

2022/07/23 Tokyo.R #100 応用セッション

# RとPythonは仲良し

---

RStudioユーザーのためのPythonの始め方



やわらかクジラ



:@matsuchiy

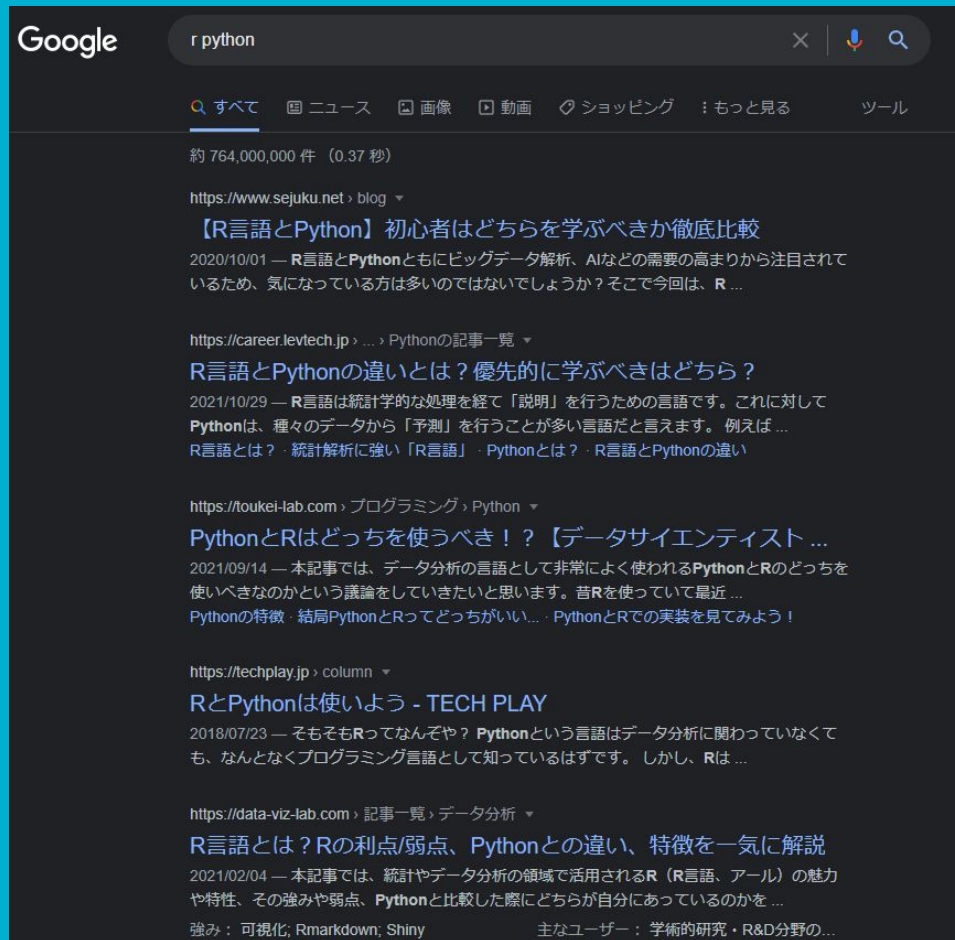
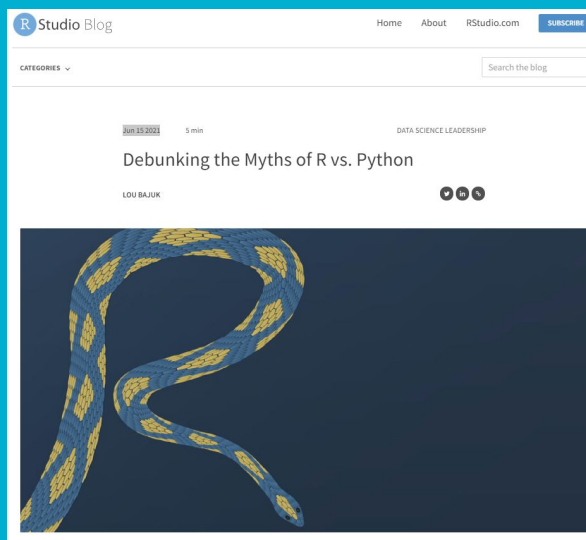
# 発表内容

- RとPythonは仲良し？
- Pythonを動かす
- データフレームの加工

RとPythonは仲良し？

# 🔥 Python vs R 🔥

- どっちを学ぶか論争
- 英語圏も似たような結果



Rユーザーの約 55% が  
Pythonも使用

2019 R Community Survey (n = 2,006)

[Why RStudio Supports Python for Data Science](#) (R Studio blog; Oct 30 2020)

# 高校生からPythonとRに触れる時代

統計学習の指導のために **先生向け** | [統計局ホーム](#) | [統計局の紹介](#) | [ご意見・ご感想](#) |

[授業モデル](#) | [補助教材](#) | [学校における統計教育の位置づけ](#) | [リンク集](#) | [サイトの歩き方](#)

[統計学習の指導のために](#) > [補助教材](#) > [総合学習のための補助教材](#) > [「高等学校における「情報II」のためのデータサイエンス・データ解析入門」](#)

## 総合学習のための補助教材

<a href="#">「小学生のための統計ってなに」</a>	<a href="#">「中学校のための統計社会が、自然が、生活がみえる統計」</a>	<a href="#">「高校生のための統計学習教材」</a>	<a href="#">「高等学校における「情報II」のためのデータサイエンス・データ解析入門」</a>
---------------------------------	---	---------------------------------	--

[「高等学校における「情報II」のためのデータサイエンス・データ解析入門」](#)

目次

### 第 7 章

## プログラミングの基本

R と Python のプログラミングの基本構文を学習します。実際にコードを書いてプログラムを実行し、プログラミングにより行われていることを理解します。



この章のゴール

R と Python の基本構文を理解する。

高校でRとPythonやるので知人の大学教員が呼ばれた話

<https://twitter.com/itsukoh0702/status/1522783450742530048>

娘の高校の数学の宿題でRを使っているの  
で、Python使いの父が手伝ったらRにはまっ  
た話

[https://twitter.com/Koji\\_Oyama/status/1545786102229123072](https://twitter.com/Koji_Oyama/status/1545786102229123072)

