

初心者セッション： データサイエンスのための リーダブルコードのススメ

BeginnerR Session: Readable Code for Data Science

2nd September 2023, Tokyo.R #108

Yuta Kanzawa @yutakanzawa



Data Scientist at Zurich Insurance Company Limited, Japan Branch



神沢雄大 Yuta Kanzawa

- データサイエンティスト@チューリッヒ保険会社
 - 日本支店
- Twitter: [@yutakanzawa](https://twitter.com/yutakanzawa)
- 好きなもの：オペラとワイン
 - ワーグナー
 - ブルゴーニュ (WSET Lv 3→?)
- 使用可能言語：7
 - 人間：日本語、英語、ドイツ語
 - コンピューター：R, Python, SAS, SQL



アジェンダ

- 今日話すこと
 - データサイエンスのコーディング（一部はショッキングかも）
 - ポエム
- 今日話さないこと
 - 目から鱗な話
 - コードそのもの

おことわり

- 「データサイエンス」、「データサイエンティスト」という主語の大きな話をしますが、原則としてスピーカーの実体験に基づいたものです。
- 事例を一般化するように努めていますが、全てのデータサイエンティストが当てはまる訳ではありません。コードを書くという点で、模範となるデータサイエンティストも数多く存在します。
- 開発環境やエディタ、IDEそのものについては割愛します。

TL;DR

- 読みにくいor保守性が低いコード = 自分やチームの足枷
- しかし、データサイエンティストにとってコードは手段。
 - 「動けばOK」という文化はなくなる。
- データサイエンティストがエンジニアを見習うべきポイント：
 - 処理の内容や機能に応じて、フォルダやコードを分割。
 - コメント付け、リファクタリングの実施、フォーマッターの使用。
 - チームと協力して習慣づける。
 - レビューし合う。

なぜデータサイエンティストも
読みやすく保守性が高いコードを
書く必要があるのか。

Why readable codes also for data science?

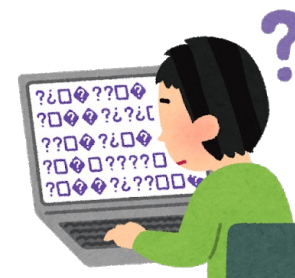
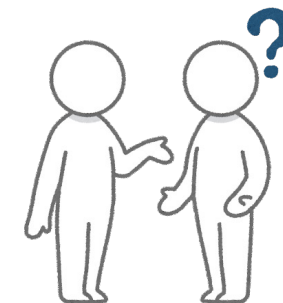
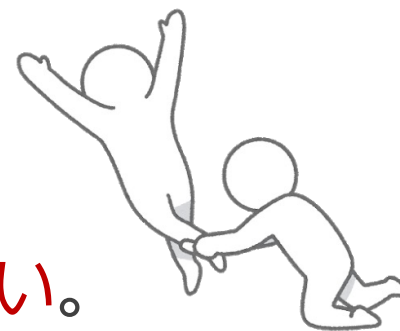
いきなりですが、質問！

- 今日の朝食のメニューは？
- 今の服の色は？
- 今夜の懇親会の飲み物は？



落とし穴ワーストスリー

- **未来の自分は他人**（だと思った方がいい）。
 - コードやドキュメントに**書いていないことは覚えていない**。
 - 過去（今）の自分が**足を引っ張る**可能性大。
- **来年も自分が作業するとは限らない**。
 - 異動、昇進 → 後任からの質問対応 → 結果的に**属人化**
- **他人の書いたコードを「解読」**しないといけないこともある。
 - 不十分な引き継ぎ
 - **秘伝のタレ**
 - もういない → **本人に聞けない**。



コーディングにおける エンジニアとの違い

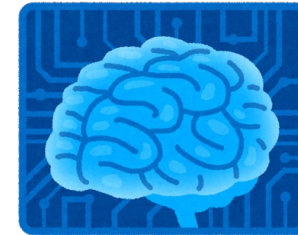
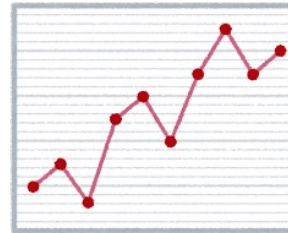
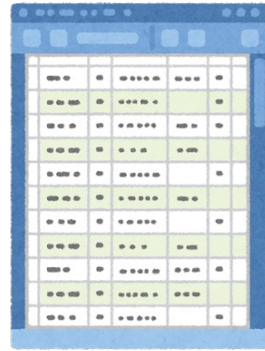
Differences in coding from engineers

