データサイエンスのための リーダブルコードのススメ

13th January 2021, みんなのPython勉強会 #65
Yuta Kanzawa @yutakanzawa

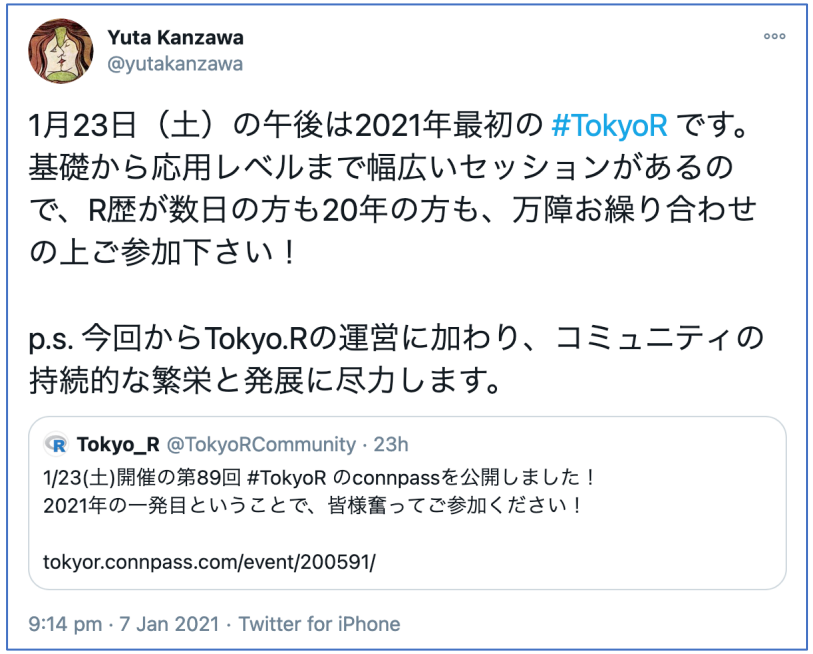
Data Scientist at Janssen Pharmaceutical K.K., Tokyo
A Family Company of Johnson & Johnson

I am...

- **神沢雄大 Yuta Kanzawa** (twitter: @yutakanzawa)
- Data scientist at Janssen Japan, Tokyo
  - A pharmaceutical company of **J&J**
- Opera & wine lover
  - Wagner
  - Bourgogne (WSET Lv 2→3)
- 7 languages
  - Human: Japanese, English, German
  - Computer: R, Python, SAS, SQL

（宣伝）Tokyo.R（R勉強会@東京）

次回：2021年1月23日（土）14-18時
https://tokyor.connpass.com/event/200591/




```
import qrcode
img = qrcode.make("https://tokyor.connpass.com/event/200591/")
img.save("TokyoR_89.png")
```

* https://twitter.com/yutakanzawa/status/1347154528076922882

アジェンダ

- 今日話すこと
  - エンジニアにとってはベタな話（でも一部はショッキングかも）
  - ポエム

- 今日話さないこと
  - エンジニアにとっては目から鱗な話
  - 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については割愛します（個人的なベストプラクティスがまだない）。

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らない。

- 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。
    - レヴューし合う。

なぜデータサイエンティストも読みやすくて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?

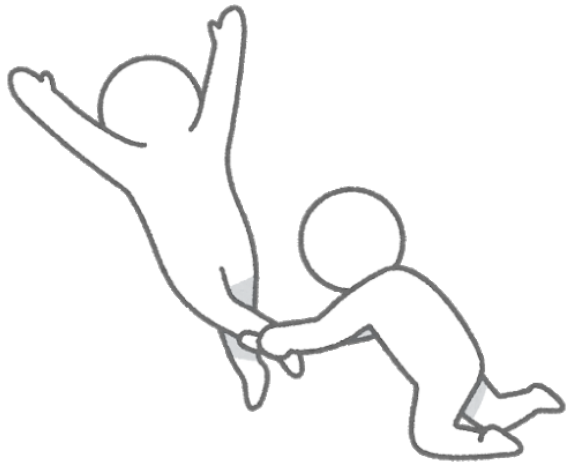
いきなりですが、質問！

- 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？

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ある以外のことは覚えていない。
  - 仕様
  - 実行時に注意すべき点
  - 今後解決すべき課題

- 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

- 異動や昇進などで自分の**手を離れる**。
  - 引き継ぎはした。

- **後任からの質問対応**に追われる。
  - 引き継ぎが足りなかったっぽい。

- 結果的に**属人化**しかねない。←今ここ
  - いつまでたっても付いて回る。

他人の書いたコードを「解読」しないといけないことも

- 引き継ぎが**不十分**。
  - ドキュメントがない。
  - コードにコメントがない。
  - バージョン管理がされていない。
- **秘伝**のタレ
  - 誰も何も知らない。
  - 何も足したり引いたりできない。
- もういない（退職、異動した）ので本人に聞けない。

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）

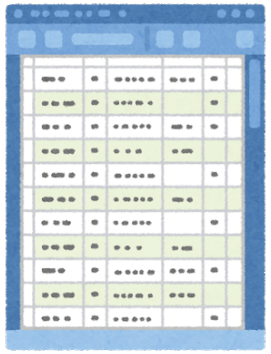
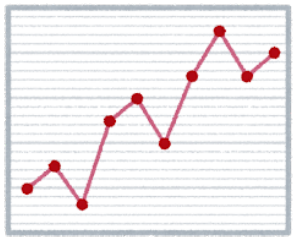
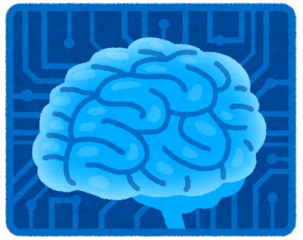
- 「データサイエンス」、「データサイエンティスト」という主語の大きな話をしますが、原則としてスピーカーの実体験に基づいたものです。