# RとPythonの仲良し例

---

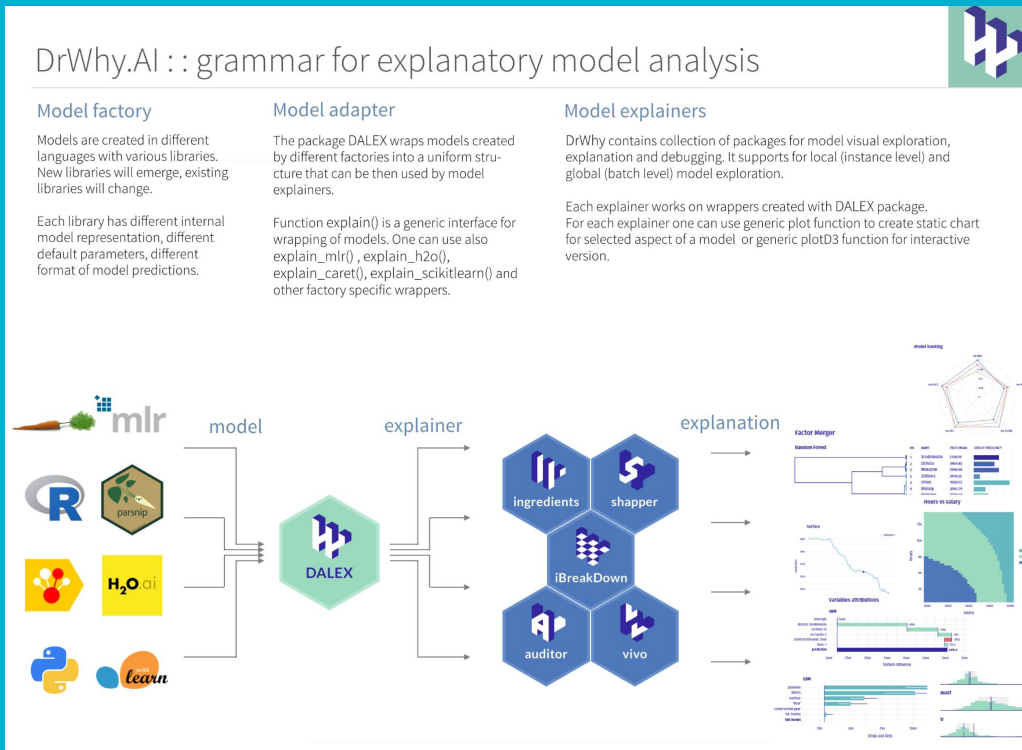
- RとPythonの両方を同時に習熟させるカリキュラムも
- 競争力を高めて労働市場でうまくやるための課題や工夫



データサイエンスの専門修士プログラム@ブリティッシュコロンビア大学

# RとPythonの仲良し例

- DrWhy.AI
  - 説明可能なAIのRパッケージ群






# RとPythonの仲良し例

- {theft}

- 時系列データの特徴量抽出のためのRパッケージ

theft 0.4.0 Get started Reference

Search for



## Introduction to theft

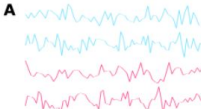
Trent Henderson  
2022-07-04  
Source: [vignettes/theft.Rmd](#)

```
library(theft)
```

### Purpose

theft facilitates user-friendly access to a structured analytical workflow for the extraction, analysis, and visualisation of time-series features. This structured workflow is presented in the graphic below (note that theft has many more functions than displayed in this graphic – keep reading for more):

#### 1. Load in raw time-series dataset

**A**

**B**

id	time	value	group
1	1	0.75	Control
2	1	1.24	Control
3	1	0.42	Treatment
...	...	...	...
...	...	...	...

#### 2. Extract features for each unique time series

**C**

```
init_theft
```

(? using Python feature sets)

**D**

```
calculate_features
```

catch22

feasts

tsfeatures

tsfresh

TSFEL

Kats

- [catch22](#) (R; see [Rcatch22](#) for the native implementation on CRAN)
- [feasts](#) (R)
- [tsfeatures](#) (R)
- [Kats](#) (Python)
- [tsfresh](#) (Python)
- [TSFEL](#) (Python)

# 本発表の意図

---

- 想定聴衆

- R(tidyverse)のデータ加工は慣れてるが Python 全然知らない
- Pythonでデータ加工するが, R(tidyverse)でどう書くか知りたい
- どちらも初心者なので両方学びたい

- 到達目標

- データフレームの加工を通じてお互いを知る
- 動かし方が大体分かれば, さらなる学びへのハードルが下がる
- みんな仲良く

# Pythonを動かす



# 自分のために作ったRとPython対応ページ

- 以下は主にこちらを元にお送りします

1 WindowsでのPythonインストール
2 パッケージの管理 (pip関連)
3 コマンドプロンプト or Terminalの操作
4 ブラウザからjupyter labを使う
5 仮想環境:venv
6 基本用語
7 Rとpythonで相互の読み込み
8 データフレーム操作の基本動詞
9 データフレームの連結や構造変換
10 ファイルの読み込み
11 ファイルの書き出し
12 関数定義
13 ※テンプレ
14 参考情報

