My Data Science

Hiroshi Suzuki

4/18/23

Table of contents

この文書について

データサイエンス、AI（Artificial Intelligence 人工知能）関連で考えたこと、および調べたことを、メモがわりに、書いていこうと思っている。いずれは、データサイエンスと言える内容を発信できると嬉しい。

まとまった内容を書く予定はいまのところない。

現在は、データサイエンスと言える内容を、書くだけの力がないだけではなく、以下のプロジェクトが進行中で、そのための、下調べのメモを中心に書いていく。

データサイエンスをはじめましょう

リンク・GitHub Repository

基本的に、自習型の、学習コンテンツで、大学などで、教えるためにも、学習者に参考にしてもらうための、内容を書いていきたい。

断片的なものは、多くあるが、まとまったものは、英文では豊富にあるにもかかわらず、日本語では、十分ではないと感じているからである。

むろん、教科書と言われるような、標準的なものを目指しているわけではない。自分が学んでいて、たいせつだと考えることを、順を追って、書いていく予定である。

以下を土台としている。

- 大学院一般向けに、さまざまな背景の学生に何年か教えてきた内容
  - Data Analysis for Researchers 2022
  - Data Analysis for Researchers 2021
- 中級マクロを受講している学生向けの、特別講師として、教えた内容に、後日加筆したもの
  - R ではじめるデータ・サイエンス
- 経済学の大学院生むけの、特別講師として教えた内容

  - Introduction to R - A Short Course

データサイエンスを教えませんか

リンク・GitHub Repository

データサイエンス教育に関する素材を集めたものとしてスタートしたが、基本的には、データサイエンスにあまり馴染みがないが、学びながら、教えていきたい方向けのコンテンツを目指している。

データサイエンス教育は喫緊の課題であるが、情報科学の一部の教員または技術者以外は、特に日本では、データサイエンスに馴染みがない。しかし、大学（または、高等専門学校や高校など）で全学的に、データサイエンス教育を推進しようとすると、多くの教員の関与が必要である。

社会科学系を中心として、大学教員の中には、データ分析を使う分野は多い。経済学、社会学、心理学、言語学などなど。そのような教員にも、積極的にに、データサイエンス教育に加わっていただくための支援を目的として、書いていこうと思っている。

なお、この背景にある考え方は、以下の講演などで、語っている。

- 日本でのデータサイエンス教育の課題にどう向き合うか
  - スライド・ビデオ（2023 年 3 月 9 日の数理科学研究所での午前の講演）
- 生涯学び続ける基盤を構築するデータサイエンス・コースの開発
  - スライド・ビデオ（2023 年 3 月 9 日の数理科学研究所での午後の講演前半・後半は「R ではじめるデータ・サイエンス」
- 教養としてのデータサイエンス教育～MOOCs の活用を視野に入れて～　日本数学会教育委員会主催教育シンポジウム：文理共通して行う数理・データサイエンス教育における講演　時：2019 年 9 月 17 日　於：金沢大学角間キャンパス
  - スライド・音声付き画面収録ビデオ

第 1 章

Introduction

This is a book created from markdown and executable code.

See Knuth (1984) for additional discussion of literate programming.

```
1 + 1
```

```
[1] 2
```

第 2 章

Articles

Articles のレビューなどを書いていく

2.1 AI

2.1.1 GPT-4 Technical Report

- Source: https://arxiv.org/abs/2303.08774
- Abstract

  We report the development of GPT-4, a large-scale, multimodal model which can accept image and text inputs and produce text outputs. While less capable than humans in many real-world scenarios, GPT-4 exhibits human-level performance on various professional and academic benchmarks, including passing a simulated bar exam with a score around the top 10% of test takers. GPT-4 is a Transformer- based model pre-trained to predict the next token in a document. The post-training alignment process results in improved performance on measures of factuality and adherence to desired behavior. A core component of this project was developing infrastructure and optimization methods that behave predictably across a wide range of scales. This allowed us to accurately predict some aspects of GPT-4's performance based on models trained with no more than 1/1,000th the compute of GPT-4.

- アブストラクト

  GPT-4 は、画像とテキストを入力し、テキストを出力することができる大規模なマルチモーダルモデルであり、その開発について報告する。GPT-4 は、多くの実世界のシナリオにおいて人間より能力が劣るものの、模擬司法試験に受験者の上位 10%程度のスコアで合格するなど、様々な専門的・学術的ベンチマークにおいて人間レベルの性能を発揮することができる。GPT-4 は、文書中の次のトークンを予測する

ために事前に学習された Transformer ベースのモデルです。GPT-4 は、文書中の次のトークンを予測するよう事前に学習させた Transformer ベースのモデルで、学習後のアライメントプロセスにより、事実の正確さと望ましい行動への忠実さを評価するパフォーマンスが向上しています。このプロジェクトの中核をなすのは、幅広いスケールで予測可能な振る舞いをするインフラと最適化手法の開発でした。これにより、GPT-4 の 1,000 分の 1 以下の計算量で学習したモデルから、GPT-4 の性能の一部を正確に予測することができるようになりました。

