

A large-scale comparative analysis of Coding Standard conformance in Open-Source Data Science projects

オープンソースデータサイエンスプロジェクトにおける、
コーディング標準適合性の大規模な比較分析

Andrew J. Simmons , Scott Barnett , Jessica Rivera-Villicana ,

Akshat Bajaj , Rajesh Vasa

(Deakin University)

ESEM2020

紹介者: 西山知弥

コーディング標準とは

- コード規約, コーディングスタイルとも呼ばれる
- インスタンス変数を宣言する場所,
クラス, メソッド, 変数に名前を付ける方法などの**ガイドライン**
- コーディング標準に準拠するソフトウェアは保守性を向上させる
- 一貫性のあるコーディング標準により, コードの可読性も向上する

背景

- データサイエンスの機械学習アルゴリズムの予測精度を向上させる研究 → 盛んに行われている
- データサイエンスのソフトウェアエンジニアリング
→ **あまり注意が払われていない！**
- 機械学習におけるアーキテクチャ上の負債が増大していると報告されている[27]

背景

- データサイエンティストは, ソフトウェアエンジニアリング以外のバックグラウンドから来ていることが多いため, 一般的なコーディング標準に従わない可能性がある
- データサイエンスプロジェクトに焦点を当てたコーディング標準の大規模な実証研究はなかった

目的

- データサイエンスプロジェクトがコーディング標準に準拠している範囲を調査すること
- 特に, **どの標準が守られ, どれが無視されているのか,**
及び**従来のソフトウェアプロジェクトとどのように異なるか**

アプローチ

- Pythonコーディング標準について,
1048のオープンソースデータサイエンスプロジェクトのコーパスを, 同様のレベルの品質と成熟度を持つ1099の非データサイエンスプロジェクトと比較
- Pythonコードのスタイルガイドには, **コードレイアウト, 命名標準, プログラミングの推奨事項, 及びコメント**に関するコーディング標準が含まれている

