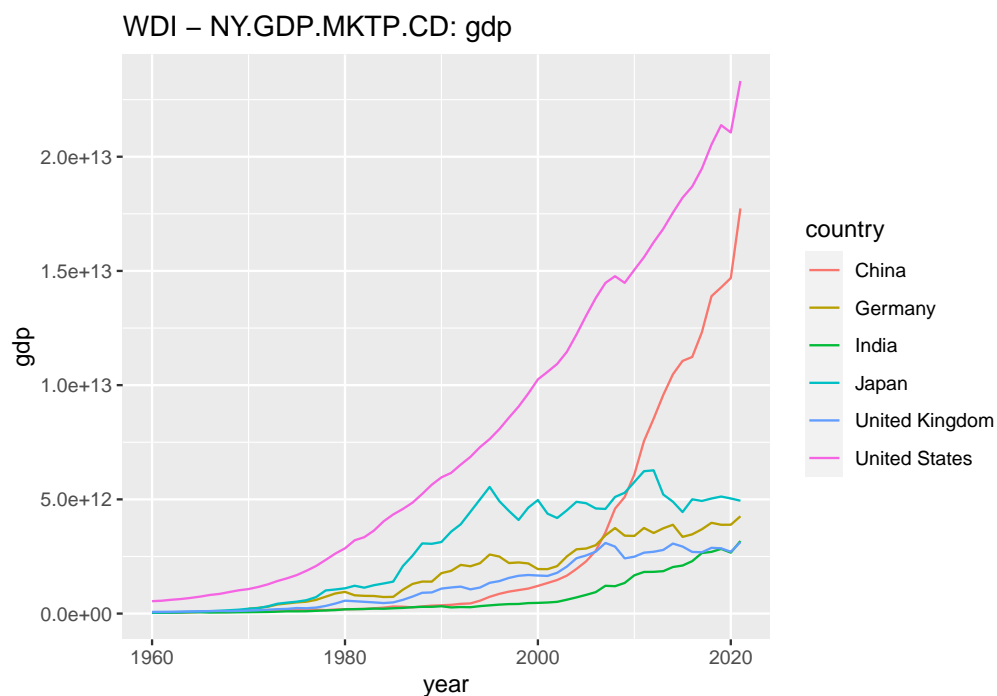


Warning として、missing values があると出ています。どこかは、分かりませんが、図を書くときですから、y に対応する、gdp の値がないものと思われる。

7.3.2 グラフ 2

`drop_na(gdp)` で、gdp の値が、NA であるものを削除します。また、`labs` で、図にタイトルをつけます。

```
df_gdp4 %>% drop_na(gdp) %>%  
  ggplot(aes(year, gdp, col=country)) + geom_line() +  
  labs(title = "WDI - NY.GDP.MKTP.CD: gdp")
```



7.3.3 テンプレート Templates

下に、テンプレートをつけます。コピーして、指標コードや、略称、国などを、それぞれ置き換えて、試して見てください。少し、複雑な変形をしていますが、少しずつ説明します。

7.3.3.1 一つの国についての、一つの指標（WDI）と、その略称から、折線グラフを作成

Line Plot with one indicator with abbreviation and one country

```
chosen_indicator <- "SL.UEM.TOTL.NE.ZS"
short_name <- "unemployment"
chosen_country <- "United States"
WDI(country = "all", indicator = c(short_name = chosen_indicator), extra=TRUE, cache=wdi_cache) %>%
  filter(country == chosen_country) %>%
  ggplot(aes(year, short_name)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator, ": ", short_name, " - ", chosen_country),
       y = short_name)
```

7.3.3.2 一つの国についての、一つの指標（WDI）から、折線グラフを作成

Line Plot with one indicator and one country

```
chosen_indicator <- "SL.UEM.TOTL.NE.ZS"
chosen_country <- "United States"
WDI(country = "all", indicator = c(chosen_indicator = chosen_indicator),
     extra=TRUE, cache=wdi_cache) %>%
```



```
filter(country == chosen_country) %>%
  ggplot(aes(year, chosen_indicator)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator, " - ", chosen_country),
       y = chosen_indicator)
```

7.3.3.3 いくつかの国についての、一つの指標（WDI）と、その略称から、折線グラフを作成

Line Plot with one indicator with abbreviation and several countries

```
chosen_indicator <- "SL.UEM.TOTL.NE.ZS"
short_name <- "unemployment"
chosen_countries <- c("United States", "United Kingdom", "Japan")
WDI(country = "all", indicator = c(short_name = chosen_indicator), extra=TRUE, cache=wdi_cache) %>% drop_na(value)
  filter(country %in% chosen_countries) %>%
  ggplot(aes(year, short_name, col = country)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator, ": ", short_name), y = short_name)
```

7.3.3.4 一つの国についての、二つの指標（WDI）と、その略称から、折線グラフを作成

Line Plot with two indicators with abbreviation and one country

```
chosen_indicator_1 <- "NY.GDP.DEFL.KD.ZG"
short_name_1 <- "gdp_deflator"
chosen_indicator_2 <- "CPTOTSAXNZGY"
short_name_2 <- "cpi_price"
chosen_country <- "United States"
WDI(country = "all", indicator = c(short_name_1 = chosen_indicator_1, short_name_2 = chosen_indicator_2))
  filter(country == chosen_country) %>%
  pivot_longer(c(short_name_1, short_name_2), names_to = "class", values_to = "value") %>% drop_na(value)
  ggplot(aes(year, value, col = class)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator_1, ": ", short_name_1, "\n", chosen_indicator_2, ": ", short_name_2),
       y = value)
  scale_color_manual(labels = c(short_name_1, short_name_2), values = scales::hue_pal()(2))
```

```
chosen_indicator_1 <- "SL.TLF.CACT.MA.NE.ZS"
short_name_1 <- "male"
chosen_indicator_2 <- "SL.TLF.CACT.FE.NE.ZS"
short_name_2 <- "female"
chosen_country <- "United States"
WDI(country = "all", indicator = c(short_name_1 = chosen_indicator_1, short_name_2 = chosen_indicator_2))
  filter(country == chosen_country) %>%
  pivot_longer(c(short_name_1, short_name_2), names_to = "class", values_to = "value") %>% drop_na(value)
  ggplot(aes(year, value, col = class)) + geom_line() +
```

```
labs(title = paste("WDI ", chosen_indicator_1, ": ", short_name_1, "\n", chosen
scale_color_manual(labels = c(short_name_1, short_name_2), values = scales::hue
```

7.3.3.5 いくつかの国についての、二つの指標（WDI）と、その略称から、折線グラフを作成

Line Plot with two indicators with abbreviation and several countries

```
chosen_indicator_1 <- "NY.GDP.DEFL.KD.ZG"
short_name_1 <- "gdp_deflator"
chosen_indicator_2 <- "CPTOTSAXNZGY"
short_name_2 <- "cpi_price"
chosen_countries <- c("United States", "France", "Japan")
WDI(country = "all", indicator = c(short_name_1 = chosen_indicator_1, short_name_
  filter(country %in% chosen_countries) %>%
  pivot_longer(c(short_name_1, short_name_2), names_to = "class", values_to = "va
  ggplot(aes(year, value, linetype = class, col = country)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator_1, ": ", short_name_1, "\n", chosen
  scale_linetype_manual(labels = c(short_name_1, short_name_2), values = c("solid
```

```
chosen_indicator_1 <- "SL.TLF.CACT.MA.NE.ZS"
short_name_1 <- "male"
chosen_indicator_2 <- "SL.TLF.CACT.FE.NE.ZS"
short_name_2 <- "female"
chosen_countries <- c("United States", "France", "Japan")
WDI(country = "all", indicator = c(short_name_1 = chosen_indicator_1, short_name_
  filter(country %in% chosen_countries) %>%
  pivot_longer(c(short_name_1, short_name_2), names_to = "class", values_to = "va
  ggplot(aes(year, value, linetype = class, col = country)) + geom_line() +
  labs(title = paste("WDI ", chosen_indicator_1, ": ", short_name_1, "\n", chosen
  scale_linetype_manual(labels = c(short_name_1, short_name_2), values = c("solid
```

7.4 課題 Assignment

上のテンプレートをコピーして、下に貼り付け、指標 `indicator` と、略称 `short_name` と、いくつかの国名 `chosen_countries` を、入れ替えて、試してみてください。

第 IV 部

PART IV EDA

第 8 章

探索的データ解析

8.1 探索的データ解析 (EDA) とは

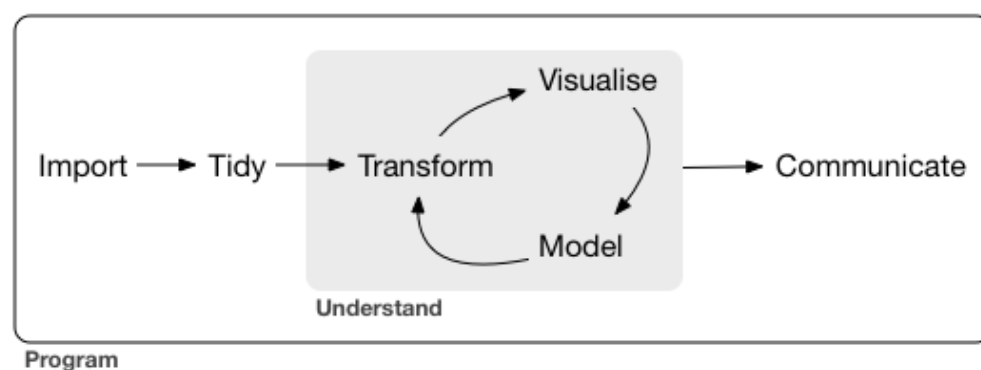


図8.1 image from r4ds

以下は、Posit Primers: Visualise Data から

探索的データ解析 (EDA) は、データを理解するための反復的なサイクルです。EDA では、以下のことを行います。

1. データに関する問いを作成する
2. データの可視化、変形・整形、モデリングによって、問いの答えを探索する。
3. 学んだことを使って、問いをより洗練されたものとする。

EDA は、あらゆるデータ分析において重要な役割を担います。EDA によって、課題解決のいとぐちを発見することもありますし、他の課題との関係性を発見する場合があります。EDA を使用してデータの問題や品質を確認したり、データが信頼できるものであるかを見極める問いを作成できる場合があります。