- Contents

2.1.2 Self-Refine: Iterative Refinement with Self-Feedback

- Source: https://arxiv.org/abs/2303.17651
- Abstract

Like people, LLMs do not always generate the best text for a given generation problem on their first try (e.g., summaries, answers, explanations). Just as people then refine their text, we introduce SELF-REFINE, a framework for similarly improving initial outputs from LLMs through iterative feedback and refinement. The main idea is to generate an output using an LLM, then allow the same model to provide multi-aspect feedback for its own output; finally, the same model refines its previously generated output given its own feedback. Unlike earlier work, our iterative refinement framework does not require supervised training data or reinforcement learning, and works with a single LLM. We experiment with 7 diverse tasks, ranging from review rewriting to math reasoning, demonstrating that our approach outperforms direct generation. In all tasks, outputs generated with SELF-REFINE are preferred by humans and by automated metrics over those generated directly with GPT-3.5 and GPT-4, improving on average by absolute 20% across tasks.

- アブストラクト

人と同じように、LLM も与えられた生成問題に対して、最初の試行で最適なテキスト（例：要約、回答、説明）を生成するとは限りません。人がテキストを改良するように、私たちは、フィードバックと改良を繰り返しながら、LLM の最初の出力を同様に改良するフレームワーク、SELF-REFINE を紹介します。主なアイデアは、LLM を使用して出力を生成し、同じモデルが自身の出力に対して多面的なフィードバックを提供することである。従来の研究とは異なり、我々の反復洗練フレームワークは、教師付き学習データや強化学習を必要とせず、単一の LLM で動作する。レビューの書き換えから数学の推論まで、7 つの多様なタスクで実験し、我々のアプローチが直接生成よりも優れていることを実証する。すべてのタスクにおいて、SELF-REFINE で生成された出力は、GPT-3.5 や GPT-4 で直接生成された出力よりも人間や自動化されたメトリクスによって好まれ、タスク間で平均 20% 絶対的に向上していることがわかります。

第 3 章

Tech. Memo

技術的なメモを記す。備忘録であり、有用かどうかは不明。

3.1 Window's Installation of R, RStudio, TeX

Windows で、RStudio 上で R を使用し、さらに、RMarkdown 文書を PDF で出力するときに、アカウントが日本語名であること、および、OneDrive の設定が問題になることが報告されている。実験結果を以下に記す。

以下は、あくまでも、一つの PC での実験結果で、普遍性があるかどうかは不明である。異なる、問題が生じた場合には、著者のホームページにある電子メールアドレスにご一報いただけると嬉しい。かなりの部分は、macOS でも同じであるが、ここでは、時々問題が起こる、Windows についてのみ記す。

3.1.1 環境について

いずれは、Windows 11 についても、調べてみたいが、以下は、Fujitsu LIFEBOOK A577/R 上に、Windows 10 を載せたモデルである。

3.1.1.1 Command Prompt: `systeminfo`

- OS: Microsoft Windows 10 Home
- OS Version: 10.0.19044 N/A ビルド 19044
- System Model: FMVA18005

Windows は、Windows 10/11 それぞれに、非常にたくさんの種類があり、さらに、バージョンによっても、変わるため、すべてのシステムに対応することを確認することはできない。問題が出たときに、一つ一つ対応する

以外に、方法はないように思われる。

3.1.2 目標とすること・トラブル概要

この項目では、管理者権限があるアカウントの場合について記する。管理者権限がない場合は、別項目を参照。

1. R をインストール
    - The Comprehensive R Archive Network からダウンロードし、すべて初期設定（default）でインストール
2. RStudio をインストール
    - Posit: Download RStudio から、RStudio Desktop Free を選択すると、RStudio Desktop のページに行き、R をまずは、インストールするように指示があり、その右から、ダウンロードできるようになっていますから、すべて初期設定（default）でインストール
3. RNotebook を新規作成することで、いくつかのパッケージをインストール
    - RStudio を実行、New Project を作成し、File から `New File > R Notebook` を選択。この段階で、インストールが必要だと表示されるので、インストール。ファイル名を付けて保存。Preview を実行。Code Chunk を実行して、Preview。
    - RStudio が提供する、テンプレートを使うことで、それ自体には、問題がないので、確認がしやすい。自分で作成した、R Notebook などを使う場合は、その中に、他のエラーが含まれている場合もり、問題が複雑になるので、上記の方法をお勧めする。
4. RNotebook を PDF で出力
    - R Notebook の knit より、`knit PDF`。TeXLive などが導入されていなければ、最初は、TeX System が見つけられないと出る。
    - Console で、`tinytex::install_tinytex()` を実行してから、`knit PDF`
    - 3 と同じ理由で、R Notebook のテンプレートを変更せずに、実行することをお勧めする。エラーの背後にある問題を切り分けるためである。