コーディング標準の例

- ・ **関数, 変数の名前**

小文字のみにする → `my name`

必要に応じて単語をアンダースコア (`_`) で区切る → `class_`

- ・ **コードレイアウト**

1レベルインデントするごとに, スペースを4つ使う

Python 3 では, インデントにタブとスペースを混ぜることを禁止

など

詳しくは <https://pep8-jp.readthedocs.io/ja/latest/>

コーパスの形成

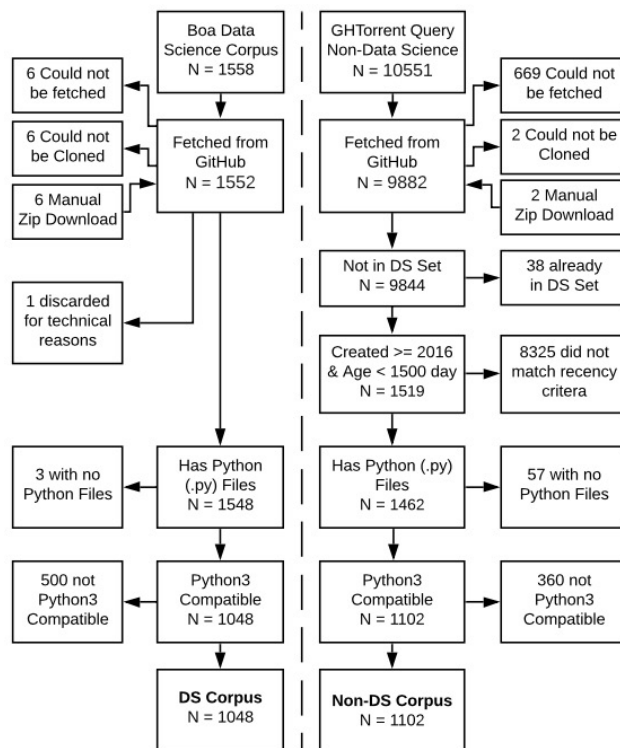


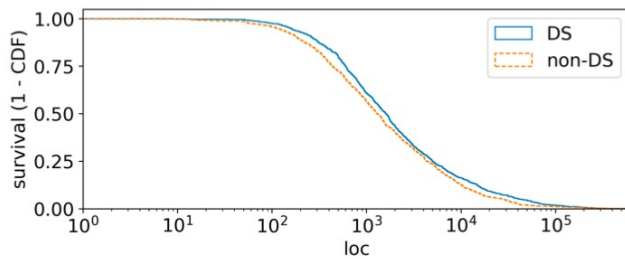
Figure 3: Curation of Projects

- プロジェクトの成熟度を考慮し,プロジェクトの経過時間をGitログの最初のコミットからデフォルトブランチへの最新のコミットまでの日数として定義し,1500日未満(約4.1歳)のプロジェクトに制限
- Python 3と互換性のないプロジェクトを削除

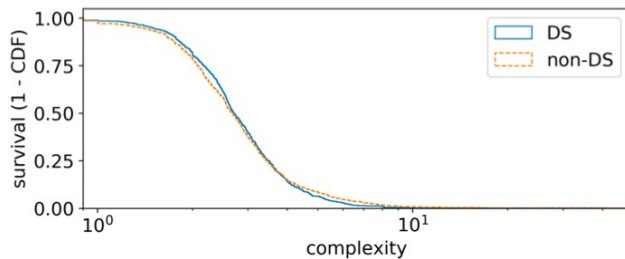
リサーチクエスチョン

- RQ1 : データサイエンスプロジェクトの現在の状況は？(成熟度, LOC, 循環的複雑度, 分布 など)
- RQ2 : 従来のプロジェクトとの間で, データサイエンスプロジェクトのコーディング標準への準拠はどのように異なるか？
- RQ3 : データサイエンスプロジェクト内のコーディング標準違反はどこで発生するか？

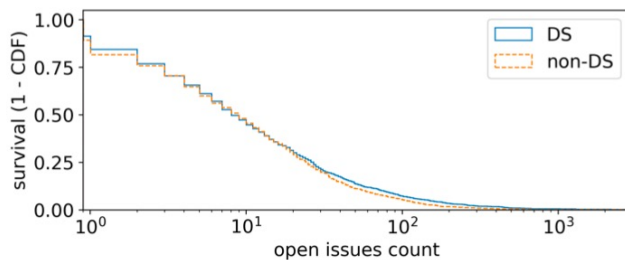
RQ1 : データサイエンスプロジェクトの現在の状況は？



- コード行が多い



- 循環的複雑度は類似(KS statistic=0.06, p=0.05)



- 未解決のissues は有意差(KS statistic=0.03, p=0.6)を示していない

RQ1 : データサイエンスプロジェクトの現在の状況は？

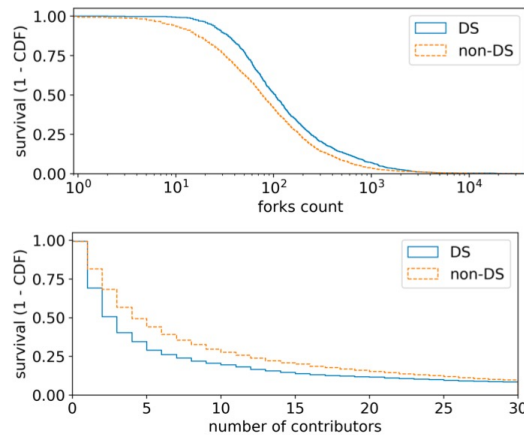


Figure 5: Survival plots showing characteristics of the projects selected for the DS corpus and the control group.

- 非データサイエンスプロジェクトよりも貢献者の数が少ないが,フォークの数は多い

→互いのコード/モデルをコピー/フォークしてから, 単独または小規模なチームで作業してユースケースに合わせて調整して開発するから？

RQ2 : 従来のプロジェクトと比較して, データサイエンスプロジェクト間でコーディング標準への準拠はどのように異なるか?