8.2 探索的データ解析 (EDA) の一例

WDI の一つの指標を使って、流れを見てみましょう。

8.2.1 データの取得と読み込み - Data Import

NY.GDP.PCAP.CD: GDP per capita (current US\$)

```
df_wdi_gdppcap <- WDI(country = "all", indicator = c(gdp_pcap = "NY.GDP.PCAP.CD"))
write_csv(df_wdi_gdppcap, "./data/df_wdi_gdppcap.csv")
```

```
df_wdi_gdppcap
#> # A tibble: 16,492 x 5
#>   country                iso2c iso3c year gdp_pcap
#>   <chr>                  <chr> <chr> <dbl>   <dbl>
#> 1 Africa Eastern and Southern ZH   AFE   2021   1550.
#> 2 Africa Eastern and Southern ZH   AFE   2020   1364.
#> 3 Africa Eastern and Southern ZH   AFE   2019   1512.
#> 4 Africa Eastern and Southern ZH   AFE   2018   1565.
#> 5 Africa Eastern and Southern ZH   AFE   2017   1629.
#> 6 Africa Eastern and Southern ZH   AFE   2016   1444.
#> 7 Africa Eastern and Southern ZH   AFE   2015   1539.
#> 8 Africa Eastern and Southern ZH   AFE   2014   1719.
#> 9 Africa Eastern and Southern ZH   AFE   2013   1730.
#> 10 Africa Eastern and Southern ZH   AFE   2012   1759.
#> # i 16,482 more rows
```

8.2.2 データ変形・整形 - Data Transformation

8.2.2.1 列を select

どの変数について分析するかを選ぶ。

```
df_wdi_gdppcap_small <- df_wdi_gdppcap %>%
  select(country, year, gdp_pcap)
df_wdi_gdppcap_small
#> # A tibble: 16,492 x 3
#>   country                year gdp_pcap
#>   <chr>                  <dbl>   <dbl>
#> 1 Africa Eastern and Southern 2021   1550.
#> 2 Africa Eastern and Southern 2020   1364.
#> 3 Africa Eastern and Southern 2019   1512.
#> 4 Africa Eastern and Southern 2018   1565.
#> 5 Africa Eastern and Southern 2017   1629.
#> 6 Africa Eastern and Southern 2016   1444.
#> 7 Africa Eastern and Southern 2015   1539.
#> 8 Africa Eastern and Southern 2014   1719.
```

```
#> 9 Africa Eastern and Southern 2013 1730.
#> 10 Africa Eastern and Southern 2012 1759.
#> # i 16,482 more rows
```

8.2.2.2 行を filter

いくつかの国に、フォーカスして調べる。

```
df_wdi_gdppcap_short <- df_wdi_gdppcap %>%
  filter(country %in% c("Japan", "Germany", "United States"))
df_wdi_gdppcap_short
#> # A tibble: 186 x 5
#>   country iso2c iso3c year gdp_pcap
#>   <chr>   <chr> <chr> <dbl>   <dbl>
#> 1 Germany DE    DEU    2021  51204.
#> 2 Germany DE    DEU    2020  46773.
#> 3 Germany DE    DEU    2019  46794.
#> 4 Germany DE    DEU    2018  47939.
#> 5 Germany DE    DEU    2017  44653.
#> 6 Germany DE    DEU    2016  42136.
#> 7 Germany DE    DEU    2015  41103.
#> 8 Germany DE    DEU    2014  48024.
#> 9 Germany DE    DEU    2013  46299.
#> 10 Germany DE    DEU    2012  43856.
#> # i 176 more rows
```

列（変数）と、行（国）の選択を続けて、実行すると次のようになる。一つ一つ変形したデータ（オブジェクト）に名前をつけて、保存する必要がないので、パイプ（%>%）の活用は有用である。

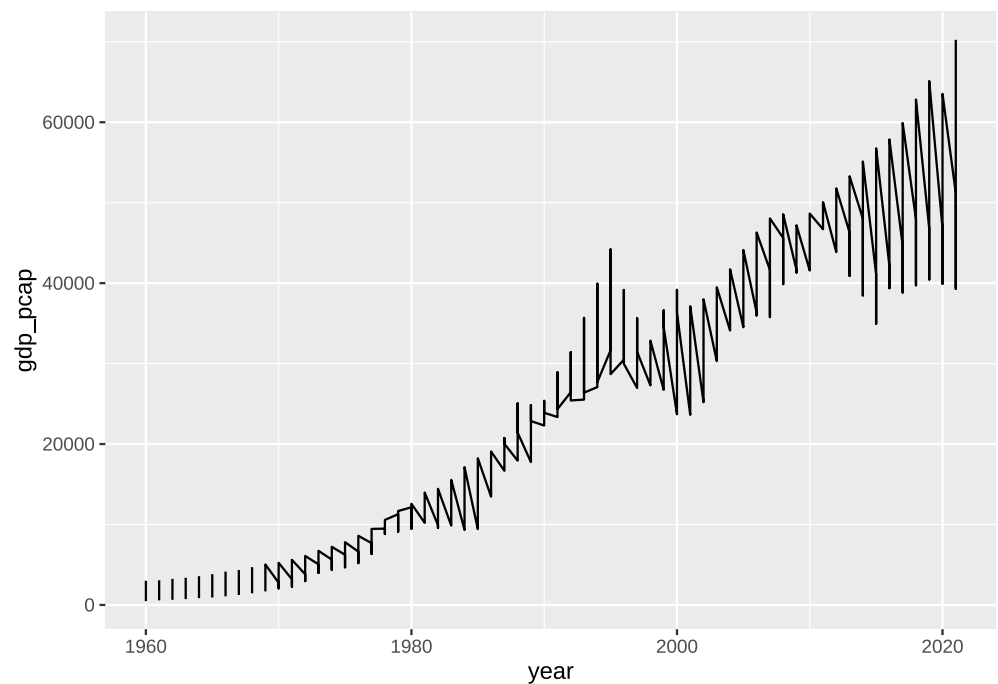
```
df_wdi_gdppcap_small_short <- df_wdi_gdppcap %>% select(country, year, gdp_pcap) %>%
  filter(country %in% c("Japan", "Germany", "United States"))
df_wdi_gdppcap_small_short
#> # A tibble: 186 x 3
#>   country year gdp_pcap
#>   <chr>   <dbl>   <dbl>
#> 1 Germany 2021  51204.
#> 2 Germany 2020  46773.
#> 3 Germany 2019  46794.
#> 4 Germany 2018  47939.
#> 5 Germany 2017  44653.
#> 6 Germany 2016  42136.
#> 7 Germany 2015  41103.
```

```
#> 8 Germany 2014 48024.
#> 9 Germany 2013 46299.
#> 10 Germany 2012 43856.
#> # i 176 more rows
```

8.2.3 可視化 Data Visualization

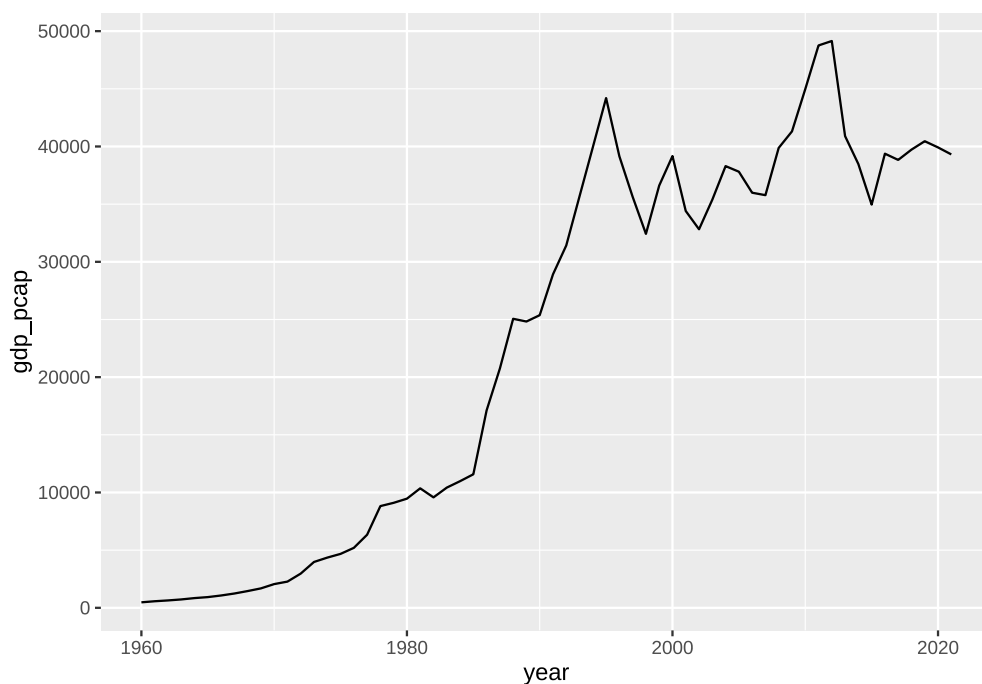
次は、よく生じる、誤りの例で、ノコギリの歯 (sawtoothed) のようなギザギザ・グラフと呼ばれます。なぜこのようなことが起きているかわかりますか。

```
df_wdi_gdppcap_small_short %>%
  ggplot(aes(x = year, y = gdp_pcap)) + geom_line()
#> Warning: Removed 1 row containing missing values
#> (`geom_line()`).
```



