Friend or Foe: Comparison of R & Python in Data Wrangling & Visualisation

11th October 2019, PyCon DE & PyData Berlin 2019
Yuta Kanzawa @yutakanzawa

SFE Senior Analyst at Janssen Pharmaceutical K.K., Tokyo
A Family Company of Johnson & Johnson

Agenda

1. Comparison of R and Python as Language
2. Data Wrangling
3. Data Visualisation
4. Other Cases (Brief Overview)
   - ML, Big Data, NLP
5. Conclusion
   - R versus Python?
   - To be bilingual

Ich bin…

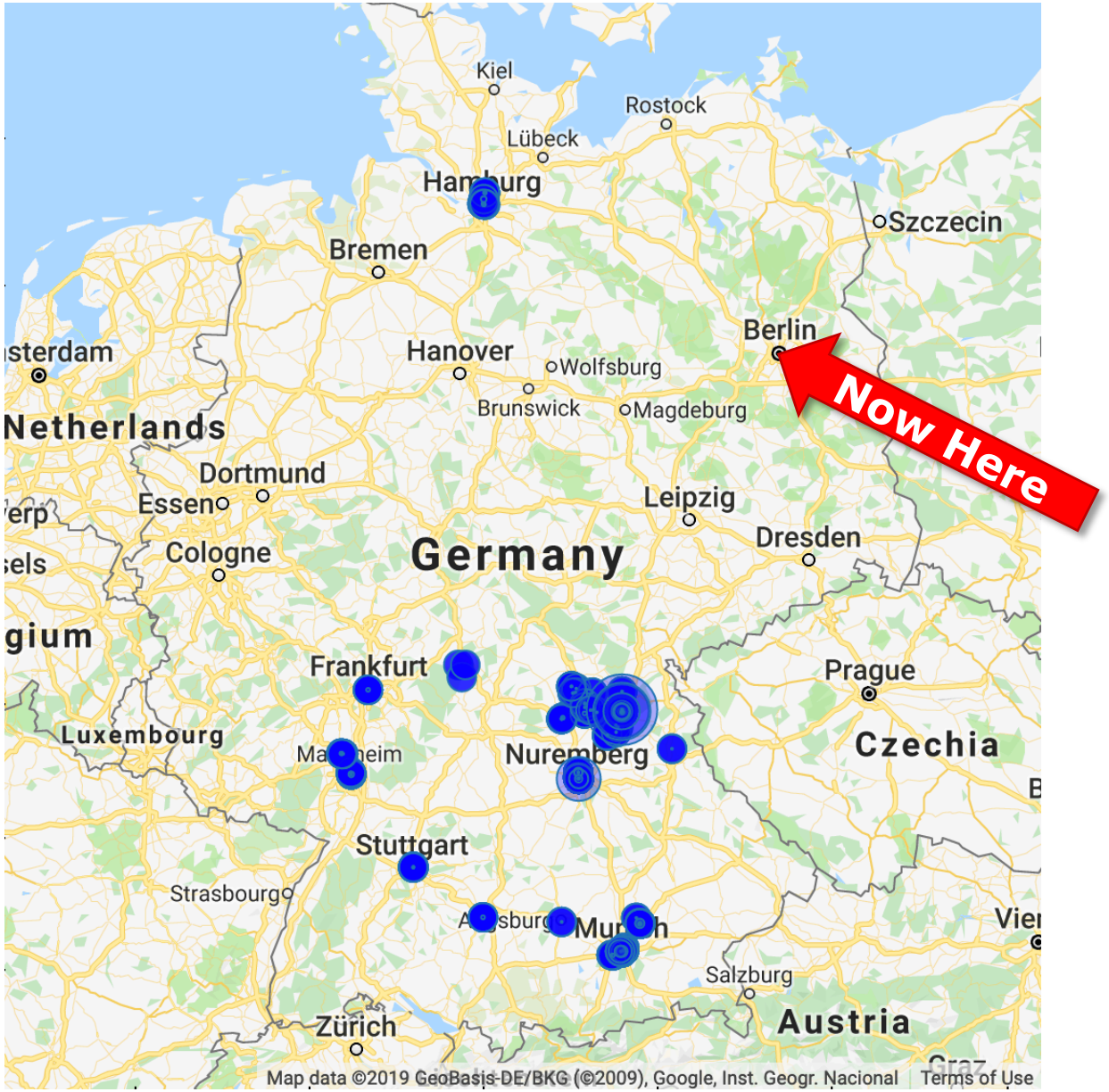
- Yuta Kanzawa (twitter: @yutakanzawa)
- Data scientist an Janssen Japan, Tokyo
  - Eine pharmazeutische Firma von Johnson & Johnson
  - Vertriebseffektivität, Marketing
- Oper- & weinliebhaber
  - Wagner
  - Burgunder
- 7 Sprachen
  - Menschen: Japanish, Englisch, Deutsch (Grundlagen)
  - Computer: R, Python, SAS, SQL