おことわり（再掲）

- 「データサイエンス」、「データサイエンティスト」という主語の大きな話をしますが、原則としてスピーカーの実体験に基づいたものです。
- 事例を一般化するように努めていますが、全てのデータサイエンティストが当てはまる訳ではありません。コードを書くという点で、模範となるデータサイエンティストも数多く存在します。
- 開発環境やエディタ、IDEそのものについては割愛します。

データサイエンティストのアウトプット（最終成果物）

- データセット、数値
- グラフ
- モデル
- 考察



→ コードはアウトプットを作成する**手段**。

一度きりのコードを書く割合が高い。

（でも、一度きりのはずがそうでなくなることも...）



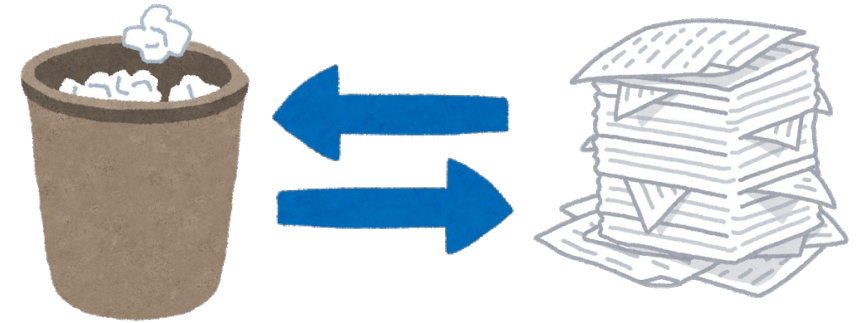
データサイエンティストにとってのコードの位置付け

- 動くコードが正義。
 - よく分からないけど期待通りの結果が出ている。
 - 動かすのが面倒だけど期待通りの結果が出ている。
 - キレイな書き方ではないけど期待通りの結果が出ている。
- 動くコードが完成したら、Mission completed!
 - テスト、なにそれ、おいしいの？
 - フォーマッター、なにそれ、おいしいの？
 - リファクタリング、なにそれ、おいしいの？



データサイエンティストの作業形態

- 分析の恥は**書き捨てる**。
 - **動くコード**を目指して多数の試行錯誤。



- 目的のためには**手段を選ばない**。
 - ネットからコピペしたコードのつぎはぎ
 - 複数のツールを経由する処理フロー

- 例：
Excelでデータを加工してから、RやPythonに投入、グラフを出力し、
グラフの軸ラベルはPowerPointでテキストボックスを貼って調整。



データサイエンティストが エンジニアを見習うべきポイント

What data scientists should learn from engineers

ファイルとフォルダの構成

- 一言でいうと「役割分担」
- 分析コードあるある（アンチパターン）
 - データ読込から結果出力まで1つのファイル。
 - コードと入力データ、出力データが同じフォルダ。
 - プロトタイプを試行錯誤して作るのには向いているけど...
- 入力と出力、**コードとデータは別々に保管！** *1
- 処理の内容や大まかな機能に応じて、**コードを分割！** *2,3



*1 <https://socinuit.hatenablog.com/entry/2020/09/16/123811>

*2 <https://socinuit.hatenablog.com/entry/2020/10/03/173920>

*3 『リーダブルコード』第II、III部

コメント

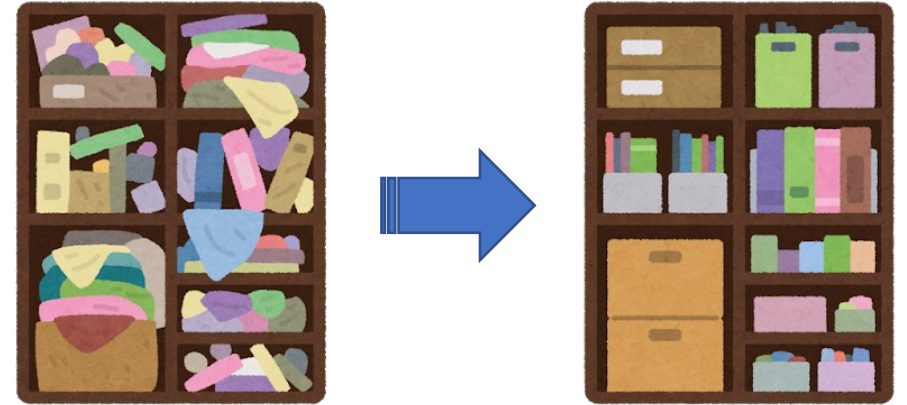
- 一言でいうと「メモ書き」
- 読みやすさの向上
 - ただし、自分では分かりやすいと思いがち。
- 例：
 - 変数、処理、関数の説明*
 - 補足事項
 - 今後の課題の記述
 - 「TODO」



