I am Bilingual - Python and R

Priyanga Dilini Talagala Thiyanga S. Talagala

2021-10-25

Contents

Preface

WIP!!

This book is still in progress in various draft forms.

Chapter 1

Introduction to R and Python

R Vs Python: What's the Difference? https://www.guru99.com/r-vs-python.html

1.1 About R and Python

1.1.1 R

R is an object oriented, open source programming **language** and **environment** for statistical computing and graphics. R is not a statistics system but an environment within which statistical techniques are implemented. Further, R gains more capabilities via packages, its fundamental shareable units that bundle together R functions, code, data, documentation, and tests etc. (R Core Team, 2020).

1.1.2 Python

Python is an object-oriented, interpreted, and interactive programming language. The motto of Python language is "Batteries included" as the functionality of the language can be performed via its comprehensive standard in built Libraries (Wikipedia contributors, 2020a).

1.2 History of R and Python

1.2.1 R

R is an implementation of the S programming language which was created by John Chambers in 1976. In 1991, an alternative implementation of the basic S language was developed by Ross Ihaka and Robert Gentleman, University of Auckland, New Zealand. It was published in 1993 (Wikipedia contributors, 2020b).

1.2.2 Python

In 1989, Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands started the implementation of Python as a successor to ABC programming language. Python 2.0 was released in 2000. Python 3.0, a major revision of the language that is not completely backward-compatible was released in 2008 (Wikipedia contributors, 2020a) . Today many developers create libraries strictly for the use with Python 3.

1.3 Story behind their names

1.3.1 R

R was introduced by **R**oss Ihaka and **R**obert Gentleman and it was named after the first names of the two authors. The name of the "S" language also had some influence on the selection of its name and it was selected partly as a play on the name of S (Wikipedia contributors, 2020b).

1.3.2 Python

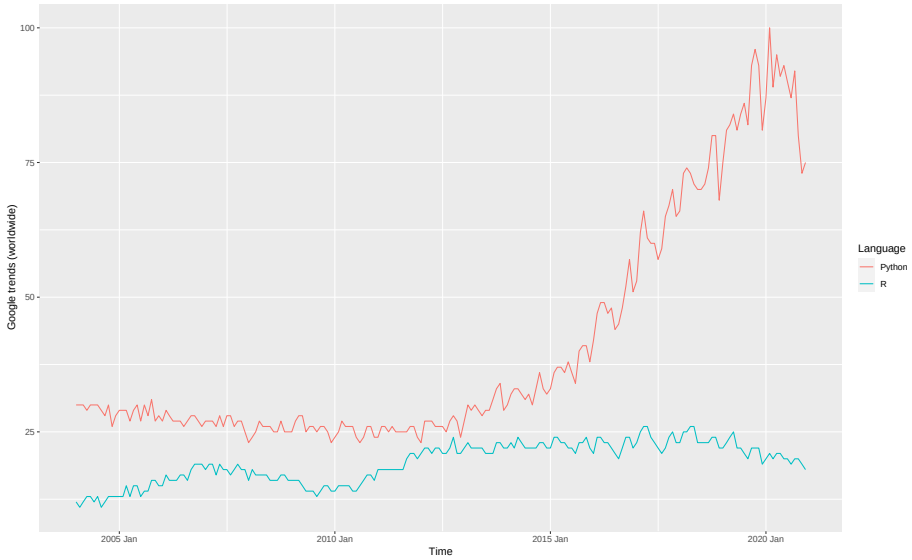
Python was named after a famous TV show 'Monty Python's Flying Circus'. Guido van Rossum, the creater of Python was a big fan of the TV show. He wanted to name his invention with a short, unique and slightly mysterious name and chose Python as a working title for his ongoing project.

Figure 1.1: Retrieved from: https://www.r-project.org/logo/

Figure 1.2: Retrieved from: https://www.python.org/community/logos/

1.4 Logo

1.5 Worldwide Google Trends



1.6 Installation

1.6.1 R

You can download it for free from the following websites: - R (https://cran.r-project.org/)

Install Rstudio IDE

RStudio is an integrated development environment for R, a programming language for statistical computing and graphics. It is available in two formats: RStudio Desktop is a regular desktop application while RStudio Server runs on a remote server and allows accessing RStudio using a web browser.

- RStudio (https://www.rstudio.com/products/rstudio/download/#download).