## RとPythonは仲良し

やわらかクジラ

2022/07/18

### 1 WindowsでのPythonインストール

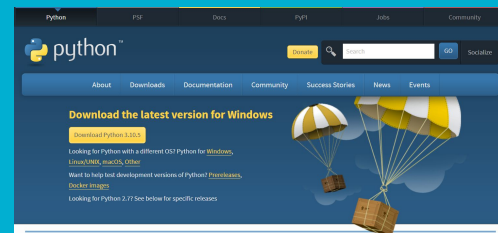
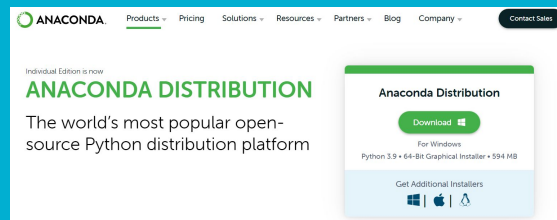
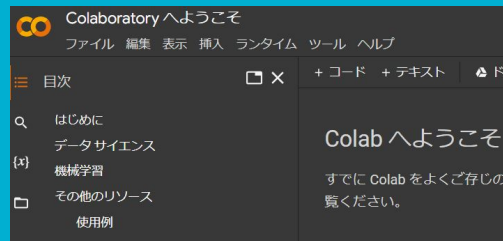
- Add Python 3.x to PATH にチェックを入れる
  - <https://www.python.jp/install/windows/install.html>

### 2 パッケージの管理 (pip関連)

- 以下, RStudioのTerminalから実行コマンドプロンプトから実行
  - ただし仮想環境あるとできないかい?
  - Windowsでは, コマンドプロンプトからも実行できる - 「ここに入力して検索」にcmdを入れて検索
- pipが適用されるPythonのバージョン確認
  - `python -V`
- 省略しても実行可能だが, `python -m` を最初につけることが推奨されている
- インストール済みパッケージの確認
  - `python -m pip list`
- パッケージインストール
  - `python -m pip install ここにパッケージ名`
- アップデートが必要なパッケージ一覧
  - `python -m pip list -o`
- パッケージのアップデート
  - `python -m pip install -U ここにパッケージ名`
- pip自体のアップデート
  - `python -m pip install --upgrade pip`

# Pythonを始める方法(用途:基本的なデータ分析)

- Google Colaboratory
  - Pythonインストールせずブラウザで使う
    - ただしRStudio連携はできない?
- Anaconda
  - 企業での利用は有料化(従業員200名以上)<sup>[1]</sup>
  - minicondaはOK
    - いずれにせよ環境によくない?
- 直にインストール【解決策】
  - Windowsの場合(Macも同じかい?)
    - 公式サイトからDLLしてインストール(手順例)
    - 「Add Python 3.x to PATH」にチェックを入れる
  - JupyterLab, Visual Studio Codeなどでも使える(RStudio外)

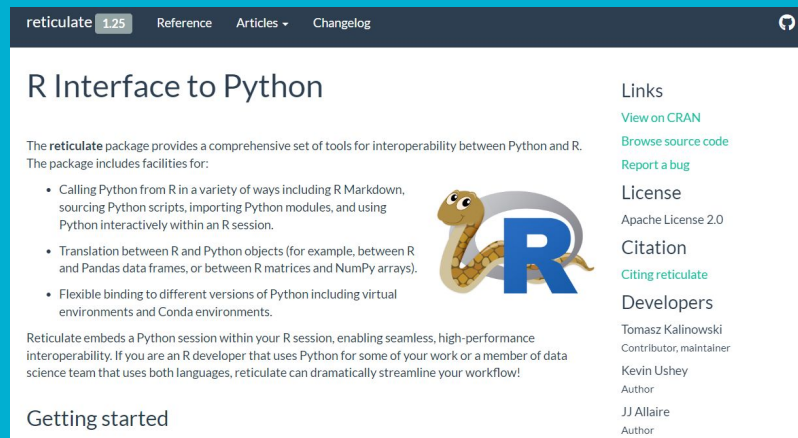


[1] <https://www.anaconda.com/blog/anaconda-commercial-edition-faq>

参考: <https://support.rstudio.com/hc/en-us/articles/1500007929061-Using-Python-with-the-RStudio-IDE>

# reticulateパッケージ

- Python実行する前に要インストール
- RとPythonの相互運用を可能にする
  - 由来は、東南アジアに生息する、世界最長のアミメニシキヘビ ( Python reticulatus)



# 実行環境

- OS
  - Windows 10
- R
  - 4.2.0
- RStudio
  - 2022.2.1.461
- Python
  - 3.10
- 仮想環境については勉強不足なのでパスします...

## 7.1 R

```
sinfo <-  
sessioninfo::session_info()
```

```
sinfo$platform
```

```
## setting value  
## version R version 4.2.0 (2022-04-22 ucrt)  
## os Windows 10 x64 (build 19043)  
## system x86_64, mingw32  
## ui RTerm  
## language (EN)  
## collate Japanese_Japan.utf8  
## ctype Japanese_Japan.utf8  
## tz Asia/Tokyo  
## date 2022-07-19  
## pandoc 2.17.1.1 @ C:/Program Files/RStudio/bin/quarto/bin/ (via rmarkdown)
```

## 7.2 RStudio version

```
rsinfo <-  
rstudioapi::versionInfo()
```

```
rsinfo$version
```

```
## [1] '2022.2.1.461'
```

## 7.3 Python

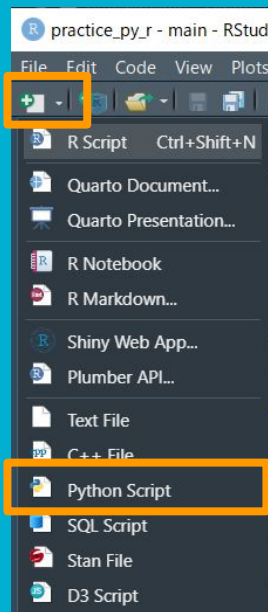
```
reticulate::py_version()
```

```
## [1] '3.10'
```