I am…

- Yuta Kanzawa (twitter: @yutakanzawa)
- Data scientist at Janssen Japan, Tokyo
  - A pharmaceutical company of Johnson & Johnson
  - Sales force effectiveness, marketing
- Opera & wine lover
  - Wagner
  - Bourgogne
- 7 languages
  - Human: Japanese, English, German (basic)
  - Computer: R, Python, SAS, SQL

Germany & I

4sq check-ins: from July 2012 to Sept 2019



Jupyter NB for this plot: To be updated

Quick Survey

- R? Python? Both?

```
if (you use now or have ever used R){
  Raise your hand.
}
```

```
if you use now or have ever used Python:
    Raise your hand.  # Not an error
```

If you use now or have ever used both, raise your hand (and jump!)

Comparison of R and Python as Language

- Differences
- Similarities

Differences

	R	Python
Purpose	Specific: Statistical analysis	General: Web app, system dev, data science
Paradigm	Procedural	Object-oriented
IDE/Editor/Dev Tool	RStudio	Jupyter Notebook, PyCharm, VS Code
Dots	Allowed in names	Dot notation
Indexing	1-based	0-based

Operators, functions to be applied to each element of a vector in R.

If you want to get each value of vector x squared as y, where x = (1, 2, 3):

R

```
> x <- c(1, 2, 3)
> y <- x ** 2
> y
[1] 1 4 9
```

Python*

```
>>> x = [1, 2, 3]
>>> y = x**2
TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int’
>>> y = [e**2 for e in x]
>>> y
[1, 4, 9]
```

* Some of numpy functions support this kind of operation.

Similarities

	R	Python
First appeared in	1993*1	1990*2
Major conference	useR! (since 2004)	PyCon (since 2003*3)
Current stable ver.*4	3.6.1 (July 2019) "Action of the Toes"*5	3.7.4 (July 2019)
Typing	Dynamic	Dynamic, optionally static
Iris dataset	Built-in	`from sklearn.datasets import load_iris` `iris_org = load_iris()`

*1 https://en.wikipedia.org/wiki/R_(programming_language)
*2 https://en.wikipedia.org/wiki/Python_(programming_language)
*3 First EuroPython was held in 2002.
*4 As of 09:00 CEST, 11th October 2019
*5 Each release version of R has a nickname.

Data Wrangling

- tidyverse
- pandas

tidyverse and pandas

	tidyverse	pandas
Description	Collection of R packages designed for data science	Library providing data structures and data analysis tools
Status	Modernising base R for ‘tidy data’	De facto standard
Flow	Pipe operator %>%	Method chaining

What‘s ‘tidy’ data?

- Codd’s 3rd normal form*1,2
  1. Each variable forms a column.
  2. Each observation forms a row.
  3. Each type of observational unit forms a table.