1.6.2 Python

Ref: https://www.w3schools.com/python/python_getstarted.asp

Many PCs and Macs will have python already installed.

To check if you have python installed on a Windows PC, search in the start bar for Python or run the following on the Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

To check if you have python installed on a Linux or Mac, then on linux open the command line or on Mac open the Terminal and type:

```
python --version
```

If you find that you do not have python installed on your computer, then you can download it for free from the following website: https://www.python.org/

Install PyCharm IDE

PyCharm is a cross-platform IDE that provides consistent experience on the Windows, macOS, and Linux operating systems.

PyCharm is available in three editions: Professional, Community, and Edu. The Community and Edu editions are open-source projects and they are free, but they have fewer features. PyCharm Edu provides courses and helps you learn programming with Python. The Professional edition is commercial, and provides an outstanding set of tools and features. For details, see the editions comparison matrix.

Jupyter Notebook

Is Anaconda necessary to be installed to PC besides Jupyter Notebook?

source

Is there a way to install python packages within Jupyter Notebook, or should I really install Anaconda? What is the benefit of installing Anaconda (besides Jupyter Notebook)?

The packages in python are generally installed with "pip" (or "conda") in the terminal and are then available regardless where you run your script from. Assuming you don't have multiple python versions set up on your PC, they should then all be available in your jupyter-notebook also.

If you don't want to open another window to do this, you can also run BASH code from Jupyter itself, just start the line with an exclamation mark (!)

Source - How to use Pip from the Jupyter Notebook

```
# Not Recommended
# DON'T DO THIS
!pip install numpy
```

Here is a short snippet that should generally work:

Install a pip package in the current Jupyter kernel

```
import sys
!{sys.executable} -m pip install numpy
```

The benefits of Anaconda are that it bundles everything you need to at least start your more basic projects (python release, basic packages, IDE) and that you can set-up project-specific environments that do not interfere with your system-wide package installations.

1.7 Install and Load Libraries

1.7.1 R

R Packages: A Beginner's Guide https://www.datacamp.com/community/tutorials/r-packages-guide?utm_source=adwords_ppc&utm_campaignid=1655852085&utm_adgroupid=61045434222&utm_device=c&utm_keyword=%2Bload%20%2Bpackage%20%2Br&utm_matchtype=b&utm_network=g&utm_adpostion=&utm_creative=469789579329&utm_targetid=aud-522010995285:kwd-589281898774&utm_loc_interest_ms=9071445&utm_loc_physical_ms=1009919&gclid=Cj0KCQjwyZmEBhCpARIsALIzmnKGh4ZVHa4OxhLq0JUzpoBMMRhQvCGEmvscwcB

An **R package** is a way to organize your own work and share it with others. Typically, a package contains code, documentation for the package and the

functions inside, some tests to check everything works as it should, and data sets.

Three of the most popular repositories for R packages are: CRAN, Bioconductor and Github.

1.7.1.1 Installing Packages From CRAN

```
install.packages("package_name")
```

Example

```
install.packages("tidyverse")
```

After running this, some messages will be diplayed on the console. They will depend on what operating system you are using, the dependencies, and if the package was successfully installed.

To install more than a package at the same time, we can use a character vector

```
install.packages(c("vioplot", "MASS"))
```

The function `install.packages` will download the source code from on the CRAN mirrors and install the package (and any dependencies) locally on your computer.

You have to install a package only once.

1.7.1.2 Load Packages

After a package is installed, you are ready to use its functionalities.

If you just need a sporadic use of a few functions or data inside a package you can access them with the notation

`packagename::functionname()`.

If you will make a more intensive use of the package, then maybe is worth to load it into memory. The simplest way to do this is with the `library()` command.

Please note that the input of install.packages() is a character vector and requires the name to be in quotes, while library() accepts either character or name and makes it possible for you to write the name of the package without quotes.

Once you have the package installed, you can load the library into your R session for use. Any of the functions that are specific to that package will be available for you to use by simply calling the function as you would for any of the base functions. Note that quotations are not required here.

```
library(tidyverse)
```

1.7.2 Python

Use 'import module' or 'from module import'? https://stackoverflow.com/questions/710551/use-import-module-or-from-module-import

Method 1: `import module`

Method 2: `from module import foo`

The difference between `import module` and `from module import foo` is subjective. User can select one method and be consitent in the use of it.