* 『リーダブルコード』5、6章参照

リファクタリング

- 一言でいうと「コードの整理整頓」
- 保守性の向上
- 特に：
 - 変数名、関数名の見直し
 - 機能、内容に即したもの*¹にする。
 - 処理の分割（既出）
 - 関数化
 - for文、apply系関数
 - **知識の継続的アップデート**が役立つ&必要。
 - e.g. `tidyr::gather()` → `tidyr::pivot_longer()`*²



*1 『リーダブルコード』2、3章参照

*2 <https://notchained.hatenablog.com/entry/2019/06/29/234050>

コードスタイルガイド

- コードを書く際の様式のルール（名前、空白、改行、文字数、など）
 - 書式が整ったコードは読みやすい。→ デバッグや保守をしやすい。
 - 意識しながら書くのは大変 → フォーマッターやリンターの使用
- **tidyverse style guide**
 - <https://style.tidyverse.org>
 - Google's R Style Guideのオリジナル版*1がベース → styler、lintr
- Google's R Style Guide
 - GoogleのRコミュニティがまとめ、tidyverse style guideの元になった*2。
 - 現行版： <https://google.github.io/styleguide/Rguide.html>



*1 オリジナル版（スタンフォード大学のコピー）： <https://web.Stanford.edu/class/cs109I/unrestricted/resources/google-style.html>
→日本語訳（RjpWiki）： <http://www.okadajp.org/RWiki/?Google%27s+R+Style+Guide>

*2 ただし、現行版はtidyverse style guideから派生したもの。

フォーマッターやリンターの使用

- フォーマッター（見た目を整えるツール）
 - **styler**^{*1}: tidyverse style guide準拠
 - RStudioのアドイン同梱
 - formatR^{*2}
 - RStudioのアドイン : addinexamples^{*3}
 - どちらも**RStudio上でアドイン**として実行可能。
 - ショートカットキーも設定可能。
 - R Markdownでも利用可能。
- リンター（静的解析ツール）
 - linter^{*4}: 静的チェック（**バグにつながりやすいコード**をチェック）



*1 <https://github.com/r-lib/styler>

*2 <https://yihui.org/formatr/>

*3 <https://rstudio.github.io/rstudioaddins/>

*4 <https://github.com/r-lib/linter>

以上を身に付けるには。

- 漸進
 - 時間の取れる**趣味のプロジェクト**から始めてみる。
 - チームに提案して、コードの改善にかける**時間を確保**。
- 習慣付け
 - コメント付けは**付箋を貼る感じ**で、最初は**頻繁に**。
 - 命名**規則**の導入（個人またはチームで）
 - コードを書いたら
 - `styler`を実行。
 - リファクタリングしてみる。
- レビューまたは鑑賞
 - 人に**見てもらい**、人のコードを見る。
 - チーム内、ブログ記事やGitHubのコード

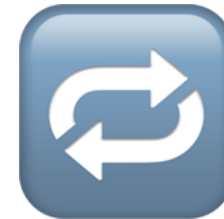


Next Steps: MLOpsを見据えて（模索中）

- データサイエンスチームのコード → プロダクション環境での運用
 - 直接の利害関係者が急拡大。
 - ユーザー、エンジニア（IT部門）、マネジメント層
 - → ちょっとしたバグが一大事に。



- 対策：再現性の確保*
 - コードのバージョン管理
 - ライブラリーのバージョン管理
 - 乱数のシード固定
 - テストの積極的導入（CI/CD）



* Valliappa Lakshmanan, Sara Robinson, and Michael Munn. Machine Learning Design Patterns. Sebastopol: O'Reilly, 2020.