Not tidy

product	week01	week02
A	NA	2
B	16	11
C	3	1

Tidy

product	week	amount
A	1	NA
A	2	16
B	1	3
B	2	2
C	1	11
C	2	1

*1 E.F. Codd (1990)
*2 H. Wickham (2013)

Example case: iris dataset

Original

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa

Tidy

id	Species	part	dim	value
1	setosa	Sepal	Length	5.1
1	setosa	Sepal	Width	3.5
1	setosa	Sepal	Length	1.4
1	setosa	Sepal	Width	0.2
2	setosa	Sepal	Length	4.9
2	setosa	Sepal	Width	3.0

Example codes

tidyverse*1

```
library(tidyverse)
tidy_iris <- iris %>%
  rowid_to_column("id") %>%
  pivot_longer(Sepal.Length:Petal.Width) %>%
  separate(name, into = c("part", "dim"))
```

	id	Species	part	dim	value
1	1	setosa	Sepal	Length	5.1
2	1	setosa	Sepal	Width	3.5
3	1	setosa	Petal	Length	1.4
4	1	setosa	Petal	Width	0.2
5	2	setosa	Sepal	Length	4.9
6	2	setosa	Sepal	Width	3.0

pandas*2

```
import pandas as pd
tidy_iris = iris\
    .reset_index().rename(columns={'index': 'id'})\
    .melt(id_vars=['species'])
tidy_iris[['part', 'dim']] = tidy_iris.variable\
    .apply(lambda v: pd.Series(str(v).split()))
tidy_iris.drop(columns='variable', inplace=True)
```

id	species	value	part	dim
0	setosa	5.1	sepal	length
1	setosa	4.9	sepal	length
2	setosa	4.7	sepal	length
3	setosa	4.6	sepal	length
4	setosa	5.0	sepal	length
5	setosa	5.4	sepal	length

n.b. Orders of columns and rows could be different in the results.
*1 Update tidyr to 1.0.0 or higher.
*2 Jupyter NB of this code snippet: To be updated

In my experience:

- **R** offers quick and simple ways to talk with data.
  - e.g. **Exploratory data analysis** at your hand

- **But** not so suitable for **data pipeline**
  - Unless you use some tools/libraries
  - e.g. When your data flows into database

Data Visualisation

- ggplot2
- matplotlib
- plotly

ggplot2 and matplotlib

	ggplot2	matplotlib
Description	A system for declaratively creating graphics	A Python 2D plotting library which produces publication quality figures
Status	Modernising base R plot function	De facto standard
Key feature	Aesthetics	Axes
Interactive	Not implemented	Implemented
ggplot style	Built-in	`import matplotlib.pyplot as plt` `plt.style.use('ggplot')`

ggplot2 Usage*

- (0) Start with `ggplot()`
  - supply a dataset and aesthetic mapping with `aes()`
- Add on (1) Layers
  - e.g. Scatter plot: `geom_point()` , Histogram: `geom_histogram()`
- And (2) Scales
  - e.g. Specify colour sets.
    Reverse x axis.
- (3) Faceting specifications
  - e.g. Lay out panels in a grid.
- (4) Coordinate systems
  - e.g. Flip coordinates.

e.g.

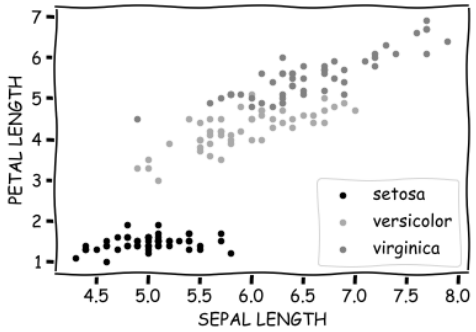
```
ggplot(data, aes(…)) +
  geom_point() +
  scale_colour_brewer(…) +
  scale_x_log10()
```

* https://ggplot2.tidyverse.org/

Example case: iris dataset

- Boss: Show me relationships between sepal length and petal length by species ASAP!

- Me: (Draw a scatter plot!)
  - x: Sepal length
  - y: Petal length
  - Colour each point based on its species.