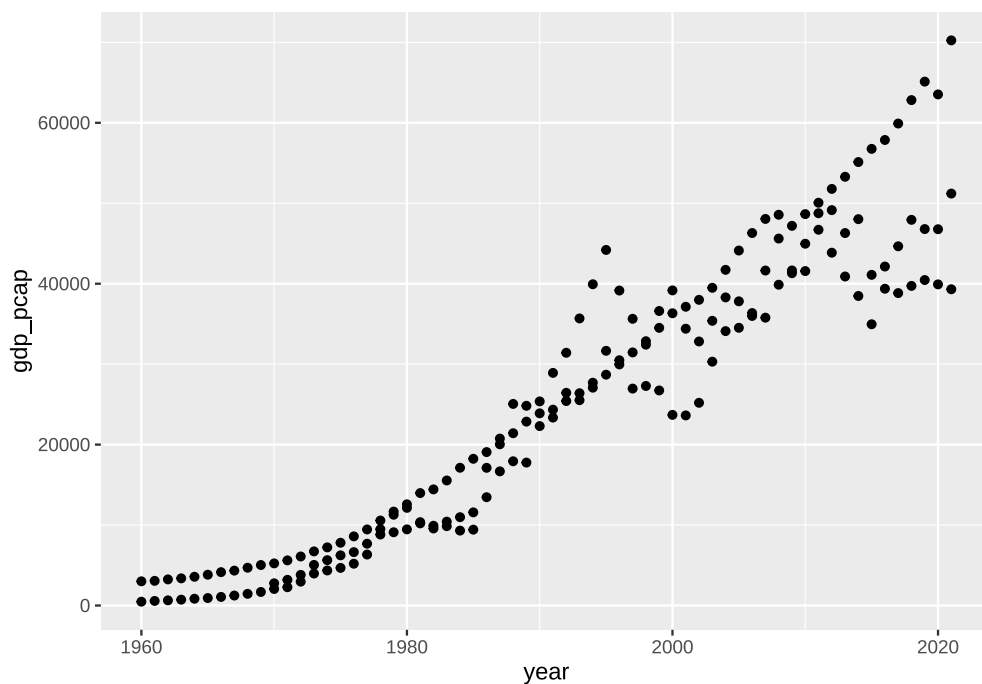
同じ年に、多くのデータがあるので、折れ線グラフを適切に書くことができませんでした。

```
df_wdi_gdppcap_small_short %>% filter(country %in% c("Japan")) %>%
  ggplot(aes(x = year, y = gdp_pcap)) + geom_line()
```

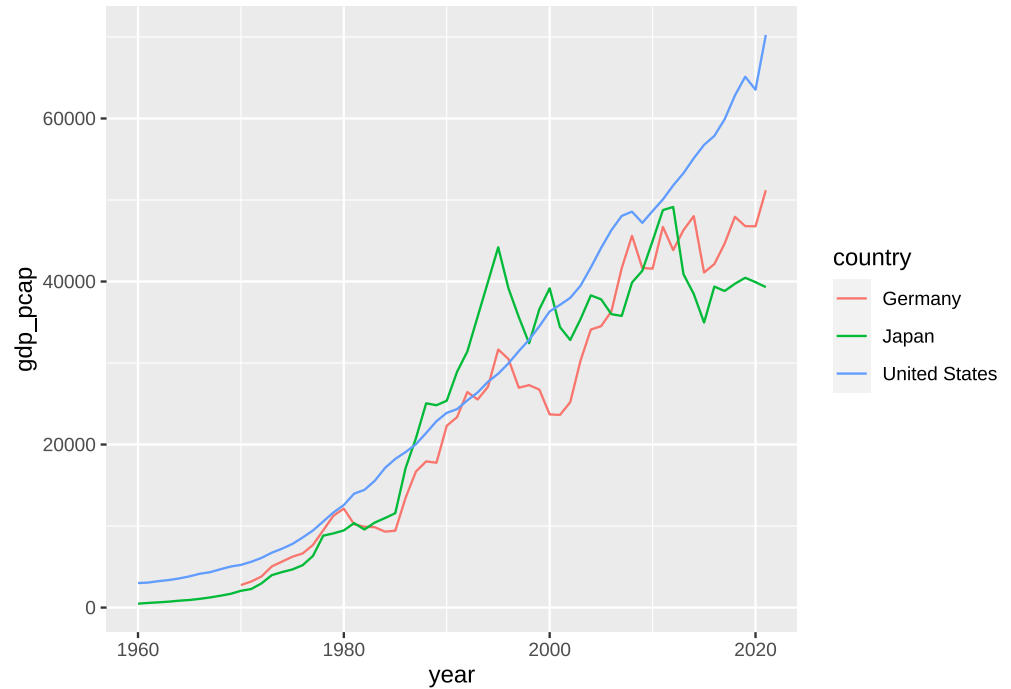
一般的には、散布図をまず、書いてみるのも一つです。

```
df_wdi_gdppcap_small_short %>%  
  ggplot(aes(x = year, y = gdp_pcap)) + geom_point()  
#> Warning: Removed 10 rows containing missing values  
#> (`geom_point()`).
```



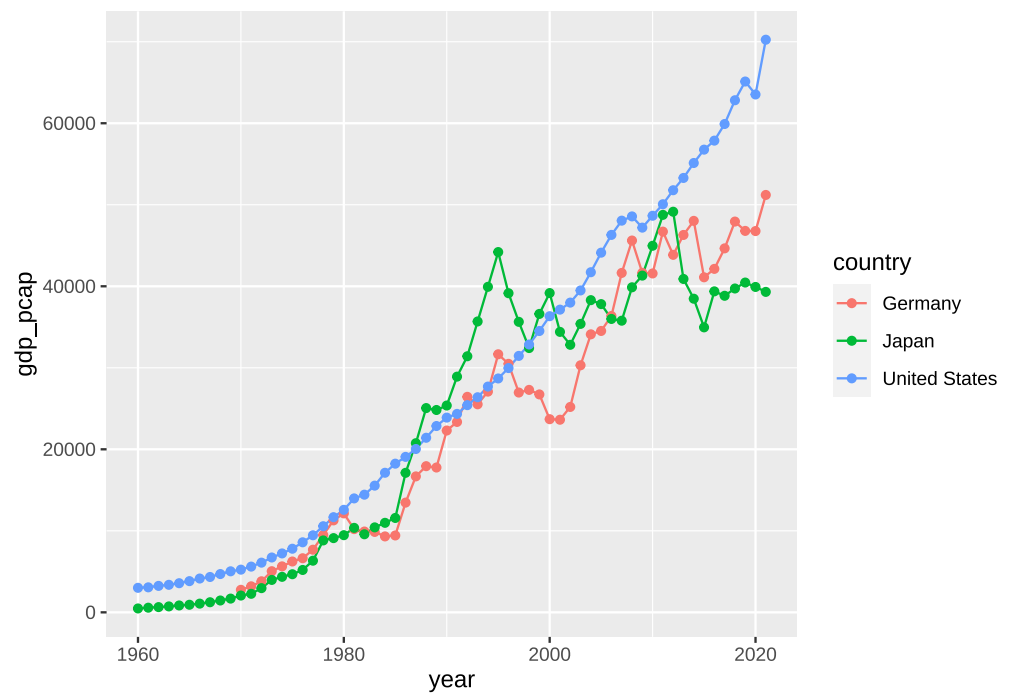
国別に、異なる色を使うことで、折れ線グラフを書くことも可能です。

```
df_wdi_gdppcap_small_short %>% drop_na(gdp_pcap) %>%
  ggplot(aes(x = year, y = gdp_pcap, col = country)) + geom_line()
```



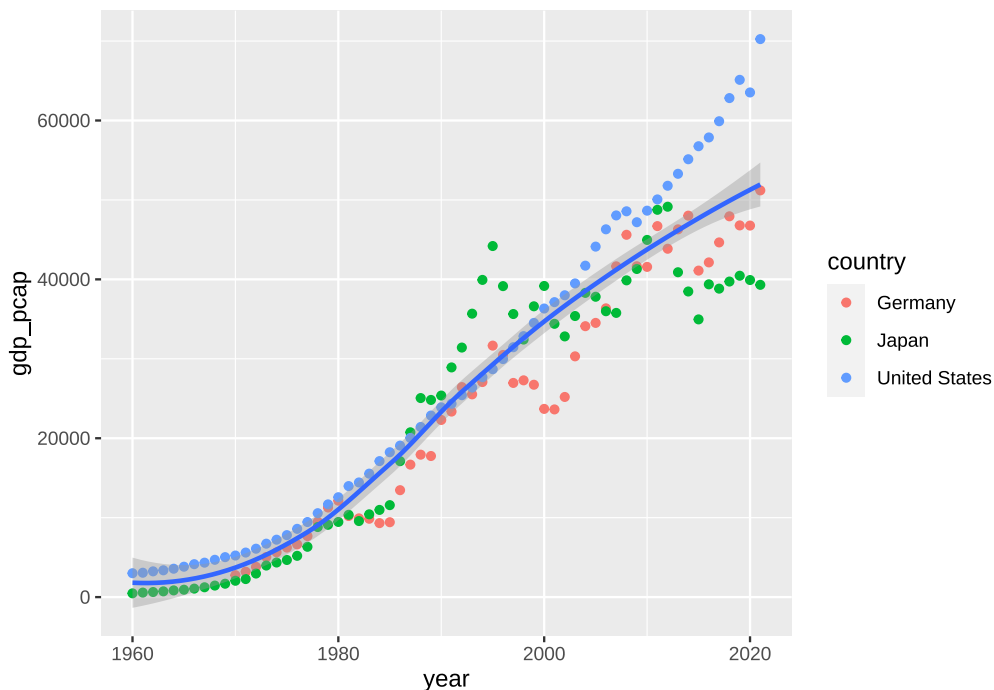
折線グラフと、散布図を同時に描くこともかこのうです。

```
df_wdi_gdppcap_small_short %>% drop_na(gdp_pcap) %>%
  ggplot(aes(x = year, y = gdp_pcap, col = country)) + geom_line() +
  geom_point()
```



点を、曲線で近似する方法はいくつも知られているが、ある幅で、近似していく、LOESS が初期値となっている。method='loess' を省略しても、同じ近似がなされる。span という値を調節することで、ことなる幅での近似曲線を書くことも可能である。初期値は、0.75。