まとめ

Long story short

Long story short

- 読みにくいor保守性が低いコード = 自分やチームの足枷



- しかし、データサイエンティストにとってコードは手段。
 - 「動けばOK」という文化はなくなる。



- データサイエンティストがエンジニアを見習うべきポイント：
 - 処理の内容や機能に応じて、フォルダやコードを分割。
 - コメント付け、リファクタリングの実施、フォーマッターの使用。
 - チームと協力して習慣づける。
 - レビューし合う。

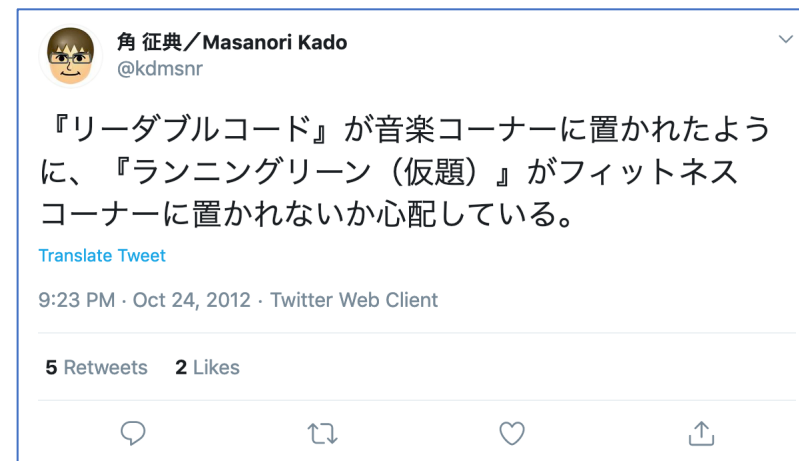
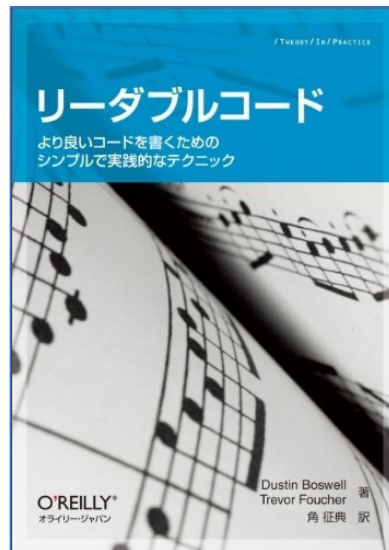


参考

- 『データ分析をちゃんと管理しよう with R【フォルダ構成編】』
 - kinuit
 - <https://socinuit.hatenablog.com/entry/2020/09/16/123811>
- 『データ分析をちゃんと管理しよう【コーディング編】』
 - kinuit
 - <https://socinuit.hatenablog.com/entry/2020/10/03/173920>
- 『Rにおけるバージョン管理とプロジェクト運用』
 - Uryu Shinya
 - https://speakerdeck.com/s_uryu/rstudio-for-team

参考（続き）

- 『リーダブルコードーより良いコードを書くためのシンプルで実践的なテクニック』
 - Dustin Boswell, Trevor Foucher, 角征典（訳）（2012）
 - <https://www.oreilly.co.jp/books/9784873115658/>



* <https://twitter.com/kdmsnr/status/261080519792553984>

Enjoy!

おまけ

Appendix

(Pythonの場合) フォーマッターやリンターの使用*1

- フォーマッター

- Black: PEP8*2準拠

- 書式が整ったコードは読みやすい。→ デバッグや保守をしやすい。
 - JupyterLabのプラグイン: JupyterLab Code Formatter*3
 - JupyterLab上でボタン1つで実行可能。すごく便利!



- リンター (静的解析ツール)

- flake8: 静的チェック (バグにつながりやすいコードをチェック)
 - mypy: 型ヒントチェック



*1 ここに挙げたツールの詳しくて分かりやすい説明 → PyCon JP 2019 ビギナーセッション『Pythonでの開発を効率的に進めるためのツール設定』<https://www.slideshare.net/aodag/python-172432039>

*2 Pythonのコーディング規約 <https://www.python.org/dev/peps/pep-0008/>

*3 <https://github.com/drillan/jupyter-black>

参考 (Pythonの場合)

- 『Pythonでの開発を効率的に進めるためのツール設定』
 - Atsushi Odagiri, PyCon JP 2019 ビギナーセッション
 - <https://www.slideshare.net/aodag/python-172432039>
- 『忙しい研究者のためのテストコードとドキュメントの書き方』
 - hmkz (2020)
 - <https://qiita.com/hmkz/items/0689cd85fb3e1adcda1a>
- 'Machine Learning Design Patterns'
 - V. Lakshmanan, S. Robinson & M. Munn (2020)
 - <https://www.oreilly.com/library/view/machine-learning-design/9781098115777/>