# RStudioでPython開く

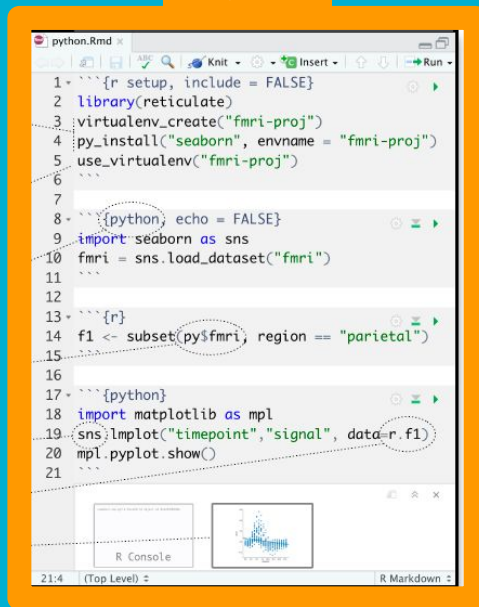
- 大事なことは{reticulate}の  
チートシートに
- 実行手段
  - a. Pythonスクリプト
  - b. Rスクリプト
  - c. Rmarkdown上なら、  
両方使える！

注意: 初回実行時にminicondaのインストール  
求められる場合もあるかもしれないが,nを選択

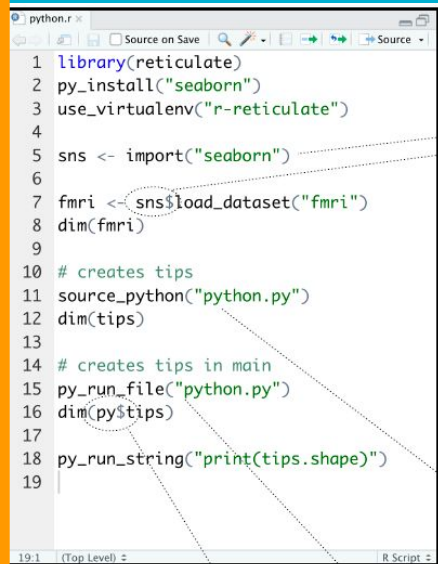


a

仲よし



c



b



# データフレームの交換！

- PythonでRのオブジェクトを読む
  - **r**.オブジェクト名
- RでPythonの変数を読む
  - **py\$**変数名

Rというオブジェクトは、  
Pythonでいう変数に該当する

## 7.1.1 Rでデータフレーム読み込み

```
library(reticulate)

df_r <- palmerpenguins::penguins
```

パッケージ名::関数など、の書き方で直接読みだせる

## 7.1.2 PythonでRのデータフレーム読み込みして列選択

```
r.df_r.head()
```

```
##   species   island bill_length_mm ... body_mass_g   sex year
## 0  Adelie  Torgersen         39.1   ...      3750   male 2007
## 1  Adelie  Torgersen         39.5   ...      3800 female 2007
## 2  Adelie  Torgersen         40.3   ...      3250 female 2007
## 3  Adelie  Torgersen          NaN   ... -2147483648   NaN 2007
## 4  Adelie  Torgersen         36.7   ...      3450 female 2007
##
## [5 rows x 8 columns]
```

```
df_py = r.df_r[['species', 'bill_length_mm']]
```

## 7.1.3 RからPythonで加工したデータフレーム読み込み

```
head(py$df_py)
```

```
##   species bill_length_mm
## 1  Adelie         39.1
## 2  Adelie         39.5
## 3  Adelie         40.3
## 4  Adelie          NA
## 5  Adelie         36.7
## 6  Adelie         39.3
```

# データフレームの交換！（Rマークダウン）

- チャンクの色分けは  
class.source = で指定<sup>[1,2]</sup>

## 21.2 チャンクの色

- <https://bookdown.org/yihui/rmarkdown-cookbook/chunk-styling.html>

```
# {r class.source="bg-info"}  
head(mtcars)
```

##		mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
##	Mazda RX4	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
##	Mazda RX4 Wag	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
##	Datsun 710	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
##	Hornet 4 Drive	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1
##	Hornet Sportabout	18.7	8	360	175	3.15	3.440	17.02	0	0	3	2
##	Valiant	18.1	6	225	105	2.78	3.460	20.22	1	0	3	1

```
# {r class.source="bg-primary"}  
head(mtcars)
```

```
# {r class.source="bg-success"}  
head(mtcars)
```

```
# {r class.source="bg-warning"}  
head(mtcars)
```

```
# {r class.source="bg-danger"}  
head(mtcars)
```

```
### Rでデータフレーム読み込み
```

```
```{r }
```

```
library(reticulate)
```

```
df_r <- palmerpenguins::penguins
```

```
```
```

```
### PythonでRのデータフレーム読み込みして列選択
```

```
```{python class.source="bg-success"}  
r.df_r.head()
```

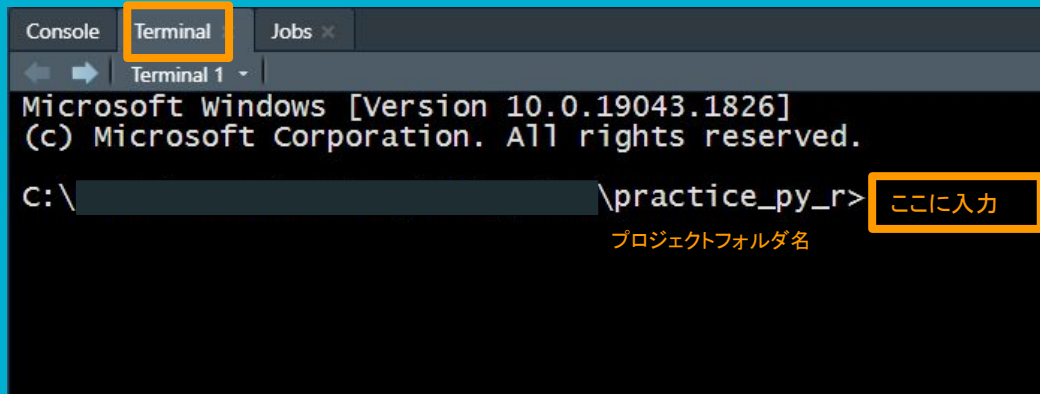
```
df_py = r.df_r[['species', 'bill_length_mm']]
```

```
```
```

# パッケージ管理はpip

---

- RStudioのTerminalでコマンド実行
  - TerminalはConsoleの横のタブにある
    - なければTools > Terminal > New Terminal
  - コマンドプロンプト(Win)を立ち上げてでもできる