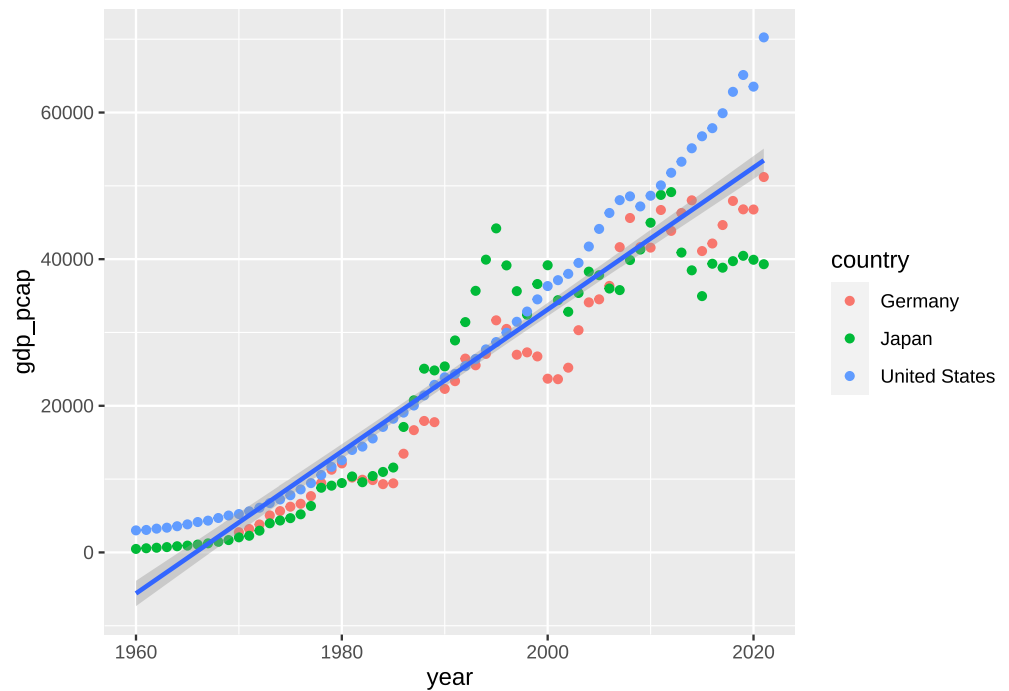
```
df_wdi_gdppcap_small_short %>% drop_na(gdp_pcap) %>%
  ggplot(aes(x = year, y = gdp_pcap)) +
  geom_point(aes(color = country)) +
  geom_smooth(method = 'loess', formula = 'y ~ x')
```



8.2.4 データモデリング Data Modeling

上の例では、曲線ではなく、直線で近似することも考えられる。

```
df_wdi_gdppcap_small_short %>% drop_na(gdp_pcap) %>%
  ggplot(aes(x = year, y = gdp_pcap)) +
  geom_point(aes(color = country)) +
  geom_smooth(method = 'lm', formula = 'y ~ x')
```



簡単な線形回帰モデルでの、回帰直線の y-切片や、傾きは、次のコードで与えられ、p-value や、R squared の値も求められる。

この例では、年とともに、増加の傾向があること。そして、線形モデルが \$\$、90% 程度説明していると表現される。すなわち、

は、良い、近似であることがわかる。

```
df_wdi_gdppcap_small_short %>% lm(gdp_pcap ~ year, .) %>% summary()
#>
#> Call:
#> lm(formula = gdp_pcap ~ year, data = .)
#>
#> Residuals:
#>      Min       1Q   Median       3Q      Max
#> -14156.8  -3200.5  -507.4   3237.7  16779.2
#>
#> Coefficients:
#>              Estimate Std. Error t value Pr(>|t|)
#> (Intercept) -1903497.5    48007.9   -39.65  <2e-16 ***
#> year          968.3         24.1    40.18  <2e-16 ***
#> ---
#> Signif. codes:
#>  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> Residual standard error: 5514 on 174 degrees of freedom
```

```
#> (10 observations deleted due to missingness)
#> Multiple R-squared:  0.9027, Adjusted R-squared:  0.9021
#> F-statistic: 1614 on 1 and 174 DF,  p-value: < 2.2e-16
```


第 V 部

PART V EXAMPLES

第 9 章

Example 1

付録 A

日本語の扱いについて

A.1 日本語・中国語・韓国語

文字化けが、起こることが多く、対応が、一定せず、特に、図の表示において、Windows や、macOS や、Linux などの、OS ごとに、フォントが違ったり、それを、図のタイトルなどに、使ったりが、難しかったのですが、どうやら、現在は、どの場合も、次の設定で、解決しているようです。下の例を確認してください。特に、フォントについては、好みも関係しますから、難しいですが、ここでは、どのプラットフォーム（OS）でも、共通に扱えることを中心に書きます。

```
# showtext を、インストールしていない場合は、一回だけ、右上の三角をクリックして実行
install.packages('showtext')
```

A.1.1 パッケージをロード

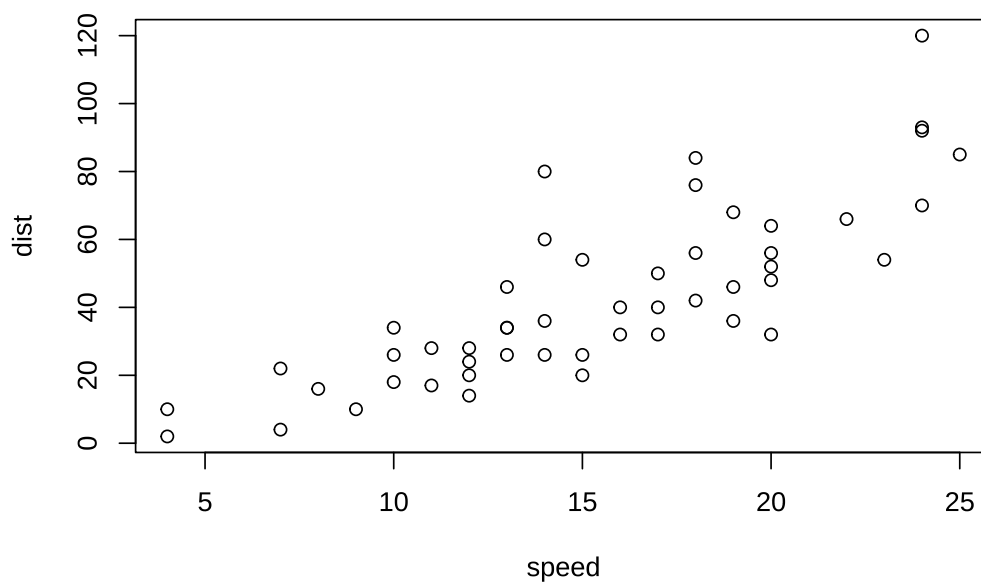
library によって、Package をロード（いつでも使えるように）します。

```
library(tidyverse)
library(showtext)
showtext_auto()
```

A.2 Base R でタイトルに日本語

```
plot(cars, main=" 散布図")
```

散布図



A.3 列名や、データに日本語

```
df_iris <- iris
colnames(df_iris) <- c(" 萼長", " 萼幅", " 葉長", " 葉幅", "Species" )
tab <- data.frame(Species = c("setosa", "versicolor", "virginica"),
                  " 種別" = c(" ヒオウギアヤメ", " ブルーフラッグ", " バージニカ"))
df_iris <- df_iris %>% left_join(tab, by=c("Species" = "Species")) %>% select(-5)
df_iris %>% slice(1:2)
#>   萼長 萼幅 葉長 葉幅      種別
#> 1  5.1  3.5  1.4  0.2 ヒオウギアヤメ
#> 2  4.9  3.0  1.4  0.2 ヒオウギアヤメ
```

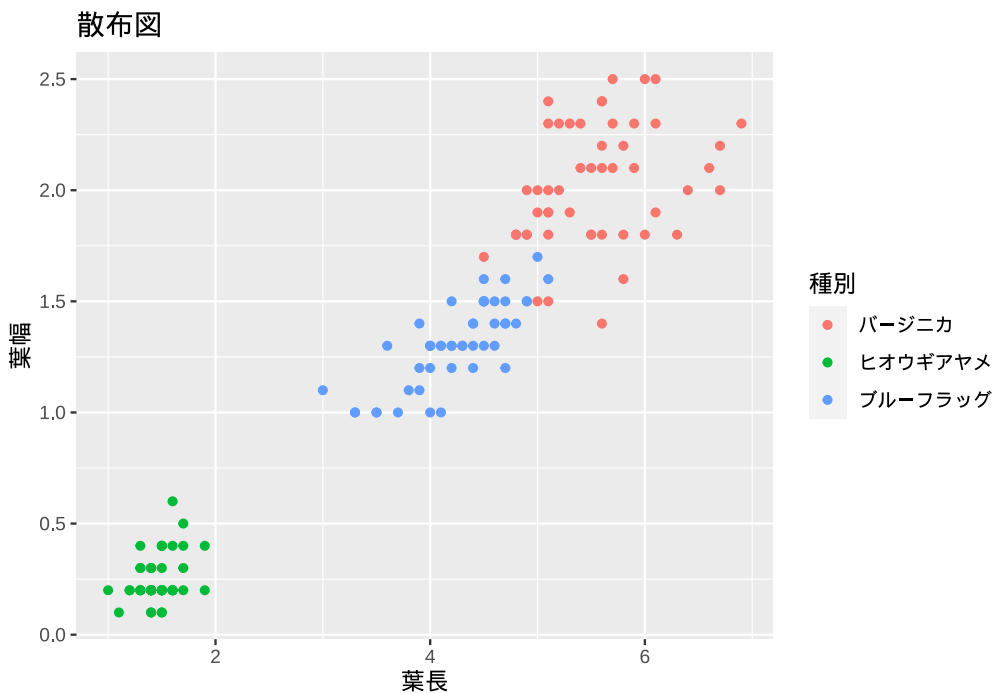
A.4 kable で表示

```
knitr::kable(df_iris[1:6, ])
```

| 萼長 | 萼幅 | 葉長 | 葉幅 | 種別 |
|-----|-----|-----|-----|---------|
| 5.1 | 3.5 | 1.4 | 0.2 | ヒオウギアヤメ |
| 4.9 | 3.0 | 1.4 | 0.2 | ヒオウギアヤメ |
| 4.7 | 3.2 | 1.3 | 0.2 | ヒオウギアヤメ |
| 4.6 | 3.1 | 1.5 | 0.2 | ヒオウギアヤメ |
| 5.0 | 3.6 | 1.4 | 0.2 | ヒオウギアヤメ |
| 5.4 | 3.9 | 1.7 | 0.4 | ヒオウギアヤメ |

A.5 ggplot でグラフを作成

```
ggplot(df_iris, aes(x = `葉長`, y = `葉幅`, col = `種別`)) +  
  geom_point() + labs(title = " 散布図", x = " 葉長", y = " 葉幅")
```



A.6 備考：

実は、一番難しいのが、PDF の作成だと思いますが、一応、上のものも、PDF を作成することが可能です。下のリンクのファイルを、いろいろな、形式で、出力してみてください。R Notebook と、PDF に出力したもののリンクを付けておきます。

- R Notebook
 - 右上の Code ボタンから、Rmd ファイルも取得できます。
- R Notebook RMD
 - Rmd の中身がテキストで表示されますから、コピーして、新規作成した、RNotebook ファイルに貼り付けることも可能です。
- PDF
 - 通常の PDF も、やはり PDF 形式の、beamer presentation も作成できます。

詳細は、これらのファイルに記載されていますから、参考にしてください。

A.7 参考：日本語の表示について

日本語が適切に表示されない!?