RNotebook を新規作成することで、かなりの数の、Package をインストールすることになり、パッケージインストールに関して、確認ができる。

PDF で出力することで、TeX (実際にはその総合開発環境の TeXLive) 環境の確認が可能である。加えて、個人的には、日本語環境の確認をすることにしているが、TeX 環境が適切に稼働していれば、適切に、PDF も作成できるので、日本語環境の問題は、別項目とする。

上記のプロセスで、全く問題がなければ、この項目でのトラブルシューティングは不要。以下簡単なメモ。

1. R のインストールでは問題は起きないと思われる。ただし、言語を英語を選択しても、システムの言語が日本語だと、日本語になるようだ。

2. RStudio は、通常のものは、管理者権限がないとインストールできない。管理者権限がない場合は、当該項目を参照。

3. Package を、経験的には、27 個ほど、インストールするが、この時点でエラーが起こる場合もあるようだ。`Tools > Global Option > Packages` から、Repository を設定すれば、問題がないようである。ネット接続に問題がある場合は別対応。OneDrive で、HomeDirectory も、バックアップしている場合は、問題が起こるようである。対応は下に記する。

4. PDF 作成には、TeX システムが必要であるが、TeX システムは、環境変数 Path に書き加えて、管理するが、このときに、Path に、日本語などの 2 バイト文字が含まれると、問題が起こる。その回避の方策は下に記する。

3.1.3 PDF の作成について

上にも書いたように、基本的に、TeX システムが適切に動いているかどうかが鍵となる。TeXLive のシステムを、PC にインストールできていれば良いので、R の Package tinytex よりも、利用者も多く、ネット上の情報も多いが、全てをインストールするには、6GB 程度のディスク容量が必要である。TeX を RMarkdown など、R 関連以外でも使う場合は別として、このディクス使用量は大きいので、tinytex を利用するのがおすすめである。TeXLive のインストールは、主要なものだけを、選択すれば、使用するディスク容量も減るが、それでも、2GB 近く必要なので、tinytex を使って、必要なもののみ、インストールしていく方法が、お勧めである。環境変数を確認しながら、インストールしていくのが望ましいが、一応、それをしなくても、可能なようなので、まずは、その方法から書く。

3.1.3.1 `tinytex, TinyTeX`

紛らわしいが、これら二つは別物である。`tinytex` は、TeXLive を管理する、R のパッケージ。`TinyTeX` は、TeXLive の最小版である。そこで、R では、tinytex を使って、TinyTeX をインストールしたり、アンインストールしたりすることになる。

トラブルは、TinyTeX をインストールする段階と、これを使って、PDF を作成する段階と 2 箇所で起こりうる。

まずは、RStudio で、R Notebook を利用しようとすると、少し待つことになるが、`tinytex` が自動的にインストールされる。(右下の窓枠の Packages を見ると、最初は入っていないが、R Notebook を利用しようとして、インストールを許可すると、tinytex がチェックは入っていないが、リストには加わっていることが確認できる。)

PDF を作成しようとすると、TeX システムが見つからないと出、TinyTeX をインストールすることが最初の

オプションとして示される。

```
tinytex::install_tinytex()
```

コンソールで実行する。すでに、TeXLive パッケージが入っていたり、以前、インストールした、TinyTeX の残骸（アンインストールしても、Path が残っているなどして）がある場合には、エラーになる。エラーが出ない場合も

```
tinytex::is_tinytex()
```

と、コンソールで実行すると、`TRUE` または `FALSE` と出る。

`TRUE` なら、R Notebook から、PDF を作成する。PDF が作成できれば、問題なし。アカウント名が日本語だったり、OneDrive の設定によっては、ここで、PDF が作成できない状態が生じる。

3.1.3.2 問題解決法

わたしの環境では、さまざまな設定を変えて実験したが、かならず問題が解決するというわけではないと思われることを言及しておく。以下の命令は、`tinytex` のマニュアルによる。

1. 問題が起こったら、TinyTeX をアンインストールする
   - `tinytex::unstall_tinytex()` をコンソールで実行
2. サインアウトし、サインインしなおす
   - 環境変数（Path）の設定に、反映される必要がある。
3. C ドライブの新しいディレクトリーにインストールする
   - `tinytex::install_tinytex(dir = "C:/myTinyTeX")` をコンソールで実行
   - `myTinyTeX` は自由に決めて良いが、C ドライブに、新しくディレクトリーを作成して、そのディレクトリーに、インストールを実行するものである。もし、すでに、ディレクトリーを作成した場合は、`tinytex::install_tinytex(dir = "C:/myTinyTeX", force=TRUE)` とする。
4. サインアウトし、サインインしなおす
   - 環境変数（Path）の設定に、反映される必要がある。
5. R Notebook から knit PDF を利用して、PDF を作成する。