# pipはこれだけ覚えれば何とかなる

- 最初にpython -mを付けることが推奨される<sup>[1]</sup>

- pipが適用されるPythonのバージョン確認
  - `python -V`
- インストール済みパッケージの確認
  - `python -m pip list`
- パッケージインストール
  - `python -m pip install` ここにパッケージ名
- アップデートが必要なパッケージ一覧
  - `python -m pip list -o`
- パッケージのアップデート
  - `python -m pip install -U` ここにパッケージ名
- pip自体のアップデート
  - `python -m pip install --upgrade pip`

| Package    | Version |
|------------|---------|
| -----      | -----   |
| pip        | 22.0.4  |
| setuptools | 58.1.0  |

pip list の初期出力例

[1] <https://www.python.jp/install/windows/pip.html>(python Japan); <https://pip.pypa.io/en/stable/> (公式ドキュメント)

# データフレームの加工



パッケージを入れておくため  
Terminalで実行

```
python -m pip install pandas
```

# 必読資料

- 大事なことは [pandas チートシート](#) に大体書いてある
  - Rstudio Data Wrangling Cheatsheet に影響を受けて書かれている
- 詳細は [pandas のドキュメント](#)

仲良し

**Data Wrangling with pandas Cheat Sheet**  
<http://pandas.pydata.org>  
[Pandas API Reference](#) [Pandas User Guide](#)

### Creating DataFrames

```
df = pd.DataFrame({
    "a": [4, 5, 6],
    "b": [7, 8, 9],
    "c": [10, 11, 12]},
    index = [1, 2, 3])
```

Specify values for each column.

```
df = pd.DataFrame([
    [4, 7, 10],
    [5, 8, 11],
    [6, 9, 12]],
    index=[1, 2, 3],
    columns=["a", "b", "c"])
```

Specify values for each row.

```
df = pd.DataFrame({
    "a": [4, 5, 6],
    "b": [7, 8, 9],
    "c": [10, 11, 12]},
    index = pd.MultiIndex.from_tuples(
        ("a", 1), ("a", 2), ("b", 1)),
    Create DataFrame with a MultiIndex
```

### Method Chaining

Most pandas methods return a DataFrame so that another pandas method can be applied to the result. This improves readability of code.

```
df = pd.DataFrame({
    "columns": ["a", "b", "c"],
    "values": ["a", "b", "c"]})
df["a"].value > 200
```

**Tidy Data** – A foundation for wrangling in pandas

In a tidy data set:  
Each variable is saved in its own column  
Each observation is saved in its own row

Tidy data complements pandas's **vectorized** operations. pandas will automatically preserve observations as you manipulate variables. No other format works as intuitively with pandas.

### Reshaping Data

Change layout, sorting, reindexing, renaming

**pd.melt(df)**  
Gather columns into rows.

**pd.pivot(columns="var", values="val")**  
Spread rows into columns.

**pd.concat([df1, df2])**  
Append rows of DataFrames

**pd.concat([df1, df2], axis=1)**  
Append columns of DataFrames

### Subset Observations - rows

**df[df.Length > 2]**  
Select rows that meet logical criteria.

**df.drop\_duplicates()**  
Remove duplicate rows (only considers columns).

**df.sample(frac=0.5)**  
Randomly select fraction of rows.

**df.sample(n=10)**  
Randomly select n rows.

**df.sort\_values("value")**  
Select and order top n entries.

**df.sort\_index("value")**  
Select top n rows.

**df.sort\_index()**  
Select top n rows.

### Subset Variables - columns

**df[["width", "length", "species"]]**  
Select multiple columns with specific names.

**df["width"] or df.width**  
Select single column with specific name.

**df.filter(regex="regex")**  
Select columns whose name matches regular expression regex.

### Using query

query() allows Boolean expressions for filtering rows.

**df.query("length > 7")**  
**df.query("length > 7 and width < 8")**  
**df.query("Name.str.startswith('abc')")**  
**df.query("Name.str.startswith('abc')", engine="python")**

### Logic in Python (and pandas)

| Logic        | Python | Boolean      |
|--------------|--------|--------------|
| Less than    | <      | Less than    |
| Greater than | >      | Greater than |
| Equal        | ==     | Equal        |
| Not equal    | !=     | Not equal    |
| And          | &      | And          |
| Or           |        | Or           |
| Not          | ~      | Not          |

### Logic in Regular Expressions (Examples)

| Logic              | Regular Expression | Example     |
|--------------------|--------------------|-------------|
| Length             | length             | length      |
| Starts with        | startswith         | startswith  |
| Ends with          | endswith           | endswith    |
| Contains           | contains           | contains    |
| Matches            | matches            | matches     |
| Does not match     | not_matches        | not_matches |
| Is in              | in                 | in          |
| Is not in          | not_in             | not_in      |
| Is a subset of     | is_subset          | is_subset   |
| Is not a subset of | not_subset         | not_subset  |

**pandas**

Getting started | User Guide | API reference | Development | Release notes

1.4.3

Search the docs...

## pandas documentation

Date: Jun 23, 2022 Version: 1.4.3

Download documentation: PDF Version | Zipped HTML

Previous versions: Documentation of previous pandas versions is available at [pandas.pydata.org](#).

Useful links: Binary Installers | Source Repository | Issues & Ideas | Q&A Support | Mailing List

pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

### Getting started

New to pandas? Check out the getting started guides. They contain an introduction to pandas' main concepts and links to additional tutorials.

### User guide

The user guide provides in-depth information on the key concepts of pandas with useful background information and explanation.