簡単ではなく、未解決の部分が何かなどを含め、わたしも十分理解できているか不明であるが、理解できていると思われる範囲で、備忘録のように記します。

R を使うという場合に限っても、R Studio IDE を使う場合、RStudio Cloud を使う場合、Google colab を使う場合、他のプラットフォームで使う場合で違ってくと思われますが、上にあげた、三種類のプラットフォームで確かめられるものについて書いていく。上に書いた以外に、R Studio IDE を、Windows 上で使う場合と、Mac 上で使う場合（Mac のシステムは Unix 系であるが、さまざまな Linux）でも、状況が異なるかもしれません。そこで、場合分けをして書いていくほうが安全ですが、それは、極力避け、どれにでも適用可能な方法を模索しながら書いていこうと思います。個人的に、日常的に判断を避ける努力をすることが大切だと思っていることも背景にある。さらに、ソフトウェア開発者は、むろん、そのような差異を理解して、どの環境でも、可能なように設計することを心がけていると思われますし、そのようなものが、R Project の正規のパッケージとして採用されていくべきだとも考えていますので、多少、理想も入っているが、これを基本として書いていこうと思います。十分なチェックができていないものもあるので、不具合などは、ぜひ、お知らせ願いたい。この文章も少しずつ、改善していければと思う。

通常、日本語、中国語、韓国語などが適切に表示できない場合は、文字のエンコーディング（Encoding: どのような情報として記録されているか）と、フォントの問題、さらに、システムがこれらをどう処理しているかの問題があると思われます。しかし、R の利用者として考えると、文字化けが起きたり、適切に文字が表示されないのは、以下の三つに分けられるようです。

1. データファイルなどを読み込んだときに適切に表示されない
2. 図の中のタイトルなどが、適切に表示されない
3. R Markdown の出力において、適切に表示されない

A.7.1 データファイルの読み込み

- tidyverse に含まれる readr には、guess_encoding が含まれており、一般的には、たとえば、
– read_csv(“./data/file_name.csv”) とすると、一番可能性の高いエンコーディングで読み込まれるようになっています。
- 使い方：guess_encoding(file, n_max = 10000, threshold = 0.2) とあり、10000 行で推測されたエンコーディング、または、確率を計算することを初期設定値（Default）にしています。Help によると、すべての行をチェックする場合は、n_max = -1 とすることが書かれています。
- これで問題がない場合が多いです。他の、readr 関数も同様です。しかし、これは、

あくまでも、CSV などのテキストファイルについてです。

- なお、`read_csv` などにも、`guess_max = min(1000, n_max)` もありますが、これは、column type を決めるためのものである。
- `read.csv()` など、base R では、`fileEncoding = “”, encoding = “unknown”` がオプションに含まれていたので、指定して読み込むことが通常でした。

A.7.2 図の中のテキスト

- 基本的には、日本語を表示できるフォントがインストールされていて、図の表示の前に `library(showtext); showtext_auto()` となっていれば、これ以降の図は、問題なく、表示されるはずです。
- 通常使っている、コンピュータ以外で、使うときに、フォントが入っていない場合などは、`tinytex` の命令を使って、`tinytex::tlmgr_install("ipaex")` とすれば、PDF を作成するときも、IPA フォント (International Phonetic Alphabet) を使えます。
- 二種類以上のフォントを使い分けたいときは、名前をつけて、それを `family = name` で指定する。
 - `showtext: Using Fonts More Easily in R Graphs` 参照。

A.7.3 R Markdown の出力

- PDF 作成には、 \TeX システムを使っているので、日本語を扱えるように、`tex-engine` や、`document class` や、`mainfont` を設定する必要があるが、R Markdown ファイルの YAML に、以下を加えれば問題がないようである。

header-includes:

- `\usepackage{xCJK}`
- `\setCJKmainfont{ipaexm.ttf}`
- `\setCJKsansfont{ipaexg.ttf}`
- `\setCJKmonofont{ipaexg.ttf}`

A.7.4 bookdown

`bookdown` を使って、電子書籍を出版する場合には、`bookdown` (リンク) を参照してください。

日本語におけるテンプレートは、こちら にあります。まずは、ページの下にある、README を読んでください。

A.7.5 参考としたもの

A.7.5.1 showtext: Using Fonts More Easily in R Graphs

- <https://CRAN.R-project.org/package=showtext>
 - <https://cran.r-project.org/web/packages/showtext/readme/README.html>
 - showtext: Using Fonts More Easily in R Graphs:
 - * <https://cran.r-project.org/web/packages/showtext/vignettes/introduction.html>
 - * <https://fonts.google.com>

A.7.5.2 sysfonts: Loading Fonts into R

- <https://CRAN.R-project.org/package=sysfonts>
 - <https://cran.r-project.org/web/packages/sysfonts/sysfonts.pdf>

A.7.5.3 foods4all: Examples of Graphs

- https://foods4all.github.io/examples/examples_of_graphs.html
 - 77.2 Japanese Environments 日本語環境（昔の記事：Last Updated: 2020-04-22）

付録 B

IT ツール

いくつかの便利なツールについて紹介します。

B.1 Git と GitHub

Git はバージョン管理システムで、GitHub はそれを、活用し、かつ他のメンバーと協力して開発など、さまざまな活動をするためのサイトです。公開が基本となっています。非公開にすることも可能ですが、公開することで、世界中のひとたちと協力していくことが可能になりますので、その利点も学んでいただければと思います。

B.1.1 概要

RStudio で R を使っている場合、Git-GitHub-RStudio の連携で使うことをお勧めします。しかし、これらは、三つとも、まったく異なるものですから、簡単な概要を書いておくことにします。

B.1.1.1 Git

これは、ファイルのバージョン（更新履歴）の管理システムで、単独で機能します。他の、プログラムなどに関係しない、他の文書ファイルであっても、バージョンを管理する場合に活用できます。特に、テキスト・ファイルの場合には、どこがどう改訂されているかを確認することもできます。また、基本的には、Unix の Shell プログラムで動作させるのが一般的です。Mac は、Unix システムの上に構築されているため、最初から、ユーティリティ（Utility）> ターミナル（Terminal）で、Shell コマンドが利用可能になっていますが、Windows の場合には、bash と呼ばれる Shell プログラムをインストールすることをお勧めします。Windows システムについてよくご存知の方は、他の方法を使っていた方が構いませんが、Git のインストールの時に、Git bash を選択して、簡単にインストールできますし、Unix システムの基本を理解するチャンスでもあり、Mac と同じ環境で説明できますから、ここでは、そちらを使います。Shell コマンドは、R Studio の中のターミナルを使って、利用することも可能です。（注：Windows のコマンド・プロンプト、

または、パワー・シェルをお使いの方は、利用環境が変化する可能性がありますから、そのまま使われる方が良いかもしれません。）

基本的なコマンドとしては、以下のものがあります。いまは、このようなものがある程度に、眺めておいてください。

- `git init`: 特定のディレクトリ（フォルダ）でバージョン管理を始める時に使います。
- `git status`: 現在の状況を確認するときに使います。
- `git diff file_name`: ファイルへの変更を確認します。
- `git log`: 過去の `commit` による履歴を確認する時に使います。
- `git add file_name`: ステージングという中間的な場所に登録します。
- `git commit -m "log message here"`: ステージングにあるものを、確定させます。引用符で囲まれた短いコメントを加えます。50 文字が上限です。
- `git help`: Help のリストが表示されます。
 - 例：`git help init` などと入力すると、説明を見ることができます。

B.1.1.2 Git Hub

Git でバージョン管理されているディレクトリ（フォルダ）の状態を示す、クラウドサービスです。更新されている、状態を確認するとともに、変更履歴なども確認できます。また、Git Hub サービスを利用して、ファイルを公開、共有することも可能です。Pages サイトを利用することで、ホームページとして HTML ファイルなどを公開することもできるため、レポートを公開したり、この電子書籍のように、`bookdown` パッケージを利用して作成したものを、インターネット上に公開することも可能です。

お気付きかと思いますが、この電子書籍も、リンクされている、他の文書も、URL（アドレス）をみると、GitHub になっていますし、パッケージや、データなども、GitHub へのリンクが示されている割合が、非常に高いと思います。

最初に、Git で管理されている、ディレクトリ（フォルダ）（これを、ローカル・リポジトリと言います）と、GitHub 内のリポジトリ（リモート・リポジトリまたはアップストリーム・リポジトリと言います。ここでは、リモート・リポジトリと呼ぶことにします）を結びつけるステップが必要です。