`import module`	`from module import foo`
Pros	**Pros**
- Less Maintanence of the import statements	- Less typying to use `foo` function
- Don't need to add any aditional imports to start using another item from the same module	- More control over whcih items of the module can be accessed
Cons	**Cons**
- Typing `module.foo` in the code be tedious (dull, boring)	to use new items from the module the user have to update the `import` statement

- It can be minimized by using `import module as mo`, then typing `mo.foo` | You loose context about `foo`. For example it is less clear `ceil()` does, compared to `math.ceil()`

Don't use

- `from modle import *`
  - Because it clutters or fills with untidy collection of things in the namespace
- `import *`
  - For any reasonable large set of code, if you `import *` you will likely be cementing it into the module, unable to be removed.
  - This is because now it is difficullt to identiify what items used in the code are coming from `module`.

1.8 Ranked:15Python packages

for Data Science

http://blog.thedataincubator.com/wp-content/uploads/2017/04/Ranked-15-Python-Packages-for-Data-Science.pdf

Chapter 2

Variables, expressions, and statements

2.1 Basic Exmaple

This is a test code

2.1.1 R code

```
# This is an R code
x <- 1
y <- 3
print(x+y)
```

```
## [1] 4
```

2.1.2 Python Code

The 'python' engine in knitr requires the `reticulate` package.

```
library(reticulate)
```

```
# This is a Python code
x = 1
y = 3
print(x+y)
```

```
## 4
```

Chapter 3

Conditional execution

WIP

Chapter 4

Functions

WIP

Chapter 5

Iteration

WIP

Chapter 6

Tidy workflow

WIP

Moving from R to Python: The Libraries You Need to Know https://www.kdnuggets.com/2017/02/moving-r-python-libraries.html

Chapter 7

Import

WIP

Chapter 8

Tidy

WIP

Chapter 9

Transform

WIP

Chapter 10

Data Visualization

10.1 Data

The Palmer penguins dataset was introduced by Allison Horst, Alison Hill, and Kristen Gorman provide a great dataset for data exploration and visualization, as an alternative to iris. It was first introduced as an R package. The released version of palmerpenguins can be instaalled from CRAN with:

R Installation `install.packages("palmerpenguins")`

Using `palmerpenguins` python package you can easily load the Palmer penguins into your python environment.

Python Installation `pip install palmerpenguins`

The palmerpenguins package contains two datasets : `penguins` and `penguins_raw`. `penguins` is a simplified version of the `penguins_raw` data.

10.2 R

Load data

```
# Load Palmer Archipelago (Antarctica) Penguin Data
library(palmerpenguins)
# Return the first part of the dataset
head(penguins)
```

```
## # A tibble: 6 x 8
##   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~
## 6 Adelie  Torge~           39.3          20.6               190        3650 male
## # ... with 1 more variable: year <int>
```

```
# Retrieve column names
colnames(penguins)
```

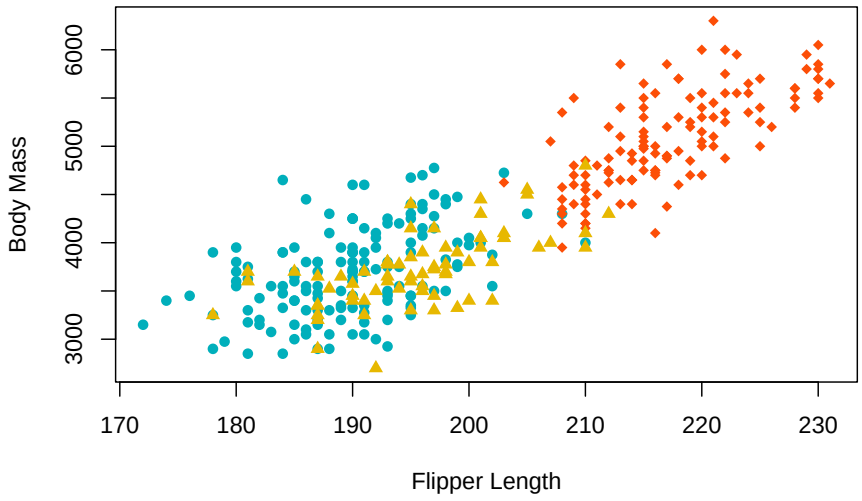
```
## [1] "species"           "island"            "bill_length_mm"
## [4] "bill_depth_mm"     "flipper_length_mm" "body_mass_g"
## [7] "sex"               "year"
```