# データフレームから始める

- Pythonにも{palmerpenguins}がコピーされている
- RのCRANに対するPythonのPyPI(パイピーアイ)からpipでインストール



<https://pypi.org/project/palmerpenguins/>

## 8.2.1 R

```
pen_r <- palmerpenguins::penguins
```

## 8.2.2 Pythonでもペンギン

- ペンギンデータ for python
  - <https://pypi.org/project/palmerpenguins/>

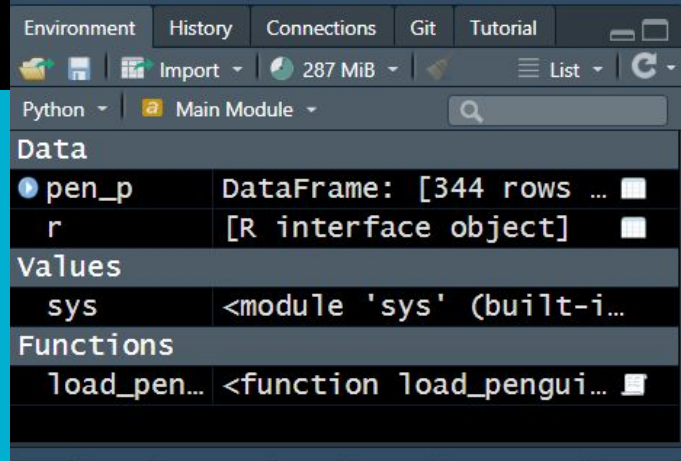
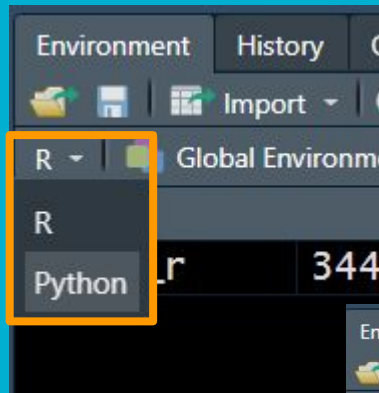
```
from palmerpenguins import load_penguins  
pen_p = load_penguins()
```

```
pen_p
```

```
##      species    island bill_length_mm ... body_mass_g    sex  year  
## 0      Adelie  Torgersen      39.1 ...      3750.0    male  2007  
## 1      Adelie  Torgersen      39.5 ...      3800.0  female  2007  
## 2      Adelie  Torgersen      40.3 ...      3250.0  female  2007  
## 3      Adelie  Torgersen      NaN ...        NaN     NaN  2007  
## 4      Adelie  Torgersen      36.7 ...      3450.0  female  2007  
## ...      ...      ...      ...      ...      ...      ...  
## 339 Chinstrap    Dream      55.8 ...      4000.0    male  2009  
## 340 Chinstrap    Dream      43.5 ...      3400.0  female  2009  
## 341 Chinstrap    Dream      49.6 ...      3775.0    male  2009  
## 342 Chinstrap    Dream      50.8 ...      4100.0    male  2009  
## 343 Chinstrap    Dream      50.2 ...      3775.0  female  2009  
##  
## [344 rows x 8 columns]
```

# 読み込んだ変数なども確認できる

- Environmentタブで, RとPythonの切り替えができる
- 読み込んだ変数や関数が確認可能





# %>% (パイプ) |> はどこ？

- Pythonでは"."(ドット, ピリオド)が  
Rのパイプと同様の働きをするみたい
  - method chainingと呼ばれる
- Rのように複数行で書くには, コード  
全体を()で囲む必要がある

## 8.9.1 R

- |> でつなげる

```
pen_r |>
  select(species, bill_length_mm) |>
  filter(species == "Gentoo")
```

```
## # A tibble: 124 × 2
##   species bill_length_mm
##   <fct>         <dbl>
## 1 Gentoo         46.1
## 2 Gentoo         50
## 3 Gentoo         48.7
## 4 Gentoo         50
## 5 Gentoo         47.6
## # ... with 119 more rows
```

## 8.9.2 Python

- . でつなげてコード全体を()で囲む

```
(pen_p[['species', 'bill_length_mm']]
  .query('species == "Gentoo"'))
```

```
##   species  bill_length_mm
## 152  Gentoo         46.1
## 153  Gentoo         50.0
## 154  Gentoo         48.7
## 155  Gentoo         50.0
## 156  Gentoo         47.6
## .. ...
## 271  Gentoo         NaN
## 272  Gentoo         46.8
## 273  Gentoo         50.4
## 274  Gentoo         45.2
## 275  Gentoo         49.9
```

# 行を選ぶ

- Pythonでは  
queryが便利

## 8.6.1.1 R

```
pen_r |>
  filter(bill_length_mm > 55)
```

```
## # A tibble: 5 × 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>         <dbl>         <dbl>          <int>      <int> <fct>
## 1 Gentoo Biscoe          59.6           17            230       6050 male
## 2 Gentoo Biscoe          55.9           17            228       5600 male
## 3 Gentoo Biscoe          55.1           16            230       5850 male
## 4 Chinstrap Dream         58            17.8          181       3700 fema...
## 5 Chinstrap Dream         55.8           19.8          207       4000 male
## # ... with 1 more variable: year <int>
```

```
pen_r |>
  filter(bill_length_mm > 55 & bill_depth_mm == 17)
```

```
## # A tibble: 2 × 8
##   species island bill_length_mm bill_depth_mm flipper_length_mm body_mass_g sex
##   <fct>   <fct>         <dbl>         <dbl>          <int>      <int> <fct>
## 1 Gentoo Biscoe          59.6           17            230       6050 male
## 2 Gentoo Biscoe          55.9           17            228       5600 male
## # ... with 1 more variable: year <int>
```

## 8.6.1.2 Python

```
pen_p[pen_p['bill_length_mm'] > 55]
```

```
##      species island bill_length_mm ... body_mass_g sex year
## 185   Gentoo Biscoe          59.6 ...      6050.0 male 2007
## 253   Gentoo Biscoe          55.9 ...      5600.0 male 2009
## 267   Gentoo Biscoe          55.1 ...      5850.0 male 2009
## 293 Chinstrap Dream          58.0 ...      3700.0 female 2007
## 339 Chinstrap Dream          55.8 ...      4000.0 male 2009
##
## [5 rows x 8 columns]
```

```
pen_p[pen_p.bill_length_mm > 55]
```

```
##      species island bill_length_mm ... body_mass_g sex year
## 185   Gentoo Biscoe          59.6 ...      6050.0 male 2007
## 253   Gentoo Biscoe          55.9 ...      5600.0 male 2009
## 267   Gentoo Biscoe          55.1 ...      5850.0 male 2009
## 293 Chinstrap Dream          58.0 ...      3700.0 female 2007
## 339 Chinstrap Dream          55.8 ...      4000.0 male 2009
##
## [5 rows x 8 columns]
```

```
pen_p.query('bill_length_mm > 55 & bill_depth_mm == 17')
```

```
##      species island bill_length_mm ... body_mass_g sex year
## 185   Gentoo Biscoe          59.6 ...      6050.0 male 2007
## 253   Gentoo Biscoe          55.9 ...      5600.0 male 2009
##
## [2 rows x 8 columns]
```