B.1.1.3 RStudio 連携

コマンドライン（シェル）で行う作業や、ローカル・リポジトリを、リモート・リポジトリに結びつける作業を、RStudio の中で行うことが可能です。

B.1.2 はじめかた

1. Git のインストール

- Windows と Mac で異なりますので注意が必要です。Mac については、**Mac** と書

いてあるところを読んでください。

- **Windows** の場合は、git-scm にアクセスしてダウンロード、インストールしてください。セットアップ (Setup) で、2箇所、注意点があります、それ以外は、すべて、初期設定のままで変更は必要ありません。
 - Choosing the default editor used by Git: 設定で、エディタ (Editor) を設定しますが、vi, vim に慣れていない方は、nano を選択することをお勧めします。(nano^{*1} は、メニューが下に出るので、それを見て操作することが可能なエディターです。)
 - Adjusting Your Path Environment: Windows のコマンドライン・ツール (command line prompt) を使っていない方は、Git Bash のインストールを選択してください。さらに、Git and optional Unix tools from the Windows Command Prompt を選択することをお勧めしますが、上で書いたように、Windows のコマンド・プロンプトになれておられる方で、それを使い続けたいかたは、Use Git from Git Bash only を選択されるのが良いかもしれません。
 - 最後に、RStudio の設定 (Tools > Global Option) で、Terminal から、Git Bash を選択^{*2}し、Tools から、New Terminal を選択^{*3}します。
- **Mac** は、最初から、Install されていると思います。ユーティリティ (Utility) > ターミナル (Terminal) を開いて^{*4}、`git --version` とすると、インストールされているバージョンが表示されると思います。バージョンがでない場合には、Install するかと聞かれます。このときに、Git だけをインストールすることも、Xcode という開発環境を同時にインストールすることも可能です。(インストールするように指示が出なければ、App Store から、インストールできます。もし、そのあとで、git などのコマンドで `xcrun: error` などとエラーが出たら、`xcode-select --install` としてください。) インストールが終了したら、もう一度、`git --version` と Terminal に入力して、結果を確認してください。

2. GitHub のアカウント取得

- GitHub サイト に、アカウントを作成します。アカウント名は、短く、分かりやすく、覚えやすいものをよく考えて決めてください。Email Address だけで、無償で作成できます。

3. RStudio の左下の窓枠の Terminal タブ^{*5}から、GitHub アカウントに連携する設定を行います。下の2行を、1行ずつ、コピーして、Terminal に入力してください。

```
git config --global user.name "Your Name" # GitHub の User Name
```

```
git config --global user.email "your@email.com" # GitHub に登録したメールアドレス
```

^{*1} GNU nano は、コンソールウィンドウで動作する小型でフレンドリーなテキストエディタです。(GNU nano is a small and friendly text editor running in the console window.)

^{*2} 自動的に選択されているかもしれません

^{*3} すでにタブがあればこの作業は不要です

^{*4} RStudio を既にお使いの方は、左下の窓枠から、Terminal タブを選択できますので、それを使うことも可能です。

^{*5} Terminal がない場合は、Tools > Terminal > New Terminal とすると表示されます。

4. RStudio の、Tools > Global Option の、Git/SVN タブを開き、Git Executable とあるところに、Git 実行プログラムのある場所を入れます。
 - **Windows** の場合は、C:/Program Files/Git/bin/git.exe だと思いますが、Browse ボタンから確認してください。
 - **Mac** の場合は、/usr/bin/git になるかと思いますが、Browse ボタンから確認してください。
5. その下に、Create RSA key とありますから、それを押し、Create ボタンを押してください*6。すると、View RSA key から、暗号キーも確認できます。（この作業は、Terminal から、ssh-keygen -t rsa として作成することも可能です。この作業で、~/.ssh/ 内に、SSH キーが記述されたファイルが作成されます。）
6. GitHub アカウントで公開鍵を利用できるようにします。まず、RStudio の方で作成した、RSA key（Tools の、Global Option の Git/SVN）の下にある view を押すと見ることができ、上に、書いてあるように、そこに出てきたものを、コピーします。次に、GitHub にログインし、右上のアイコンの右の三角から、設定 (Setting) を選択し、SSH 公開鍵 (SSH and GPG Keys) を選択します。新しい公開鍵を追加 (New SSH Key) を選択すると、SSH キーを貼り付けることができます。（リポジトリの左上にある、アカウント名をクリックし現れるダッシュボードの左上の大きなアイコンをクリックしても「アカウント設定」が現れ、SSH and GPG Keys を見つけることができると思います。）

（なお、RStudio ではなく、Terminal からコピーするときは、Unix では、pbcopy < ~/.ssh/id_rsa.pub などとします。Windows の場合は、pbcopy が使えない可能性があるので、そのときは、Terminal から、Git Bash を使い、use < ~/.ssh/id_rsa.pub とします。Terminal に慣れておられない方には、上に紹介した、RStudio からコピーする方が簡便かと思います。）

7. これで設定終了ですので、R Studio を再起動させてください。

B.1.3 GitHub にあるリモート・リポジトリ (Remote Repo) から始める場合

1. GitHub にログインして、既存のリポジトリを開きます。
2. Code の、Clone から、リンク先のアドレスを入手。https と SSH を選べますが、SSH を選び、コピーします。
3. RStudio から、New Project とし、Version Control を選択し、ディレクトリーを決めたら、上でコピーした、ものを、貼り付けて、Project を作成します。

この手続きで、リモート・リポジトリのファイルがすべて、RStudio のプロジェクトに入ります。

*6 Passphrase (optional) と出ますが、無視して下さって構いません。

実はこの手続きで、公開されている、他のリポジトリも取り込むことができます。ただし、編集して、改訂していくには、自分のリポジトリに、繋ぐことになります。そのときは、次の項目を見てください。

B.1.4 自分のコンピュータのリポジトリ (Local Repo) から始める場合

1. RStudio から新しい、プロジェクト (Project) を作成 `test0` としておきましょう。
2. GitHub に、新しい、レポジトリを作成して繋げる
 - 自分の GitHub アカウントに、新しい、レポジトリをプロジェクトと同じ名前 `test0` で作成します。同じ名前でもなくてもかまわないのですが、関連がしやすいので、同じ名前がお勧めです。
 - Quick Setup というページが表示されますから、その、下の Set up in Desktop or の右から、https と SSH を選べますが、SSH を選び、コピーします。
 - プロジェクトの中の左下の窓枠の、Terminal から次を実行します。

```
echo "# Project test0" >> README.md # README の表題を書きます。
```

```
git init
```

```
git add README.md
```

```
git commit -m "First commit. README.md"
```

```
git branch -M main
```

```
git remote add origin `git@github.com:icu-hsuzuki/test0.git`
```

```
# わたしのアカウントの場合には、コピーしたものを貼り付けると、上のようになります。
```

```
git push -u origin main
```

再起動すると、プロジェクトの右上の窓枠に、Git と表示されていると思います。開いて、いくつかファイル名があれば、これを選択し、Commit ボタンを押し、短い message を書いて、Commit とし、そのあとで、Push ボタンを押してください。HEAD -> main と表示されれば、OK です。

B.1.5 GitHub Pages について

GitHub は、Web 上のサービスですが、たとえ、リポジトリの中に、HTML ファイルなどが存在しても、ブラウザーで表示させることはできません。公開する方法として、わたしは、以下のようにしています。

公開しブラウザーで読めるようにしたいファイルを、docs というディレクトリに入れます。Setting を選択し、左の帯から、Pages を選び、Branch を、main、/docs として、Save します。すると、docs に入っているファイルは、リンクを指定すれば、ブラウザーで、見ることができます。

リンクは、まず、リポジトリの右上の、About の右のギアマークから、Use your GitHub Pages website に、チェックを入れると、URL が表示されますから、その URL に続けて、ファイル名を加えたものが、そのファイルの、リンクとなります。

B.1.6 Bookdown パッケージによる、電子書籍の執筆

bookdown を利用して、始める方法を簡単に記します。

Git-GitHub-RStudio の設定は済んでいると仮定して書きます。また、Bookdown を利用するには、bookdown パッケージのインストールが必要です。RStudio の Tools から、Install Packages を選択して、インストールしてください。また、bs4_book スタイルを利用する場合は、RStudio などのバージョンによっては、bslib と downlit も、インストールする必要があるようです。

以下では、R Studio で新しい、プロジェクトとして始める場合と、テンプレート・リポジトリを使う方法を説明します。

B.1.6.1 RStudio での設定

1. 新しい book プロジェクトを始めます、標準設定の book と、bs4_book を選択できます。このディレクトリ（フォルダ。ローカル・リポジトリと呼びます）の名前をリモート・リポジトリの名前と同じにして、作成します^{*7}。
2. Files の、Rename 機能を使って _book ディレクトリ（フォルダ）の名前を、docs に変更します。
3. _bookdown.yml を編集し次の行を加えます（私は 2 行目に入れています）。
output_dir: docs
GitHub Pages の機能を使って、公開するための変更です。
4. @GitHub: 新しいリポジトリ（リモート・リポジトリと言います）を作成します。ローカル・リポジトリと同じ名前にし、簡単な説明を加えます。
5. 左下の窓枠から、Terminal タブを選択します。以下は例です。git remote add origin のところは、適当に変更してください。

```
echo "# ds-book" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin git@github.com:icu-hsuzuki/ds4aj.git # 変更
git push -u origin main
```

5. RStudio を一旦終了し、もう一度、そのプロジェクトを立ち上げると、右上の窓枠に、Git タブが表示されるはずです。
6. Build から、Build Book ボタンを利用すれば、作成されます。
7. Git タブから: Commit “first build” とし、Push をします。
8. @GitHub: Settings から、Pages > main > docs と変更すると、公開されます。

^{*7} 異なる名前でも可能ですが、特別な理由がない限り同じにしておいた方が良いでしょう

9. Code に戻り、About の右のギアマークから、

■B.1.6.1.1 他の PC での作業

1. Login to GitHub account
2. Copy SSH address under Code>Clone
3. Create a new project using Version Control:Git with the SSH address by setting the directory name
4. Edit README.md and test Git Commit and Push

B.1.6.2 テンプレートを利用する方法

■B.1.6.2.1 bookdown-demo bookdown の電子書籍 (bookdown: Authoring Books and Technical Documents with R Markdown) の、はじめてみよう (Get started) から、GitHub リポジトリ `rstudio/bookdown-demo` にリンクがついています。

1. 自分の、GitHub アカウントに、ログインします。
2. `rstudio/bookdown-demo` にアクセスし、Use this template から、create a new repository を選択します。
3. 自分のアカウントの中に、コピーされたリポジトリがつくられます。
4. 上で説明した「GitHub にあるリモート・リポジトリ (Remote Repo) から始める場合」を利用してください。

よくできているテンプレートです。しかし、このままでは、公開されません。bookdown: Authoring Books and Technical Documents with R Markdown には、いくつかの公開方法が書かれていますから、参照してください。

■B.1.6.2.2 `jtr13/bookdown-template` YouTube ビデオ 5 分間で、bookdown の本を作成する方法 (How to create a bookdown book in 5 minutes) で紹介されているものです。