- **Pylint**を使用して, 各リポジトリ内のすべてのファイルを分析し, 空白以外の**LOC**ごとのコーディング標準違反の数を計算
- データサイエンス以外のリポジトリと比較したエラー分布の違いも調査するため, 両側のマンホイットニーU検定を行う

RQ2 : 従来のプロジェクトと比較して, データサイエンスプロジェクト間でコーディング標準への準拠はどのように異なるか?

Pylint warning (per non-blank LOC)	DS mean	DS median	Non-DS mean	Non-DS median	p-value
too-many-locals	0.34%	0.28%	0.16%	0.06%	$1.10 \cdot 10^{-59}$
import-error	2.73%	2.07%	1.83%	1.16%	$2.02 \cdot 10^{-45}$
too-many-arguments	0.33%	0.26%	0.19%	0.08%	$3.12 \cdot 10^{-42}$
invalid-name	9.88%	7.73%	6.94%	4.86%	$1.31 \cdot 10^{-26}$
broad-except	0.03%	0.00%	0.13%	0.00%	$8.94 \cdot 10^{-20}$
bad-indentation	6.58%	0.00%	2.24%	0.00%	$1.23 \cdot 10^{-14}$
consider-using-enumerate	0.05%	0.00%	0.03%	0.00%	$7.06 \cdot 10^{-14}$
missing-class-docstring	0.43%	0.26%	0.73%	0.44%	$6.17 \cdot 10^{-13}$
trailing-whitespace	1.59%	0.05%	1.05%	0.00%	$1.43 \cdot 10^{-12}$
missing-function-docstring	2.56%	2.31%	3.26%	2.94%	$6.02 \cdot 10^{-12}$
unsubscriptable-object	0.04%	0.00%	0.02%	0.00%	$7.44 \cdot 10^{-12}$
inconsistent-return-statements	0.03%	0.00%	0.07%	0.00%	$1.35 \cdot 10^{-10}$
wrong-import-position	0.32%	0.00%	0.18%	0.00%	$2.54 \cdot 10^{-9}$
bad-whitespace	5.43%	0.53%	4.90%	0.15%	$2.68 \cdot 10^{-9}$
too-many-instance-attributes	0.11%	0.05%	0.08%	0.02%	$4.82 \cdot 10^{-9}$
unnecessary-comprehension	0.03%	0.00%	0.02%	0.00%	$7.74 \cdot 10^{-9}$
missing-final-newline	0.12%	0.00%	0.20%	0.00%	$1.11 \cdot 10^{-8}$
unused-variable	0.38%	0.23%	0.33%	0.14%	$8.46 \cdot 10^{-8}$
ungrouped-imports	0.05%	0.00%	0.04%	0.00%	$1.81 \cdot 10^{-7}$
too-few-public-methods	0.17%	0.08%	0.36%	0.12%	$4.04 \cdot 10^{-7}$
too-many-public-methods	0.00%	0.00%	0.01%	0.00%	$6.24 \cdot 10^{-7}$
missing-module-docstring	0.82%	0.65%	1.17%	0.74%	$7.03 \cdot 10^{-7}$
too-many-statements	0.07%	0.03%	0.06%	0.00%	$1.78 \cdot 10^{-6}$
attribute-defined-outside-init	0.42%	0.01%	0.24%	0.00%	$2.05 \cdot 10^{-6}$
line-too-long	2.54%	1.64%	2.08%	1.02%	$3.45 \cdot 10^{-6}$
bad-continuation	2.17%	0.51%	2.22%	0.34%	$5.88 \cdot 10^{-6}$
bad-option-value	0.004%	0.00%	0.002%	0.00%	$8.38 \cdot 10^{-6}$
consider-using-dict-comprehension	0.007%	0.00%	0.003%	0.00%	$1.32 \cdot 10^{-5}$
trailing-newlines	0.09%	0.00%	0.07%	0.00%	$4.00 \cdot 10^{-5}$
invalid-unary-operand-type	0.003%	0.00%	0.001%	0.00%	$8.94 \cdot 10^{-5}$
reimported	0.024%	0.00%	0.018%	0.00%	$1.37 \cdot 10^{-4}$