# 列を選ぶ

- 複数列

- 複数列

```
pen_r |>
  select(species, bill_length_mm)
```

```
## # A tibble: 344 × 2
##   species bill_length_mm
##   <fct>         <dbl>
## 1 Adelie         39.1
## 2 Adelie         39.5
## 3 Adelie         40.3
## 4 Adelie         NA
## 5 Adelie         36.7
## # ... with 339 more rows
```

```
pen_r |>
  select(species:bill_length_mm)
```

```
## # A tibble: 344 × 3
##   species island   bill_length_mm
##   <fct>   <fct>         <dbl>
## 1 Adelie Torgersen     39.1
## 2 Adelie Torgersen     39.5
## 3 Adelie Torgersen     40.3
## 4 Adelie Torgersen     NA
## 5 Adelie Torgersen     36.7
## # ... with 339 more rows
```

```
pen_p[['species', 'bill_length_mm']]
```

```
##   species bill_length_mm
## 0      Adelie         39.1
## 1      Adelie         39.5
## 2      Adelie         40.3
## 3      Adelie         NaN
## 4      Adelie         36.7
## ..      ...           ...
## 339 Chinstrap         55.8
## 340 Chinstrap         43.5
## 341 Chinstrap         49.6
## 342 Chinstrap         50.8
## 343 Chinstrap         50.2
##
## [344 rows x 2 columns]
```

```
pen_p.loc[:, 'species':'bill_length_mm']
```

```
##   species island   bill_length_mm
## 0      Adelie Torgersen     39.1
## 1      Adelie Torgersen     39.5
## 2      Adelie Torgersen     40.3
## 3      Adelie Torgersen     NaN
## 4      Adelie Torgersen     36.7
## ..      ...           ...
## 339 Chinstrap Dream         55.8
## 340 Chinstrap Dream         43.5
## 341 Chinstrap Dream         49.6
## 342 Chinstrap Dream         50.8
## 343 Chinstrap Dream         50.2
##
## [344 rows x 3 columns]
```

# 列を選ぶ

- 正規表現で列名の文字指定
- pandasでは, filterが列と行両方に使える
  - axis引数で指定

## 8.3.2.1 R

```
pen_r |>  
  select(matches("^bill")) # starts_with("bill")
```

```
## # A tibble: 344 × 2  
##   bill_length_mm bill_depth_mm  
##           <dbl>         <dbl>  
## 1           39.1           18.7  
## 2           39.5           17.4  
## 3           40.3            18  
## 4            NA            NA  
## 5           36.7           19.3  
## # ... with 339 more rows
```

## 8.3.2.2 Python

```
pen_p.filter(regex = '^bill')
```

```
##           bill_length_mm  bill_depth_mm  
## 0              39.1             18.7  
## 1              39.5             17.4  
## 2              40.3             18.0  
## 3              NaN             NaN  
## 4              36.7             19.3  
## ..              ...              ...  
## 339             55.8             19.8  
## 340             43.5             18.1  
## 341             49.6             18.2  
## 342             50.8             19.0  
## 343             50.2             18.7  
##  
## [344 rows x 2 columns]
```

# 列名変更

- どちらもrename
- newとoldの配置が逆

## 8.4.1 R

- new = old

```
pen_r |>
  rename(species2 = species,
         island2 = island)
```

```
## # A tibble: 344 × 8
##   species2 island2 bill_length_mm bill_depth_mm flipper_length_mm body_mass_g
##   <fct>    <fct>         <dbl>         <dbl>          <int>        <int>
## 1 Adelie   Torgersen         39.1          18.7           181         3750
## 2 Adelie   Torgersen         39.5          17.4           186         3800
## 3 Adelie   Torgersen         40.3          18            195         3250
## 4 Adelie   Torgersen          NA           NA             NA           NA
## 5 Adelie   Torgersen         36.7          19.3           193         3450
## # ... with 339 more rows, and 2 more variables: sex <fct>, year <int>
```

## 8.4.2 Python

- 'old':'new'

```
pen_p.rename(columns = {'species':'species2',
                        'island':'island2'})
```

```
##   species2   island2 bill_length_mm ... body_mass_g sex year
## 0    Adelie Torgersen         39.1 ...    3750.0 male 2007
## 1    Adelie Torgersen         39.5 ...    3800.0 female 2007
## 2    Adelie Torgersen         40.3 ...    3250.0 female 2007
## 3    Adelie Torgersen          NaN ...         NaN   NaN 2007
## 4    Adelie Torgersen         36.7 ...    3450.0 female 2007
## ..      ...      ...      ...      ...      ...      ...
## 339 Chinstrap Dream          55.8 ...    4000.0 male 2009
## 340 Chinstrap Dream          43.5 ...    3400.0 female 2009
## 341 Chinstrap Dream          49.6 ...    3775.0 male 2009
## 342 Chinstrap Dream          50.8 ...    4100.0 male 2009
## 343 Chinstrap Dream          50.2 ...    3775.0 female 2009
##
## [344 rows x 8 columns]
```

# 列を作成

- Pythonではassignを使い、列の中身を持ってくる感じで指定する
- Rのcase\_whenのように条件指定して作成するような場合は大変そう?