3.1.3.3 問題が解決しないとき

丁寧に、エラーメッセージを確認するしか基本的に方法はないが、何度も、インストール、アンインストールを繰り返していたり、他の方法も含めて、試している場合には、環境変数が適切に書き換えられない場合もあるようなので、以下の確認をお勧めする。

- Windows の検索ボックスに、set などと入力して、設定を開く。（他にも方法はたくさんある。）

- 設定の検索窓に、kan などと入力、または環境変数の編集と入力すると、「環境変数の編集」を選択できる。
- 上の段に、自分の環境変数雨、下に、システム環境変数が表示される。設定を開くときに、右クリックなどで、管理者として実行をすると、下の段のシステム環境変数を書き換えることができるが、トラブルが生じる場合もあり、現在の作業には、不要。
- 環境変数（上の段の自分の環境変数）の Path を確認する。長いと、右端まで確認できないが、ダブルクリックすると、Path 上の、ひとつひとつのディレクトリーが各行に表示され、確認がしやすくなり、編集も可能になる。なお、一続きの場合には、; が、区切りになている。
- 環境変数の Path は初期状態では、次のようになっている。'UseName' がそのアカウント名である。ここに、日本語が入っていると、TeX や、TinyTeX はインストールできない。
  - `Path: C:\Users\'UserName'\AppData\Local\Microsoft\WindowsApps`
- TinyTeX をインストールすると上の Path の後ろに、以下のようなものが追加される。'myTinyTeX' の部分が、実際にインストールされた場所になる。
  - `C:\myTinyTeX\bin\windows`
- 注：`\`, `/`, `¥` で混乱した方もおられるかもしれないが、Windows システムでは、通常、/ forward slash を使うが、Unix などのシステムでは、\ を使い、日本語システムでは、\ が ¥ マークに置き換わる。Windows では、場合によって、両方が使われている。どのような区別がされているかは、理解できていません。
- R や RStudio や texlive 以外にも、さまざまなものをインストールすると、通常は、`C:\Users\'UserName'\AppData\Local\M` にインストールされますが、texlive のように、システム全体に関わる作業をするものは、別の場所に、インストールされ、その Path が追加されます。どんどん、増えていき、非常に複雑になる場合もあります。一般的には、後ろに追加されます。問題が起こる場合は、Path をみて、TinyTeX 関連の Path がどのように記述されているかをみて、その部分だけ、削除し、一旦、サインアウトし、もう一度、サインインしてから、実行してみてください。
- エラーメッセージ（左下のいくつかのタブにエラーメッセージが表示されます、また、TeX 関係は、右下の Files のもとに、.log とついたファイルが生成されます）を保存して、詳しい方に質問するのが良いでしょう。コンピュータの設定自体を聞かれたら、コマンドプロンプトで `systeminfo` としたり、R で `Sys.getenv()` として表示されるものに、情報が含まれています。
- Console で、`Sys.setenv(LANG = "en")` と英語に切り替えると、エラーメッセージが英語で出されまう。Google 検索などで、そのエラーメッセージで検索すると、解決策が得られることがあります。日本語では、コミュニティが小さいので、解決策が限られます。日本語に戻すときには、`Sys.setenv(LANG = "ja")` です。
- なにかプログラムをインストールするときは、アンインストールする方法も確認しておくことをお勧めし

ます。Windows のアプリは、設定の、アプリの項目から、当該のアプリをみつけ、右クリックすると、アンインストールできるようになっています。その後には、必ず、サインアウトをして、サインインしなおしてください。

どうしても解決策が見つからないときは、別のアカウントを作成して、インストールすることをお勧めします。

3.1.4 別のアカウントを作成する方法

1. 設定のアカウントから、新しいユーザを作成を選択
2. その他のユーザーをこの PC に追加
3.「このユーザーのサインイン情報がありません」を選択、実行
4.「Micorsoft アカウントを持たないユーザを追加する」を選択、実行
5. ユーザー名、パスワードを設定、その下の質問三つを選択して回答
   - 実験の場合は別として、ユーザー名は、ローマ字名（半角）にしてください。
   - 新しく作成すると、忘れることが多いので、記録を残す。
6. アカウントを選択し、アカウントの種類の変更から、管理者を選択

3.1.5 非管理者として R と RStudio をインストール

- R は管理者の場合と同じようにして、インストールできます。ただ、管理者ではないので、全てはインストールできないことの注意が表示され、さらに、インストールする場所を聞かれます。初期設定では、`C:\Users\'UserName'\AppData\Programs` の中のディレクトリーが指定されます。
  - 古いバージョンのものが必要になったり、もとのものを残して最新のものを試すなどにも使えます。管理者の場合に、すべてのユーザ向けではなく、自分用、または、実験用にインストールするときは、インストールするディレクトリーを、以下のものを参考にして設定してください。
- RStudio は、通常のものは、インストールできません。同じページ RStudio Desktop の、下の方に、All Installers and Tarballs とあり、その下に、Zip/Tarballs とありますから、そこから、最後に Zip とついているファイルをダウンロードしてください。ダウンロードしたものを選択し、すべてを展開を選択して、展開します。すると、RStudio.exe が見つかると思います。エクスプローラの表示から、拡張子にチェックをいれると、exe まで確認できると思います。
  - ダウンロードフォルダにある、Zip ファイルと、展開したものが紛らわしいので、Zip ファイルのほうは、展開後、消去しておくことをお勧めします。また、Windows のデストップで、右クリックす