1. 自分の、GitHub アカウントに、ログインします。
2. `jtr13/bookdown-template` にアクセスし、Use this template から、create a new repository を選択します。
3. 自分のアカウントの中に、コピーされたリポジトリがつくられます。
4. 上で説明した「GitHub にあるリモート・リポジトリ (Remote Repo) から始める場合」を利用してください。

ビデオの中では、簡単に、GitHub Pages (GitHub のホームページサービス) に、公開する方法が説明されています。最も、大切な部分は、テンプレートがコピーされた、自分

のリポジトリの Settings から、Pages を選び、Branch を main > docs とする部分です。このようにすることで、公開されます。上の、「GitHub Pages」で説明していることと同じです。

■B.1.6.2.3 日本語テンプレート bs4_book スタイルでの、簡単な、日本語テンプレートを作成しました。

1. 自分の、GitHub アカウントに、ログインします。
2. icu-hsuzuki/bs4_book_template にアクセスし、Use this template から、create a new repository を選択します。
3. 自分のアカウントの中に、コピーされたりポジトリがつくれます。
4. 上で説明した「GitHub にあるリモート・リポジトリ (Remote Repo) から始める場合」を利用してください。

まだ、不十分ですが、少しずつ、改訂していきます。使い方は、リポジトリの README に、注意点などは、最後の章 Bookdown に書いておく予定です。

B.1.7 大きなファイルに関すること

100M 未満のファイルだけを利用するのがよいですが、それより大きいものを、GitHub に挙げようとして、問題が起こることがあります。その対処法を書いておきます。

個人的には、大きなデータや、本などを、Build して、100M 以上の、非常に大きな、TeX 関連のファイルができることができました。

備忘録も含めて、書いておきます。

B.1.7.1 大きなファイルの取り除きかた

下のサイトからの引用です。非常に分かりやすく書かれています。Terminal で作業を行います。いま、Add した場合と、いくつか、前のステップで、Add した場合で、対応の方法が異なります。

- Tutorial: Removing Large Files from Git
 - Scenario 1: The Large File Was Just Added in the Most Recent Commit
 - * `git rm --cached big_file_name`
 - * `git commit --amend -C HEAD`
 - Scenario 2: The Large File Was Committed Prior to The Most Recent Commit
 - * Locating the Last “Good” Commit: `git log --oneline`
 - * Initiate a Rebase Between the Last “Good” Commit and the Current Commit: `git rebase -i 8464da4`

- * This will open up a file in your Git editor (in my case, Vim), that looks something like this:
 - pick -> edit
- * `git rm --cached csv_building_damage_assessment.csv`
- * `git commit --amend -C HEAD`

B.1.8 複数のコンピュータから利用する方法

わたしも、いくつかのコンピュータから、同じ GitHub アカウントにアクセスして作業しています。

同じ SSH キーを、複数のコンピュータで利用することも可能です。特に、コンピュータの更新時に、移行する場合は、元の環境をそのまま使うことも可能で便利です。

ただし、基本的には、それぞれのコンピュータで別々の SSH キーを使用するのがお勧めです。問題が発生した時に、個別のコンピュータの課題として解決することができます。

(たとえば、RSA 形式で作成した) 複数の SSH キーを使用するときは、GitHub アカウントに公開鍵を追加する必要があります。

GitHub アカウントに別の公開鍵を追加するには、GitHub にログインし、右上のアイコンの右の三角から、設定 (Setting) を選択し、SSH 公開鍵 (SSH and GPG Keys) を選択します。新しい公開鍵を追加 (New SSH Key) を選択すると、SSH キーを貼り付けることができます。(リポジトリの左上にある、アカウント名をクリックし現れるダッシュボードの左上の大きなアイコンをクリックしても「アカウント設定」が現れ、SSH and GPG Keys を見つけることができますと思います。)

コピーを貼り付ける時には、RStudio の、Global Option の、Git/SVN タブから、View public key を見ると、コピーできるようになっています。

Terminal からコピーするときは、Unix では、`pbcopy < ~/.ssh/id_rsa.pub` などとします。Windows の場合は、`pbcopy` が使えない可能性があるので、そのときは、Terminal から、Git Bash を使い、`use < ~/.ssh/id_rsa.pub` とします。Terminal に慣れておられない方には、上に紹介した、RStudio からコピーする方が簡便かと思います。

SSH キーの最後には、コンピュータ名とコンピュータのアカウント名などが入っていると思います。

この設定をすれば、どちらのマシンからでも SSH 経由で Github リポジトリにアクセスできるようになります。

B.1.9 複数のアカウントを一つのコンピュータから利用する方法

わたしも複数の GitHub アカウントを利用しています。

- `~/.ssh` 内に複数 (例では三つ)、`ssh-keygen -t rsa` でファイル作成

- `id_rsa` , `id_sub1_rsa`, `id_sub2_rsa`
- 上の複数のコンピュータから利用する時に説明してあるように、SSH キーを GitHub に登録
- `~/.ssh` 内の `config` ファイル (`~/.ssh/config`) を編集 (nano などを利用)
- `~/.gitconfig`, `~/.gitconfig_sub1`, `~/.gitconfig_sub2` を作成

詳しくは、参考にしたサイトを参照してください。

B.1.10 共同作業をする場合

以下では、ひとり管理者（とりまとめ役）がいて、その人がまずは、レポジトリを作成し、何人かの編集者で、共同編集することを想定して、書きます。内容は本でなくても、プログラムでも、ノートでも構いませんが、イメージしやすくするため、本を編集することを想定して書いていきます。

1. 編集に関わる全員が、それぞれ、個人の、GitHub アカウントを作成します。
2. リポジトリをフォーク (Fork) : 共同編集する (本の) レポジトリ (オリジナル・レポジトリと呼ぶことにします*8) にいき、フォークします。フォークすることで、編集者のリポジトリに、コピーが作成され、大元のリポジトリ (オリジナル・レポジトリ) の変更を依頼することができます。名前をつけますが、元のものと同じでも、フォークしたことがわかるようにしておいても良いでしょう。
- 一回フォーク (Fork) したら、2 度目以降は、オリジナル・レポジトリから、フォーク (Fork) する必要はありません。ただし、改訂されているかもしれないので、更新 (Sync fork) する必要があります。GitHub 上の、フォーク (Fork) した、自分のレポジトリを見ると、更新されている場合は、オリジナル・レポジトリに、Sync することができますから、Sync し*9、最新の状態にしておきます。よくみると、オリジナル・レポジトリとの関係が読み取れるかと思います。
3. (フォークした) リポジトリをクローン: 編集者のコンピュータにフォークして得られたリポジトリを、クローンし、自分のコンピュータ内に、ローカル・リポジトリを作成します。
4. RStudio の右上の、Git タブの右にある、New Branch に、新しい、Branch (編集用枝) を作成します。すると、その右に、Branch 名が記載されます。それが、現在の Branch 名です。その右の三角を押すと、main など、すべての Branch が見えるはずです。編集のための Branch が表示されている (アクティブ) になっていることを確認します*10。

*8 upstream repository と呼ぶこともあります。自分のリモート・リポジトリを upstream repository と呼ぶこともあるので、ここでは、オリジナル・リポジトリと呼ぶことにします。

*9 Synk fork ボタンを押すと This branch is out-of-date などと表示され、Update branch を選択すると、update されます

*10 New Branch の右に、現在どの Branch がアクティブになっているかが表示されています。そこに現れている Branch に、commit など、編集を加えていくことになります。

5. 編集: 編集者のコンピュータ (RStudio 上) で編集を行います。
6. 編集結果をコミット: 編集者のコンピュータで編集した後に、簡単な説明を書いて、コミットします。
7. 編集結果をプッシュ: フォークした GitHub の自分のリポジトリに、プッシュします。
 - 実際にどうなるかをみるために、Build をする場合もあるかもしれませんが、すると、たくさんのファイルが書き換えられます。管理者も、一つ一つ丁寧に見るのは大変ですから、実際に、書き換えた、Rmd ファイルなどだけに、チェックを入れて、Commit + Push するのも良いでしょう。
 - この時点で、RStudio 上の、Branch を main に戻しておくとも良いでしょう^{*11}。
8. プル依頼を作成: フォークした自分の (リモート) リポジトリで、編集の内容を簡単に記述して、プルリクエスト (Pull Request) をします。
 - 編集して、プッシュした、Branch を選択し、オリジナル・レポジトリと異なった状態であることを確認します。
 - Pull Request を選択します。
 - オリジナル・レポジトリの管理者は、修正をしやすいように、修正の要点を書いておきます。
 - ここまでで終了です。管理者にメールが届きます。
 - このあと、編集者が何をするか、確認してください。
9. 管理者: 編集結果の確認: オリジナル・リポジトリの所有者は、編集結果を確認し、承認する場合は、main に、統合 (マージ) します。
 - 統合 (マージ) すると、Pull Request をした編集者に、メールが届きます。
10. 変更の同期: 編集者は、承認された、オリジナル・リポジトリを、フォークした自分のリポジトリに、同期 (Sync Fork) させます。
 - 自分のコンピュータ上の、ローカル・リポジトリの main branch も (アクティブな、Branch 名を確認して) Pull をして、更新しておくとも良いでしょう。
 - 少し、複雑な、変更の場合には、注意が出ます。詳細は省略しますが、Terminal で、一つの選択肢の、`git config pull.rebase true # rebase` を入力して、それから、もう一度、Pull をすると良いでしょう。これによって、オリジナル・リポジトリと同じ状態を維持することができます。

B.1.10.1 要点

- フォークして作成したリポジトリ (リモートも、ローカルも) の、main branch は、オリジナル・レポジトリと同じ状態にしておく。

^{*11} New Branch の右のところで、選択します

- 編集は、編集用に、branch を作成して、その branch を、リモート・リポジトリにプッシュして、プル・リクエストを行う。
- プル・リクエストの結果が反映されたら、リモートも、ローカルも、同期しておく。
- プル・リクエストが反映される前に、また編集するときは、別の branch を作成して、また、プル・リクエストをするのがよいでしょう。
- 管理者は、merge で、編集結果を反映するときに、注意を要しますから、管理者が編集結果を反映しやすいように、丁寧に、プル・リクエストのときの説明をしておくのがよいと思います。