## 8.7.1.1 R

```
pen_r |>
  mutate(species2 = species,
         island2 = island)
```

```
## # A tibble: 344 × 10
##   species island bill_length_mm bill_depth_mm flipper_length... body_mass_g sex
##   <fct>   <fct>         <dbl>         <dbl>         <int>      <int> <fct>
## 1 Adelie  Torge...         39.1          18.7          181       3750 male
## 2 Adelie  Torge...         39.5          17.4          186       3800 fema...
## 3 Adelie  Torge...         40.3           18          195       3250 fema...
## 4 Adelie  Torge...          NA           NA           NA         NA <NA>
## 5 Adelie  Torge...         36.7          19.3          193       3450 fema...
## # ... with 339 more rows, and 3 more variables: year <int>, species2 <fct>,
## #   island2 <fct>
```

## 8.7.1.2 Python

```
pen_p.assign(species2 = pen_p['species'],
             island2 = pen_p['island'])
```

```
##      species      island  bill_length_mm  ...  year  species2  island2
## 0      Adelie  Torgersen         39.1  ...  2007      Adelie  Torgersen
## 1      Adelie  Torgersen         39.5  ...  2007      Adelie  Torgersen
## 2      Adelie  Torgersen         40.3  ...  2007      Adelie  Torgersen
## 3      Adelie  Torgersen          NaN  ...  2007      Adelie  Torgersen
## 4      Adelie  Torgersen         36.7  ...  2007      Adelie  Torgersen
## ..      ...      ...      ...      ...  ...      ...      ...
## 339 Chinstrap  Dream         55.8  ...  2009  Chinstrap  Dream
## 340 Chinstrap  Dream         43.5  ...  2009  Chinstrap  Dream
## 341 Chinstrap  Dream         49.6  ...  2009  Chinstrap  Dream
## 342 Chinstrap  Dream         50.8  ...  2009  Chinstrap  Dream
## 343 Chinstrap  Dream         50.2  ...  2009  Chinstrap  Dream
##
## [344 rows x 10 columns]
```

# 要約値を作成

- 単一列
- Pythonの方が圧倒的にシンプルに書ける

## 8.8.1.1 R

```
pen_r |>
  summarise(blm_mean = mean(bill_length_mm,
                           na.rm = TRUE),
            blm_sd   = sd(bill_length_mm,
                           na.rm = TRUE),
            blm_n    = sum(!is.na(bill_length_mm)))
```

```
## # A tibble: 1 × 3
##   blm_mean blm_sd blm_n
##   <dbl>   <dbl> <int>
## 1     43.9     5.46   342
```

```
pen_p.bill_length_mm.agg(['mean', 'std', 'count'])
```

```
## mean      43.921930
## std       5.459584
## count     342.000000
## Name: bill_length_mm, dtype: float64
```

# 結果をデータフレームで返すには

```
pen_p.agg({'bill_length_mm' : ['mean', 'std', 'count']})
```

```
##           bill_length_mm
## mean      43.921930
## std       5.459584
## count     342.000000
```

# RとPythonの仲良し例

- {vetiver}<sup>[1,2]</sup>
  - MLOps<sup>[3]</sup>のフレームワーク
  - Python版もある



[1] ドキュメント: <https://vetiver.rstudio.com/>

[2] tidyverseブログ: <https://www.tidyverse.org/blog/2022/06/announce-vetiver/>

[3] 機械学習またはディープラーニングのライフサイクルを管理するための実践手法



## Train a model

For this example, let's work with data on fuel efficiency for cars to predict miles per gallon.

R PYTHON

Let's consider one kind of model supported by vetiver, a [tidymodels](https://www.tidymodels.org/) workflow that encompasses both feature engineering and model estimation.

```
library(tidymodels)

car_mod <-
  workflow(mpg ~ ., linear_reg()) %>%
  fit(mtcars)
```

## Train a model

For this example, let's work with data on fuel efficiency for cars to predict miles per gallon.

R PYTHON

Let's consider one kind of model supported by vetiver, a [scikit-learn](https://scikit-learn.org/) linear model.

```
from vetiver.data import mtcars
from sklearn import linear_model

car_mod = linear_model.LinearRegression().fit(mtcars, mtcars["mpg"])
```

```
68 ::: panel-tabset
69 ## R
70
71 Let's consider one kind of model supported by vetiver, a [tidymodels](https://www.tidymodels.org/) workflow that
72
73 ```{r}
74 #| message: false
75 library(tidymodels)
76
77 car_mod <-
78   workflow(mpg ~ ., linear_reg()) %>%
79   fit(mtcars)
80 ```
81
82 ## Python
83
84 Let's consider one kind of model supported by vetiver, a [scikit-learn](https://scikit-learn.org/) linear model.
85
86 ```{python}
87 from vetiver.data import mtcars
88 from sklearn import linear_model
89
90 car_mod = linear_model.LinearRegression().fit(mtcars, mtcars["mpg"])
91 ```
92 :::
```

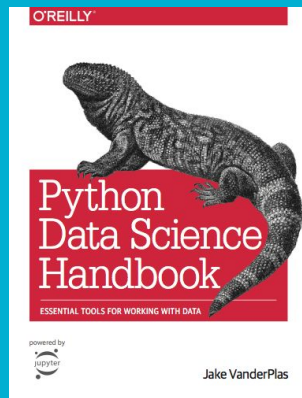
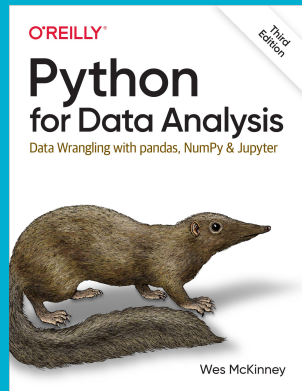


で書けばRとPythonがもっと仲良しに！

# Pythonをさらに学ぶ

---

- 無料でwebで読める
  - [Python for Data Analysis, 3E](#)
  - [Python Data Science Handbook](#)



# まとめ

---

- RとPythonは仲良し
- Pythonを直インストール & RStudioのRマークダウン使用でenjoy!
- 慣れてるデータフレーム加工でお互いを知る

# Enjoy!