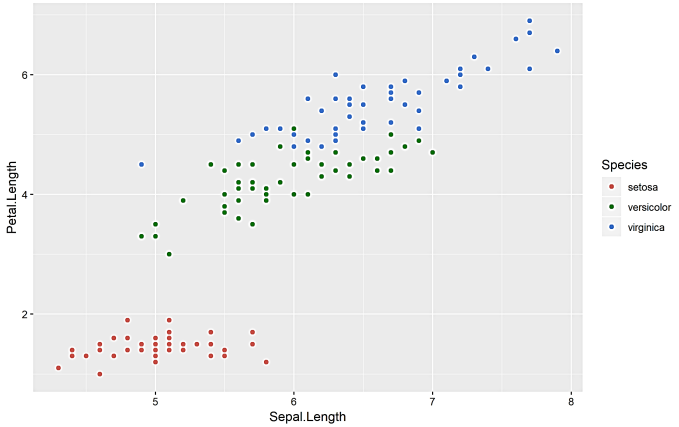


* https://xkcd.com/2207/
* Matplotlib's xkcd style: https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.xkcd.html

Example codes

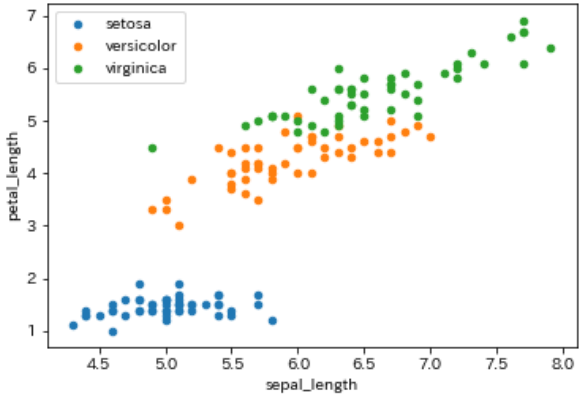
ggplot2

```
library(ggplot2)
ggplot(iris,
       aes(x = Sepal.Length,
           y = Petal.Length,
           colour = Species)) +
  geom_point()
```



matplotlib*

```
import matplotlib.pyplot as plt
fig, ax = plt.subplots()
cmap = plt.get_cmap('tab10')
for i, (key, df) in enumerate(iris.groupby('species')):
    df.plot.scatter(x='sepal_length', y='petal_length',
                    ax=ax, color=cmap(i), label=key)
ax.legend()
plt.show()
```



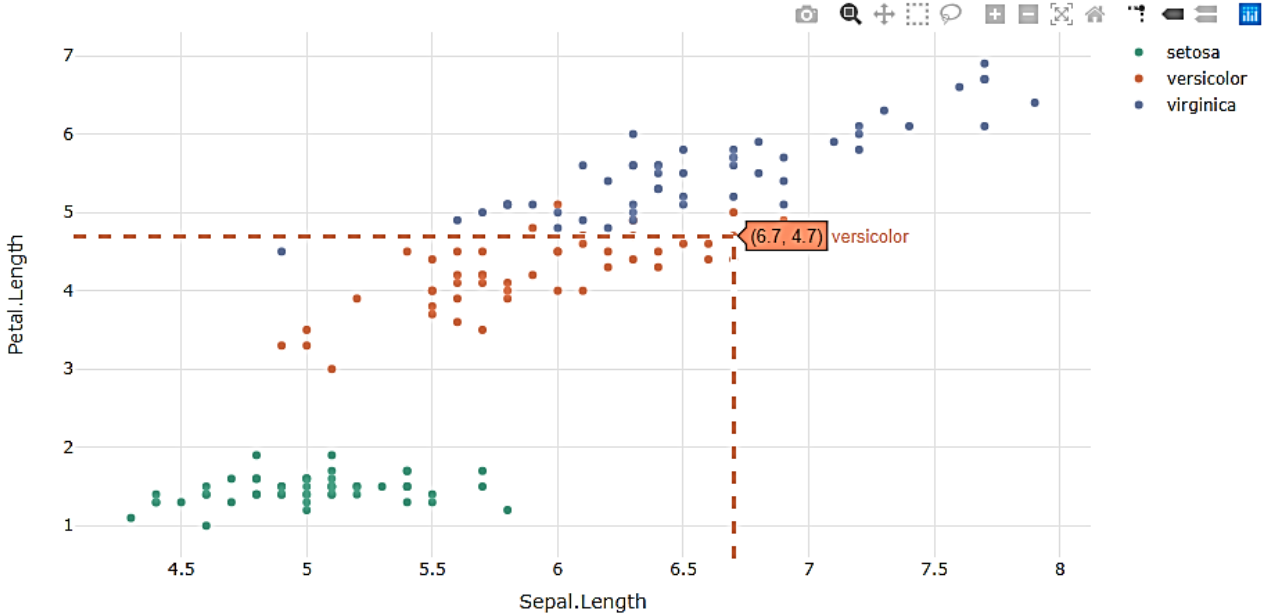
* Jupyter NB of this code snippet: To be updated