10.2.1 base R package

```
# Define color for each of the 3 penguine species
colors <- c("#00AFBB", "#E7B800", "#FC4E07")
colors <- colors[as.numeric(penguins$species)]

# Define shapes
shapes = c(16, 17, 18)
shapes <- shapes[as.numeric(penguins$species)]

plot(x = penguins$flipper_length_mm,
          y = penguins$body_mass_g,
          col = colors,
          pch = shapes,
          xlab = "Flipper Length",
          ylab = "Body Mass" )
```

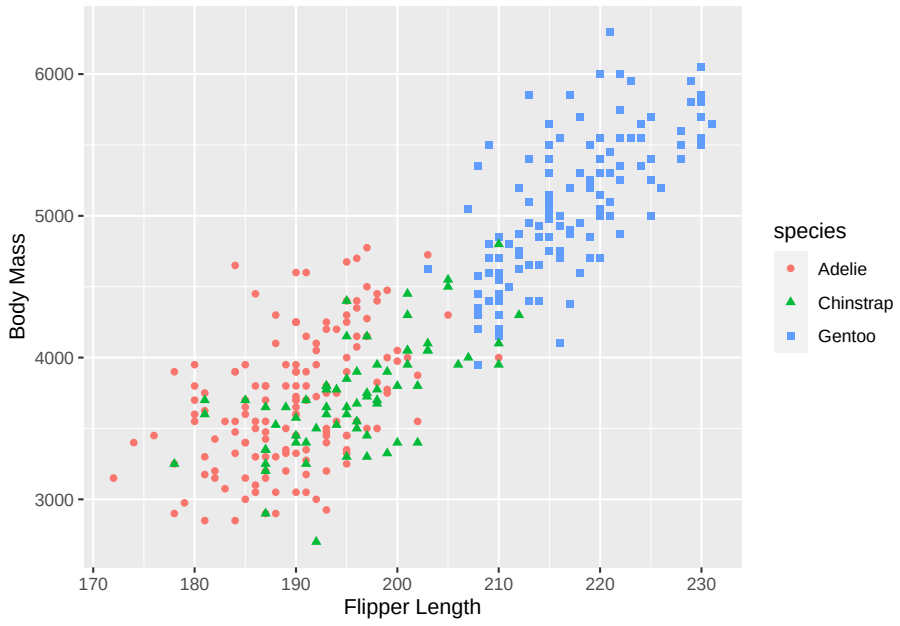


10.2.2 gggplot2 Package

ggplot2 is an R package dedicated to data visualization which is based on The Grammar of Graphics (Wilkinson, 2012).

```
#load ggplot2 package to make statistical graphics
library(ggplot2)
p <- ggplot(penguins) +
  geom_point( aes(x = flipper_length_mm,
                  y = body_mass_g,
                  color = species,
                  shape = species)) +
  xlab("Flipper Length")+
  ylab("Body Mass")

print(p)
```



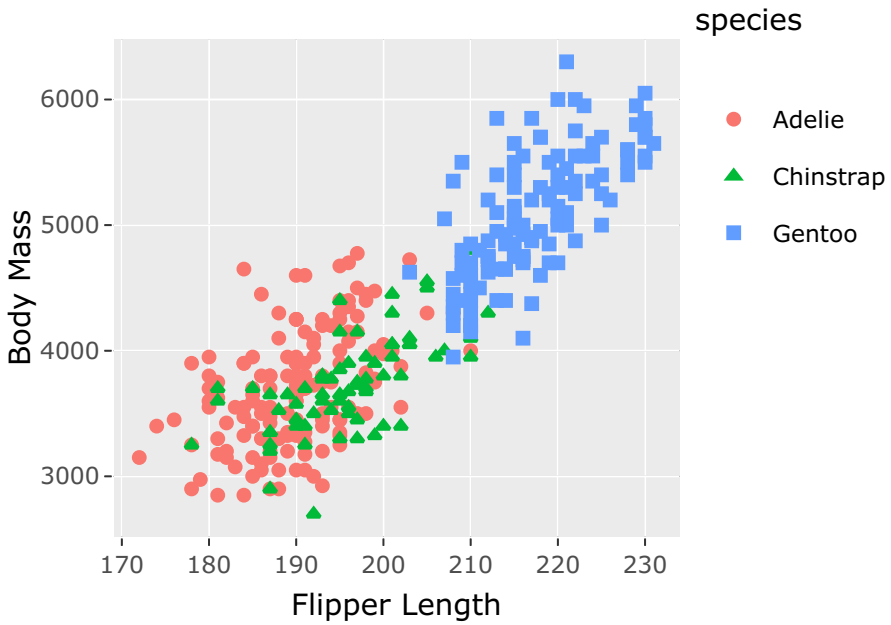
10.2.3 `plotly` R package for interactive data visualization