- 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については割愛します（個人的なベストプラクティスがまだない）。

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。

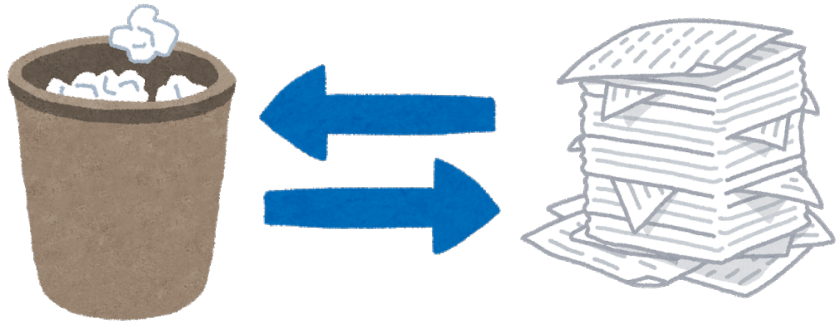
データサイエンティストにとってのコードの位置付け

- 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**。
  - 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で加工してから、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」
- Jupyter Notebookあるある
  - 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

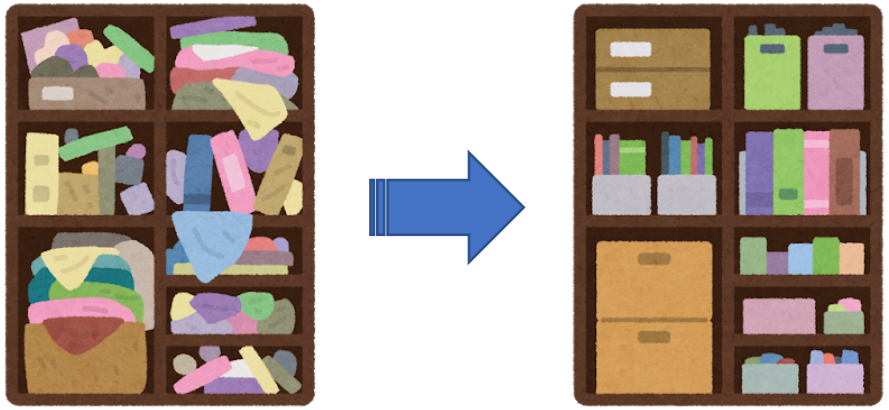
- 一言でいうと「メモ書き」
- 読みやすさの向上
  - ただし、自分では分かりやすいと思いがち。

- 例：
  - 変数、処理、関数の説明*1
  - 補足事項
  - 今後の課題の記述
    - 「TODO」

*1『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、リスト内包表記
  - 知識の継続的アップデートが役立つ。
    - e.g. f文字列、セイウチ演算子*2



*1『リーダブルコード』2、3章参照
*2 https://atsuoishimoto.hatenablog.com/entry/2019/09/03/110508

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。
    - Jupyter Notebookのエクステンション：Jupyter Black*3
      - Jupyter Notebook上でボタンorショートカットキー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

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（セルが1つ完成したら）
    - Blackを実行。
    - 少しでもリファクタリングしてみる。
- レヴューまたは鑑賞
  - 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

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らない。

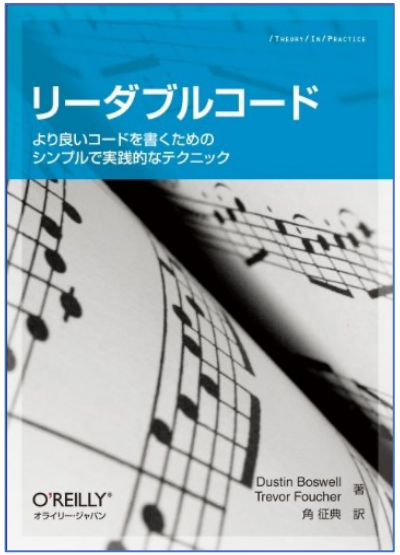
- データサイエンティストがエンジニアを見習うべきポイント：
  - 処理の内容や機能に応じて、フォルダやコードを分割。
  - コメント付け、リファクタリングの実施、フォーマッターの使用。
  - チームと協力して習慣づける。
    - レヴューし合う。

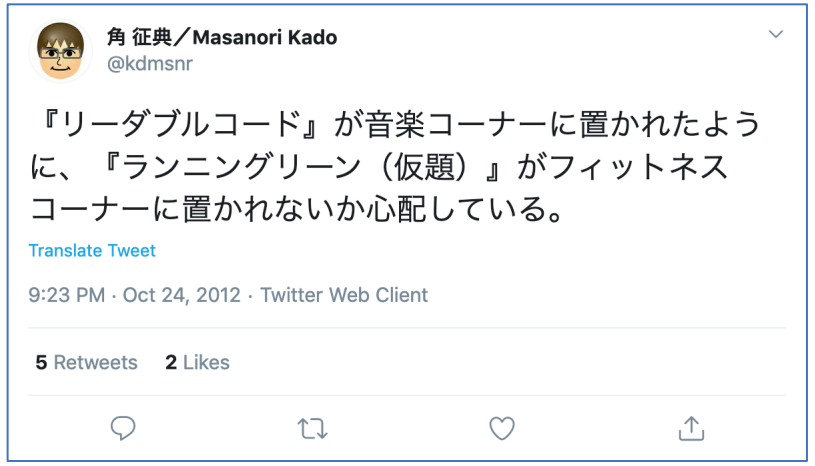
参考

- 『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
- 『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

参考（続き）

- 『リーダブルコード―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

参考（さらに）

- 『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

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

Enjoy!