ると、新規から、ショートカット作成を選び、RStudio のショートカットを作成しておくことをお勧めします。

あとは、他の場合と同様です。

3.1.6 TeXLive のインストール

- 自分のアカウントを確認します。ファイル・エクスプローラから C: Users（ユーザー）の自分のアカウントを見て、半角英数になっていれば、問題ありませんが、もし、全角文字が使われていたら、インストールはできません。上に書いた方法で、別の管理者アカウントを作成し、そのアカウント名で、インストールしてください。そうすれば、自分のアカウントに戻って、利用することが可能なはずです。
  - アカウントを移動するのは、面倒ですから、その場合は、Full インストールしておくことをお勧めします。`tlmgr` という、TeXLive Manager で、不足しているパッケージなどをインストールするときに、アカウント名の問題が生じる可能性があるからです。
- TeX Live on Windows に行き、ここでも推奨している、install-tl-windows.exe からダウンロードし、インストールします。「Windows によって、PC が保護されました。」というメッセージが表示されますから、詳細情報をクリックして、実行します。このあと、しばらくして、「高度な設定」と下にある画面が表示されます。ハードディクス容量が十分ああり、かつ、設定に自信がない場合は、すべてインストールすることをお勧めします。
  - スキームが初期設定では、full スキームとなっていますから、変更として、basic スキーム（plain および latex）を選択し、言語を、日本語と英語、LaTeX 推奨パッケージを追加しておけば、ほとんど問題ありません。これだと、2GB 程度だと思います。ただ、一つだけ、足りないものがあります。
  - Windows の検索ボックスに、tex と入れて、TeX Live マネージャーを立ち上げ、`framed` を、すべての中から検索して、インストールしておきます。
  - 管理者権限で、コマンドプロンプトから、`tlmgr update --self`、次に、`tlmgr install framed` としても、インストールできます。この、`tlmgr` を使いたくないときは、全てをインストールしておくのが良いでしょう。

3.1.7 OneDrive の設定について

初期設定のままで使っていると、OneDrive にバックアップされる、設定になっていることが多く、ファイルが多くなってくると、最初の 5GB を越してしまうため、別契約をすることになります。そのようにして使っておられる方もおられると思いますが、設定の部分（Home）が、OneDrive 上にある場合は、インストールがうまくできない場合があります。さらに、RMarkdown などで、文書を作成するときには、一時的にファイルが作成され、エラーがない場合はあとで削除、エラーが出ると、それを残すというような作業が行われるため、バッ

クアップをしていると、どんどん、そのファイルが増えていきます。そこで、これらのためには、OneDrive を使わず、別の、場所で作業をすることをお勧めします。大切なファイルは、作業後、OneDrive にコピーするなり、Git-GitHub-RStudio 連携などを使うことをお勧めします。ただ、Git-GitHub-RStudio 連携は少し難しいので、興味のあるかたのみとします。

まず、ファイル・エクスプローラを開くと、OneDrive と書かれた、Directory が見えます。その中のファイルが基本的に、OneDrive にバックアップされているものです。

OneDrive を左クリックすると、設定がひらけます。基本的には、デスクトップ、ドキュメント、フォトをそれぞれ、OneDrive でバックアップするかという設定になっています。設定をやめても、削除はされませんが、その後の変更は反映されません。

すでに、OneDrive を使っている場合には、そのバックアップを停止したときに、どのようなことが起こるか確認が必要ですので、C ドライブ内に、R 用の、Directory を作成し、そこで作業するのが良いのですが、さまざまな設定が、Home ディレクトリにあり、それが、バックアップされている場合には、他のディレクトリに、たとえば、プロジェクトを作成したり、Library のディレクトリを指定しても、問題が解決しない場合が多いようです。その場合は、まずは、他のアカウントを作成し、そちらで作業をすることをお勧めします。

そのあとで、バックアップするディレクトリを整理して、OneDrive のバックアップで、デスクトップや、ドキュメントのバックアップをしているのを止め、OneDrive 内に、新たに、ディレクトリを作成して、バックアップしたいものを、その中に移して、能動的に、バックアップするディレクトリを決めるのがよいと思います。