Interactive visualization focuses on graphic representations of data that improve the way we interact with information

plotly is an R package for creating interactive web-based graphs via the open source JavaScript graphing library plotly.js.

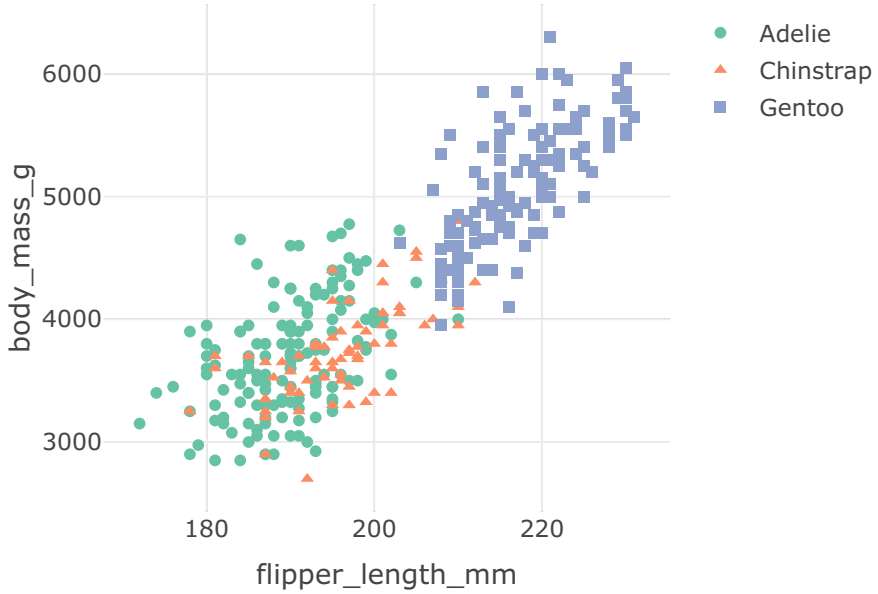
```
library(plotly)
p <- ggplot(penguins) +
  geom_point( aes(x = flipper_length_mm,
                  y = body_mass_g,
                  color = species,
                  shape = species)) +
  xlab("Flipper Length")+
  ylab("Body Mass")

# The function ggplotly converts a ggplot2::ggplot() object to a plotly object.
plotly::ggplotly(p)
```



Method 2

```
library(plotly)
fig <- plot_ly(penguins,
               x = ~flipper_length_mm,
               y = ~body_mass_g,
               color = ~species,
               symbol = ~species,
               type = "scatter")
fig
```



10.3 Python

Load data

```
#load functions in palmerpenguins package
from palmerpenguins import load_penguins
penguins = load_penguins()
# Return the first part of the dataset
penguins.head()
# Retrieve column names
list(penguins.columns)
```