B.1.11 参考にしたサイト

B.1.11.1 Git - GitHub - RStudio 関連

- git – everything is local
 - About、Documentation など
- GitHub Docs: Hellow World
 - 基本的なことがコンパクトにまとまっている GitHub のサイトです。日本語もサポートしています。
- Introduction to Data Science, by Rafael A. Irizarry
 - Git and GitHub
 - edX の、データサイエンスのコースの教科書に入っています。よく、まとまっていると思います。原語は英語ですが、Google などの翻訳機能を使っても、十分理解することができると思います。Git と GitHub の概要から、Bookdown パッケージによる、電子書籍の執筆の前までは、基本的に、この教科書を参考にしていますが、それぞれのステップでの、スクリーンショットもたくさん掲載されており、確認がしやすいようになっています。
- Git-GitHub-RStudio: 個人的メモ

B.1.11.2 Bookdown 関連

- The bookdown book: <https://bookdown.org/yihui/bookdown/>
- The bookdown package reference site: <https://pkgs.rstudio.com/bookdown>
- How to create a bookdown book in 5 minutes: <https://www.youtube.com/watch?v=m5D-yoH416Y>

B.1.11.3 複数アカウント・複数のキー関連

- Using the same github account from multiple PCs
- 複数の GitHub アカウントを使い分けたい時の設定方法と Tips
- Adding a new SSH key to your GitHub account

付録 C

Bookdown

C.1 About

This is a *sample* book written in **Markdown**. You can use anything that Pandoc's Markdown supports; for example, a math equation $a^2 + b^2 = c^2$.

C.1.1 Usage

Each **bookdown** chapter is an .Rmd file, and each .Rmd file can contain one (and only one) chapter. A chapter *must* start with a first-level heading: **# A good chapter**, and can contain one (and only one) first-level heading.

Use second-level and higher headings within chapters like: **## A short section** or **### An even shorter section**.

The `index.Rmd` file is required, and is also your first book chapter. It will be the homepage when you render the book.

C.1.2 Render book

You can render the HTML version of this example book without changing anything:

1. Find the **Build** pane in the RStudio IDE, and
2. Click on **Build Book**, then select your output format, or select “All formats” if you'd like to use multiple formats from the same book source files.

Or build the book from the R console:

```
bookdown::render_book()
```

To render this example to PDF as a `bookdown::pdf_book`, you'll need to install XeLaTeX. You are recommended to install TinyTeX (which includes XeLaTeX): <https://yihui.org/tinytex/>.

C.1.3 Preview book

As you work, you may start a local server to live preview this HTML book. This preview will update as you edit the book when you save individual .Rmd files. You can start the server in a work session by using the RStudio add-in “Preview book”, or from the R console:

```
bookdown::serve_book()
```

C.2 Hello bookdown

All chapters start with a first-level heading followed by your chapter title, like the line above. There should be only one first-level heading (#) per .Rmd file.

C.2.1 A section

All chapter sections start with a second-level (##) or higher heading followed by your section title, like the sections above and below here. You can have as many as you want within a chapter.

An unnumbered section

Chapters and sections are numbered by default. To un-number a heading, add a {.unnumbered} or the shorter {-} at the end of the heading, like in this section.

C.3 Cross-references

Cross-references make it easier for your readers to find and link to elements in your book.

C.3.1 Chapters and sub-chapters

There are two steps to cross-reference any heading:

1. Label the heading: # Hello world {#nice-label}.
 - Leave the label off if you like the automated heading generated based on your heading title: for example, # Hello world = # Hello world {#hello-world}.
 - To label an un-numbered heading, use: # Hello world {-#nice-label} or {# Hello world .unnumbered}.

2. Next, reference the labeled heading anywhere in the text using `\@ref(nice-label)`; for example, please see Chapter C.3.
 - If you prefer text as the link instead of a numbered reference use: any text you want can go here.

C.3.2 Captioned figures and tables

Figures and tables *with captions* can also be cross-referenced from elsewhere in your book using `\@ref(fig:chunk-label)` and `\@ref(tab:chunk-label)`, respectively.

See Figure C.1.

```
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

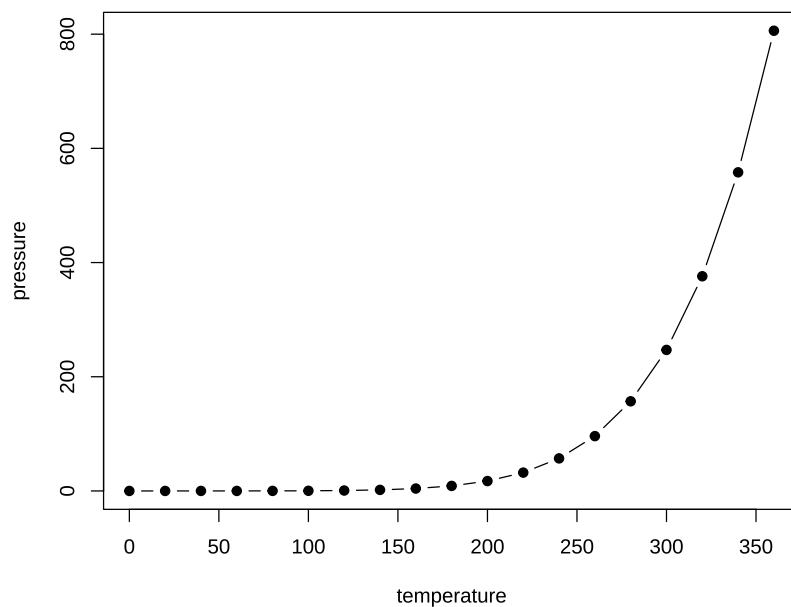


Figure C.1 Here is a nice figure!

Don't miss Table C.1.

```
knitr::kable(
  head(pressure, 10), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

C.4 Parts

You can add parts to organize one or more book chapters together. Parts can be inserted at the top of an .Rmd file, before the first-level chapter heading in that same file.

表C.1 Here is a nice table!

| temperature | pressure |
|-------------|----------|
| 0 | 0.0002 |
| 20 | 0.0012 |
| 40 | 0.0060 |
| 60 | 0.0300 |
| 80 | 0.0900 |
| 100 | 0.2700 |
| 120 | 0.7500 |
| 140 | 1.8500 |
| 160 | 4.2000 |
| 180 | 8.8000 |

Add a numbered part: # (PART) Act one {-} (followed by # A chapter)

Add an unnumbered part: # (PART*) Act one {-} (followed by # A chapter)

Add an appendix as a special kind of un-numbered part: # (APPENDIX) Other stuff {-} (followed by # A chapter). Chapters in an appendix are prepended with letters instead of numbers.

C.5 Footnotes and citations

C.5.1 Footnotes

Footnotes are put inside the square brackets after a caret `^[]`. Like this one ^{*1}.

C.5.2 Citations

Reference items in your bibliography file(s) using `@key`.

For example, we are using the **bookdown** package (Xie, 2023) (check out the last code chunk in `index.Rmd` to see how this citation key was added) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015) (this citation was added manually in an external file `book.bib`). Note that the `.bib` files need to be listed in the `index.Rmd` with the YAML `bibliography` key.

The `bs4_book` theme makes footnotes appear inline when you click on them. In this example book, we added `csl: chicago-fullnote-bibliography.csl` to the

^{*1} This is a footnote.

`index.Rmd` YAML, and include the `.cs1` file. To download a new style, we recommend:
<https://www.zotero.org/styles/>

The RStudio Visual Markdown Editor can also make it easier to insert citations:
<https://rstudio.github.io/visual-markdown-editing/#/citations>

C.6 Blocks

C.6.1 Equations

Here is an equation.

$$f(k) = \binom{n}{k} p^k (1-p)^{n-k} \quad (\text{C.1})$$

You may refer to using `\@ref{eq:binom}`, like see Equation (C.1).

C.6.2 Theorems and proofs

Labeled theorems can be referenced in text using `\@ref{thm:tri}`, for example, check out this smart theorem C.1.

Theorem C.1. *For a right triangle, if c denotes the length of the hypotenuse and a and b denote the lengths of the **other** two sides, we have*

$$a^2 + b^2 = c^2$$

Read more here <https://bookdown.org/yihui/bookdown/markdown-extensions-by-bookdown.html>.

C.6.3 Callout blocks

The `bs4_book` theme also includes special callout blocks, like this `.rmdnote`.

You can use **markdown** inside a block.

```
head(beaver1, n = 5)
#>   day time temp activ
#> 1 346  840 36.33     0
#> 2 346  850 36.34     0
#> 3 346  900 36.35     0
#> 4 346  910 36.42     0
#> 5 346  920 36.55     0
```

It is up to the user to define the appearance of these blocks for LaTeX output.

You may also use: `.rmdcaution`, `.rmdimportant`, `.rmdtip`, or `.rmdwarning` as the block name.

The R Markdown Cookbook provides more help on how to use custom blocks to design your own callouts: <https://bookdown.org/yihui/rmarkdown-cookbook/custom-blocks.html>

C.7 Sharing your book

C.7.1 Publishing

HTML books can be published online, see: <https://bookdown.org/yihui/bookdown/publishing.html>

C.7.2 404 pages

By default, users will be directed to a 404 page if they try to access a webpage that cannot be found. If you'd like to customize your 404 page instead of using the default, you may add either a `_404.Rmd` or `_404.md` file to your project root and use code and/or Markdown syntax.

C.7.3 Metadata for sharing

Bookdown HTML books will provide HTML metadata for social sharing on platforms like Twitter, Facebook, and LinkedIn, using information you provide in the `index.Rmd` YAML. To setup, set the `url` for your book and the path to your `cover-image` file. Your book's `title` and `description` are also used.

This `bs4_book` provides enhanced metadata for social sharing, so that each chapter shared will have a unique description, auto-generated based on the content.

Specify your book's source repository on GitHub as the `repo` in the `_output.yml` file, which allows users to view each chapter's source file or suggest an edit. Read more about the features of this output format here:

https://pkgs.rstudio.com/bookdown/reference/bs4_book.html

Or use:

```
?bookdown::bs4_book
```

参考文献

- Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.
- Xie, Y. (2023). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.34.