個人個人で状況が異なりますので、この程度にしておきます。

3.2 Git-GitHub-RStudio

RStudio で、Git-GitHub を利用するときのメモを書いておきます。

3.2.1 Basics

> **用語：**ワークツリー（work tree with branches）、ステージ（staging）、ローカルリポジトリ（local repository in the PC）、リモートリポジトリ（remote repository in GitHub）、アップストリームリポジトリ（upstream repository in the owner's GitHub account）、プルリクエスト（pull-request and its approval）

Git では、- と –- を使い分けるので注意が必要。- は、コマンドオプション、-- は、コマンドオプションと、コマンド変数を分けるためのもので、- のあとは、一文字。

- Help：`git -help COMMAND`
  - 例：`git commit --help`

  - 例：`git -help –-no-pager branch`

- 初期化：`git init`
  - RStudio: New Project 作成時に、Check または、Version Control Git
- GitHub 初期設定：
  - `git config --global user.name "Name"`
  - `git config --global user.email EmailAddress`
  - RStudio: SSH などの初期設定をしてあれば、Version Control Git で、Set up 可能。確認：Project Option and Global Option
- 変更記録：`git add .` / `git add README.md`
  - `git commit -m "message"`
  - RStudio: Add されたものが、Git Tab 上に表示、Staged に、チェックをつけ、それから、Commit
- 状況確認：変更内容のチェック：`diff/status`
  - ワークツリーとの差分：git diff FileName
  - リポジトリーとステージの差分：`git diff -staged`
  - 変更ファイルのチェック：`git status`
  - RStudio: Git Tab のリストにあがっているものについて、Diff ボタンを押すと、表示される。
- 履歴の確認：`git log`
  - RStudio: Git Tab の History を押すと、commit のリストが表示され、その、Log を見ることができる。
- もとに戻す：`git restore`
  - ワークツリーの変更の取り消し：`git restore FileName`
  - ステージにあげた変更をワークツリーに戻す：git restore --staged FileName
  - RStudio: Staged にチェックをつけ、More の　revert を選択すると、同様なことが可能のように思われる。
- ブランチを作成：branch、ポインター HEAD
  - 作成：`git branch BranchName`

    - RStudio: New Branch から作成可能
  - ブランチの一覧表示：`git branch`
    - RStudio: New Branch の右の三角から表示、選択
  - GitHub も含めたブランチの一覧表示：`git branch -a`
    - RStudio: New Branch の右の三角から表示、選択
  - 切り替え：`git switch BranchName`
  - 新しいブランチを作成して切り替え：git switch --c BranchName
  - マージ：`git merge BranchName`
    - 難しい。RStudio では、Terminal からで、Editor（Default は、vi 慣れていなければ、`Sys.setenv(EDITOR="nano")`）などとするのが得策。）が立ち上がる。
  - GitHub 上のものをマージ：`git merge orign/main`
  - コンフリクト conflict: 最終的なものを Editor で残す。
- GitHub Origin の追加：`git add remote orign`
  - `git add remote origin URL`
  - `git add upstream origin URL` なども可能
- Push/Pull (Fetch+Merge)/Fetch:　リモート名　ブランチ名
  - `git push orign main`
    - RStudio: Push ボタン
  - `git pull origin main`
    - RStudio: Pull ボタン
  - `git pull`（省略可能）
  - `git fetch origin`（リモート名）confilict が起こりそうな時は、fetch + merge
    - 変更途中の時は、fetch + merge 、main に取り込みたい時は、pull
  - `git fetch --prune`
  - `git remote prune origin`
    - RStudio では、Git Tab の一番右の、Refresh ボタン

- Pull Request の手順
  - main を最新の状態に更新: 例：This branch is 2 commits behind icu-hsuzuki:main.　Synk fork: This branch is out-of-date > Update branch
    - `git branch`
    - `git pull origin main`
  - ブランチを作成
    - `git switch -c BranchName`
  - ファイルを修正・コミット
    - `git add .`
    - `git commit -m "message"`
  - Push
    - `git push origin BranchName`
  - Pull Request
    - @GitHub: Pull Request > New Pull Request
    - base: main <- compare: BranchName（Comparing changes で、変更を提案している Branch-Name を確認）
    - Create Pull Request>　Add Title, Comment > Create Pull Requst ボタン
    - File Changed を確認
    - Reviewer を追加しておいてみてもらう。
    - Merge Pull Request
    - Delet Branch: git branch -d BranchName (-D force)
    - Code で確認
  - Code Review
  - Merge

3.3 Removing Large Files

- 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
- Reset changes: git reset --hard SHA-Name

3.3.1 Simple Pull Request

1. Log in to your GitHub account
2. Go to the respository you want to comment
3. Fork the repository
4. Clone the repository (forked from the original) using ssh
   1. git clone git@github.com:account/repo
   2. Enter ssh permisssion
   3. less FileName
5. git checkout -b Branch-Name
6. edit the file
7. git status

8. git diff
9. git add FileName
10. git commit -m “short message”
11. git push origin Branch-Name
12. In GitHub acount: Compare & pull request
13. Add a short comment
14. Push Create a pull request

3.3.2 Owner Side Approval

1. See commit, file changed
2. Merges pull request
3. Confirm merge

3.3.3 参考資料

- GitHub Training Kit: CheatSheet や、日本語ドキュメントもある
- Package usethis 2.0.0: Tidyverse
  - Happy Git and GitHub for the useR
- Removing Large Files
- Version Control with Git and SVN: Posit Site
  - Managing – Part 2 (Github and RStudio) [Video Lecture 48 min.]