plotly: an interactive graph option for R

- Has APIs for R (besides Python!)
  - `plot_ly()`
  - `ggplotly()`: Plotlify ggplot objects.
- Example

```
library(plotly)

p <- plot_ly(data = iris,
             x = ~Sepal.Length,
             y = ~Petal.Length,
             color = ~Species,
             type = "scatter",
             mode = "markers")
p
```



In my experience:

- **R** provides strong and efficient ways to draw graphs.
  - e.g. **Exploratory data analysis**
  - e.g. **Presentation slides**, charts in reports

- **But** not necessarily the best option for **dashboards**
  - e.g. When you build dashboards shared with marketing team

Other Cases (Brief Overview)

- Machine learning
- Big data (distributed data processing)
- Natural language processing

Task x library

	R	Python
ML	baseR, E1071, xgboost, caret, mlr, h2o, keras*1, tensorflow*2	scikit-learn, statsmodels, Keras, TensorFlow
Big Data	sparklyr*3, SparkR*3	Dask, pyspark*3, pydoop*4, mrjob*4
NLP	tm, tidytext, spacyr*5, wordcloud2	NLTK, StanfordNLP, spaCy, wordcloud

*1 Interface to keras
*2 Interface to TensorFlow
*3 Interfaces to Apache Spark
*4 Interfaces to Apache Hadoop
*5 Interface to spaCy

Conclusion

- R versus Python?
- To be bilingual

R versus Python?

I think that is not helpful because it is not actually a battle. These things exist independently and are both awesome in different ways. […] R is a weird language but it is weird for good reasons, and it's a really good fit for data science. […] There are multiple ways of attacking the same problem, and sometimes the reason R is different is good. […] ***Use whatever makes you happy.***

– Hadley Wickham*

* Creator and developer of tidyverse package in R. https://qz.com/1661487/hadley-wickham-on-the-future-of-r-python-and-the-tidyverse/

To be bilingual

- Enhances your data analysis skills
  - Exploratory data analysis
  - Publication-grade graphs

- Could widen your career path in data science field
  - Public exposure
  - Community

- And… **learning new things is just fun!** (isn’t it?)

Enjoy!
Viel Spaß!

Appendix

- Reference
- R in Jupyter Notebook
- Hybrid

Reference

- C. Roach. "R for Pythonistas", presented at PyData NYC 2017, NYC, USA, 2017.
- E.F. Codd. *The Relational Model for Database Management: Version 2*. Boston: Addison-Wesley Longman Publishing, 1990.
- H. Wickham. "Tidy Data". *Journal of Statistical Software*, vol. 59, 20th February 2013.
- D. Kopf. "What's next for the popular programming language R?" Internet: https://qz.com/1661487/hadley-wickham-on-the-future-of-r-python-and-the-tidyverse/, 17th August, 2019 [30th September, 2019].
- T. Kluyver and Philipp A. "IRkernel". Internet: https://irkernel.github.io/, [30th September, 2019].
- T. Motohashi. *Maeshori Taizen* (Comprehensive Data Preprocessing). Tokyo: Gijutsu-Hyohron, 2018.

You can run R in Jupyter Notebook.

- Just by installing **IRkernel** (R kernel for Jupyter).*

* https://irkernel.github.io/

You can hybrid them (if necessary).

rpy2 runs R code in Python.

reticulate runs Python code in R.