• 大きな違い

多くのローカル変数,
インポートエラー,
多くの引数,
無効な名前, など

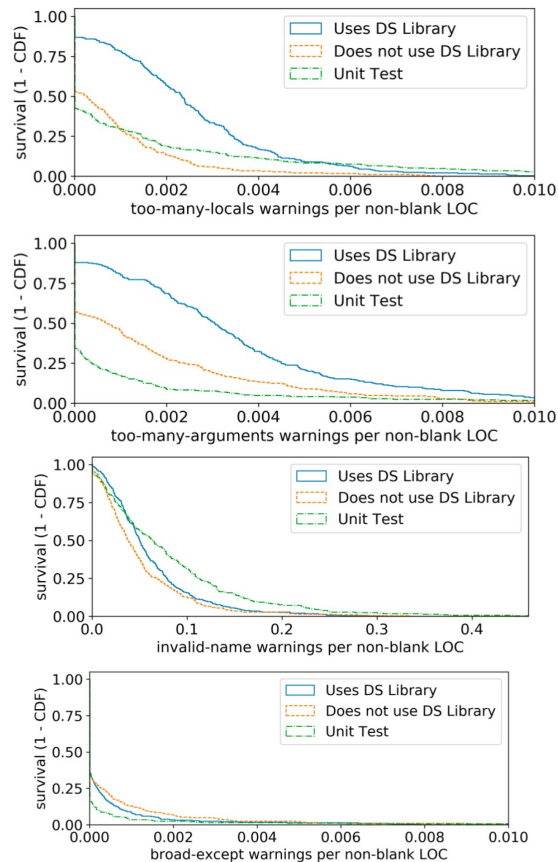
RQ2 : 従来のプロジェクトと比較して, データサイエンスプロジェクト間でコーディング標準への準拠はどのように異なるか?

- **ローカル変数の数が多すぎる関数/メソッド**が2倍以上含まれていた
- データサイエンスプロジェクトは, データサイエンス以外のプロジェクトよりも, **変数の命名基準に対する違反**が頻繁に含まれている

RQ3 : データサイエンスプロジェクト内のコーディング標準違反はどこで発生するか？

- 単体テストライブラリをインポートするモジュールに単体テストのラベルを付ける
- 残りのモジュールは , データサイエンスまたは機械学習ライブラリをインポートし, **DS**ライブラリを使用するか, **DS**ライブラリを使用しないとラベル付けされているかどうかを確認
- リポジトリごとにカテゴリごとに中間平均を計算
→多くのモジュールを備えた大規模なリポジトリの影響を防ぐ

RQ3 : データサイエンスプロジェクト内のコーディング標準違反はどこで発生するか？



- データサイエンスライブラリを直接インポートするモジュールにローカル変数が乱用されている
- 対照的に, 変数名の違反はプロジェクト全体で発生
- データサイエンスコードの大部分は, データサイエンスライブラリをインポートするモジュールで定義されている

Figure 7: Survival plots for selection of coding standard violations within projects, categorised based on module imports.

結論

- データサイエンスプロジェクトは, 他のプロジェクトと比較した場合にコーディング標準の違いを示した
- データサイエンスプロジェクトは, 過剰な数のパラメーターとローカル変数を使用する関数の割合が大幅に高い
- データサイエンスプロジェクトは, データサイエンス以外のプロジェクトとは異なる変数の命名規則に従う

所感

- 分析の手法や統計が難しく感じた
- 非データサイエンスでも,どの様なコーディング標準違反が多いのか,どこで違反が発生しているか知っておく必要がありそう