第 4 章

Windows

わたしは基本的に、Mac User なので、Windows のことは、あまり詳しくない。質問されることもあるが、常時つかっていないと、忘れてしまうことが多いので、Windows について、学んだことはその都度、備忘録として、書いておくことにする。

名称などは、正確ではないかもしれないが、ある程度共通の用語を持っていないと、質問すらできないので。

4.1 一般（GUI）Graphical User Interface

4.1.1 設定 Setup

- 環境変数：システム環境変数（S）と、環境変数（U）がある
  - 環境変数 (U)：OneDrive の Path、Path、TEMP、TMP が含まれる
    - C:\Users\‘User’\AppData\Local\Microsoft\WindowsApps;C:\ 以下続く
    - 追加される時はつねに最後に加えられる（システム環境変数も同様）

4.1.2 Windows 10

- Windows キー（以下 [Win]）を左クリックで、検索とメニューが出るのでそこで洗濯も可。場合によっては、右クリックすると、Option を選択でき、管理者で起動できる場合もある
- [Win] + [R]（[R] は R キー、プラスは押しながら、以下同様）：Run Box と呼ばれる検索窓が出る。参照から、ファイルなどを開けることも可能。
- [Win]+[X]：[Win] を右クリックも同じ
- 下にあるメニューバーのようなもの（全体がタスクバー）の各部の名称について（設定可）

  - 一番左：スタートボタン　Start Button
  - スタートボタンの上：システムアイコン System Icon 電源ボタン、アカウントなど
  - 左から二番目：検索ボックス Search Box
  - 左から三番目：Task View
  - 左から四番目：Pinned Apps
  - 右：通知領域 Notification Area 以前はタスクトレイ
    * 一番右は、通知だが、その右には、時刻と日付、言語、スピーカー音量、wifi、電源、OneDrive などが表示される。その左に天気

- Reference:
  - Windows Server documentation
    * Windows Commands

4.2 コマンドプロンプト Command Prompt

- 検索ボックスに cmd と入れると見つかる。右のメニュには、管理者として実行 Run as administrator もある
- 通常は、Case Insensitive 大文字と小文字を区別しない
- 通常は、**C:\Users\[Username]>** と表示される。日本語システムでは **C:¥Users¥[Username]>**
- 基本コマンド：
  - `cls`：クリアスクリーン clear screen
  - `[Ctrl]+[C]`：実行の中断 stop command - interrupt running command
  - **`-help`** – ヘルプ（ある場合のみ）の表示　Provides a Guide to other Commands
    * 通常は、`/?` を利用する。例：`dir /? del /?`
  - `dir`: 現在のディレクトリの内容 Lists Items in a Directory
  - **`chdir`** or **`cd`**： ディレクトリの移動 Changes the Current Working Directory to the Specified Directory
  - **`mkdir`**：ディレクトリの作成 Creates a Folder

– **rmdir**：ディレクトリの削除 Deletes a Folder

– **del**：ファイルの削除 Deletes a File

– **move**：ファイルやフォルダの移動 Moves a File or Folder to a Specified Folder

– **ren**：ファイル名の変更 Renames a File with the Syntax

  * **ren filename.extension new-name.extension**

– **tree**：ディレクトリー・ツリーの表示 Shows the Tree of the Current Directory or Specified Drive

– **echo**：メッセージなどを、ファイルなどに出力 Shows Custom Messages or Messages from a Script or File

  * 例：`echo "Hello World!" echo hello world > hello.txt`

– **more**：ファイルの内容を表示 Shows More Information or the Content of a File

  * 例：`more hello.txt`

– **ver**：Windows のバージョンを表示 Shows the Version of the OS

– **systeminfo**: コンピュータについての情報 Shows Your PC's Details

  * ホスト名、OS 名、OS バージョン、OS 製造元、OS 構成、OS ビルドの種類、登録されている所有者、登録されている組織、プロダクト ID、最初のインストール日付、システム起動時間、システム製造元、システムの種類、プロセッサ、BIOS バージョン、Windows ディレクトリ、システムディレクトリ、起動デバイス、システムロケール、タイムゾーン、物理メモリの合計、利用できる物理メモリ、仮想メモリ、ページファイルの場所、ドメイン、ログインサーバー、ホットフィックス、ネットワークカード、Hyper-V の要件

– **set**：環境変数の表示 Shows your PC's Environment Variables

– **clip**：クリップボードにコピー Copies an Item to the Clipboard

  * 例：`dir | clip` 現在のディレクトリの情報をクリップボードにコピー copies all the content of the present working directory to the clipboard.