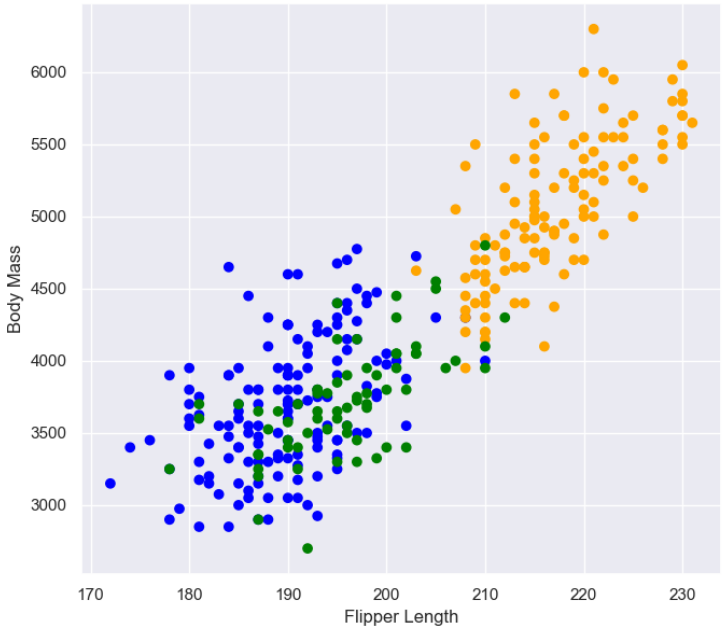
10.3.1 `Matplotlib` package

Matplotlib is mainly deployed for basic plotting. Visualization using Matplotlib generally consists of bars, pies, lines, scatter plots and so on.

```
# Import matplotlib to make statistical graphics.
# By convention, it is imported with the shorthand sns.
import matplotlib.pyplot as plt

colors = {'Adelie':'blue', 'Gentoo':'orange', 'Chinstrap':'green'}
```

```
plt.scatter(penguins.flipper_length_mm,
penguins.body_mass_g,
c= penguins.species.apply(lambda x: colors[x]))
plt.xlabel('Flipper Length')
plt.ylabel('Body Mass')
```



10.3.2 `seaborn` Package

Seaborn is an easy-to-use high level statistical plotting library which provides a variety of visualization patterns. It uses fewer syntax and has easily interesting default themes.

It tries to provide a 'grammar of graphics' style way to create plots but in a pythonic style without getting the exact syntax from ggplot as in plotnine.

Introduction to Seaborn

```
# Import seaborn to make statistical graphics.
# By convention, it is imported with the shorthand sns.
import seaborn as sns
```

```
#load functions in palmerpenguins package
from palmerpenguins import load_penguins
penguins = load_penguins()

# Apply the default theme
sns.set_theme()
# sns.set_style('whitegrid')
p = sns.relplot(x = 'flipper_length_mm',
            y ='body_mass_g',
            hue = 'species',
            style = 'species',
            data = penguins)
p.set_xlabels('Flipper Length')
p.set_ylabels('Body Mass')
```

The function `relplot()` is named that way because it is designed to visualize many different statistical relationships. The `relplot()` function has a convenient kind parameter that lets you easily switch to this alternate representation: `scatterplot()` with `kind="scatter"`; the default and `lineplot()` with `kind="line"`.

10.3.3 `plotnine` package

https://pypi.org/project/plotnine/

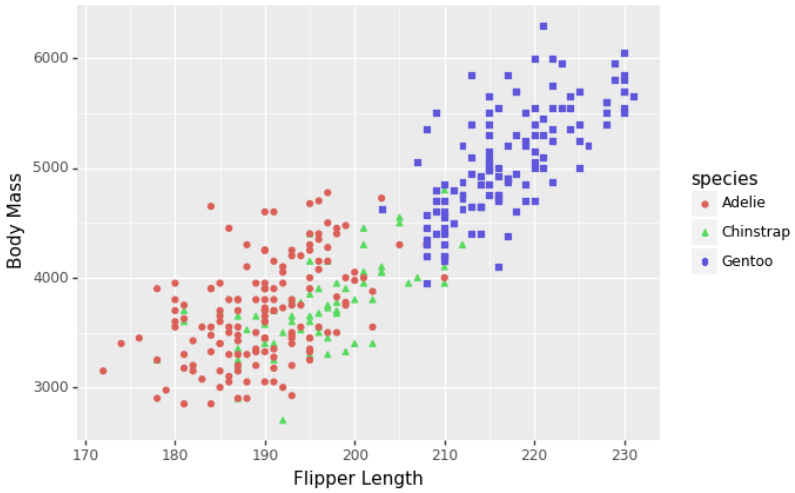
plotnine is an implementation of a grammar of graphics in Python, it is based on ggplot2. The grammar allows users to compose plots by explicitly mapping data to the visual objects that make up the plot.

Plotting with a grammar is powerful, it makes custom (and otherwise complex) plots are easy to think about and then create, while the simple plots remain simple.