– **assoc**：プログラムと拡張子を表示 Lists Programs and the Extensions They are Associated With

  * 例：`fc “file-1-path”“file-2-path”`

– **fc** – にているファイルの内容比較 Compares Two Similar Files

– **tasklist**：開いているプログラムの表示 Shows Open Programs

– **taskkill**：開いているプログラムを終了させる Terminates a Running Task

    - 例：To kill a task, run `taskkill /IM "task.exe" /F`. For example, `taskkill /IM "chrome.exe" /F`:
  - **`exit`**：コマンドラインを終了 Closes the Command Line
  - **`shutdown`**： コンピュータのシステム終了など Shuts down, Restarts, Hibernates, Sleeps the Computer
    - 例：`shutdown` とするとオプションが表示される（`shutdown /?` と同じ）
  - **`netstat -an`**：ポートの状況を表示　Shows Open Ports, their IP Addresses and States
  - **`ping`**：ウェッブサイトへの接続時間など Shows a Website IP Address, Lets you Know How Long it Takes to Transmit Data and a Get Response
    - 例：ping “icu-hsuzuki.github.io”
  - **`ipconfig`**： コンピュータのインターネットアドレスの情報の表示 Shows Information about PC IP Addresses and Connections
  - `powercfg help`: 制御設定のためのヘルプ　Controls Configurable Power Settings Help
  - `powershell start cmd -v runAs`：管理者として実行 run as administrator 確認画面が出る
  - **`sfc`**：[管理者のみ実行可] システムファイルの状況を確認 System File Checker
  - **`driverquery`** –インストールされている、ドライバーのリストを表示 Lists All Installed Drivers

4.3 パワーシェル PowerShell

- 通常は、Case Insensitive 大文字と小文字を区別しない
- Introduction to PowerShell

4.4 ギットバッシュ GitBash

- Git をインストールするときに同時に、インストールできる

4.5 WSL - Windows Subsystem Linux

- PowerShell を管理者で起動
- Enable-WindowsOptionalFeature -Online - FeatureName Microsoft-Windows-Subsystem-Linux

– だいぶ時間がかかり再起動：上になにやら linux をインストールしていそうなマークが現れる。

– 検索ボックス：store > Search Window: ubuntu [入手] だいぶ時間がかかる

  * [開く] Ubuntu: Installing, this may take a few minutes...

– Windows system for windows: ON

– 検索ボックス：ubuntu

– WSL をアンインストールする場合 uninstall: lxrun /uninstall /full

第 5 章

Quarto and Quarto Book

- Quarto Book Examples:
  - R4DS 2e: https://r4ds.hadley.nz,
    - Source: https://github.com/hadley/r4ds/
    - Current Version 1e: https://r4ds.had.co.nz
  - Python for Data Analysis 3e: https://wesmckinney.com/book/
    - Source: https://github.com/wesm/pydata-book/tree/3rd-edition
  - Visualization Curriculum: https://jjallaire.github.io/visualization-curriculum/
    - Source: https://github.com/jjallaire/visualization-curriculum
- Welcome to Quatro: https://quarto.org/
  1. Download Quatro CLI
  2. Choose your tool and get started: RStudio
- Documentation:
  - Tutorial: Computations: https://quarto.org/docs/get-started/computations/rstudio.html
  - Tutorial: Authoring: https://quarto.org/docs/get-started/authoring/rstudio.html
  - Books: https://quarto.org/docs/books/
  - Book Options: https://quarto.org/docs/reference/projects/books.html
- 2023.03.20: Created a Book as test

- Engine: Knitr or Jupyter > choose Knitr
- Options: [ ]create a git repository, [ ]use renv with this project, [x]use visual markdown editor (the last one is checked as default)
    - Checked all and open in new session
- quarto.yml: Original
    - project:
    - type: book
    - project:
    - type: book
    - output-dir: docs
- Git: commit changes
- In Terminal: `touch .nojekyll`
    - git remote add origin git@github.com:icu-hsuzuki/myds.git
    - git branch -M main
    - git push -u origin main
- @GitHub
    - Setting > Pages > Branch: Select 'main', '/docs', and then Save
    - About at Geer mark: [x]Use your GitHub Pages website
- Add new chapters in `_quarto.yml`

第 6 章

To Do List

6.1 データサイエンスをはじめましょう

- RStudio で R
  - Windows について
  - 日本語環境について

6.2 データサイエンスを教えませんか

6.3 わたしのデータサイエンス

- Quarto について vs RMarkdown

第 7 章

Summary

まとめ。あとがきなど。

References

Knuth, Donald E. 1984. “Literate Programming.” *Comput. J.* 27 (2): 97–111. https://doi.org/10.1093/comjnl/27.2.97.