NOTE: R vs Python Syntax

Unlike in R, now all the variables must be enclosed by single quotes

```
from plotnine import *
# unlike in R, now all the variables must be enclosed by single quotes
(ggplot(penguins) +
  geom_point(aes(x = 'flipper_length_mm',
                 y = 'body_mass_g',
                 color = 'species',
                 shape = 'species')) +
  xlab("Flipper Length")+
  ylab("Body Mass"))
```



10.3.4 `plotly` Python library for interactive data visualization

The `plotly.express` (Plotly Express or PX) module contains functions that can create entire figures at once. It is usually imported as `px`. Plotly Express is a built-in part of the `plotly` library.

```
import plotly.express as px

fig = px.scatter(penguins,
                 x="flipper_length_mm",
                 y="body_mass_g",
                 color= "species",
                 symbol= "species",
                 labels=dict(flipper_length_mm="Flipper Length",
                             body_mass_g="Body Mass"))
fig.show()
```

Chapter 11

Model

WIP

Chapter 12

Communicate

WIP

Chapter 13

Advanced R and Python

WIP

13.1 Time Series Forecasting

R	Python
fable-Forecasting Models for Tidy Time Series	statsmodels- Statistics based models
forecast- Forecasting Functions for Time Series and Linear Models	sktime- A unified framework for machine learning with time series
	GluonTS- Deep learning-based models.

Chapter 14

Jupyter Notebooks

- The Jupyter Notebook is an open-source web application that allows the user to create and share documents that contain live code, equations, visualizations, and narrative text.
- JupyterLab is an advanced version of Jupyter Notebook interface. It brings the classic notebooks, text editor, terminal, and directory viewer all under one roof. However, both operate in a similar fashion.

14.1 How to install Jupyter environment?

First, open a new command prompt (Windows) or terminal (Mac/Linux) on your workstation, and second, execute the following command:

```
jupyter notebook
```

If the above command fails, first, you need to install python on your workstation. There are two popular methods to install Python on your workstation.

- Installing Python using Anaconda Distribution
- Installing Raw Python

After installing python using one of the above methods, then we need to installing Jupyter Notebook using either Anaconda or pip.

14.1.1 Method 1: How to Install the Notebook Using `pip`

- If you don't want to install Anaconda, you just have to make sure that you have the latest version of `pip`.
- If you have installed Python, you will typically already have it.

- You can check the already installed pip version

```
pip3 --version
```

- You can upgrade that using

```
# On Windows
python -m pip install -U pip setuptools

# On OS X or Linux
pip3 install -U pip setuptools
```

- When you install Python directly from its official website, it does not include **Jupyter Notebook** in its standard library.
- In this case, you need to install Jupyter Notebook using the `pip`. The process is as follows:

```
python -m pip install jupyter
```

or if you are using Python 3

```
python3 -m pip install jupyter
```

or simply

```
pip install jupyter
```

Congratulations, you have installed Jupyter Notebook!

After you have installed the Jupyter Notebook on your computer, you are ready to run the notebook server.

14.1.2 Method 2: Installing Jupyter Notebook using Anaconda

- If you have installed Python using Anaconda Distribution, then it includes Python, the Jupyter Notebook, and other commonly used packages for the scientific community.
- You can follow the instructions for the installation of Anaconda here for Mac: click here or Windows: click here.

14.2 Working with Jupyter Notebook

This video: (click here) will guide you to create your first jupyter notebook.

Chapter 15

Working with jupyter notebook

1. open terminal
2. Type `jupyter notebook`
3. Go to –> new –> folder (or select and exiting folder from the files list)
4. Then creat jupyter notebook new–> Python 3

Chapter 16

Working with pycharm

1. First install pycharm
2. Open pycharm
3. Create a project (select a location)
4. Open terminal
5. Type `jupyter notebook`
6. Create jupyter notebook
7. Quit notebook

16.1 install packages

```
python -m pip install <package>
```

must read what toavoide: https://jakevdp.github.io/blog/2017/12/05/installing-python-packages-from-jupyter/

16.2 Install Python package using Jupyter Notebook

- https://www.geeksforgeeks.org/install-python-package-using-jupyter-notebook/

```
import sys !{sys.executable} -m pip install [package_name]
```

Bibliography

R Core Team (2020). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Wikipedia contributors (2020a). Python (programming language) — Wikipedia, the free encyclopedia. [Online; accessed 25-December-2020].

Wikipedia contributors (2020b). R (programming language) — Wikipedia, the free encyclopedia. [Online; accessed 25-December-2020].

Wilkinson, L. (2012). The grammar of graphics. In *Handbook of computational statistics*, pages 375–414. Springer.