RRRRRRRRRIDICULOUS LANGUAGE WARS

R AND PYTHON BF4EVER

PETER MEISSNER

2019-06-27

(UPDATED: 2019-06-25)

Chapter I

Where I tell you that winning is about not fighting.

LETS MAKE IT SHORT

Use R for everything

LETS MAKE IT SHORT

Use Python for everything

BATTLE OF THE NON-COMBATANTS

performance

programming language

quirky

learning curve

programming paradigm

R AND PYTHON

scripting languages

easy to use

multi paradigm

rule statistics, analytics, data

science, AI

Chapter II

Where I ~~tell~~ show you that winning is about not fighting.

DATA.FRAMES

R

```
df =
  data.frame(
    one   = rnorm(5),
    two   = rnorm(5),
    three = rnorm(5)
  )

df
```

```
##      one    two   three
## 1  0.44 -0.961 -1.090
## 2  0.62  0.077 -0.164
## 3 -0.47 -0.277  0.019
## 4  1.91  0.582 -0.740
## 5  0.19  1.038 -0.984
```

```
df$one %>% sum()
```

```
## [1] 2.7
```

Python

```
import numpy as np
import pandas as pd

df = pd.DataFrame(
  np.random.randn(5, 3),
  columns = ['one', 'two', 'three']
)

df
```

```
##         one       two     three
## 0  1.156346 -0.364994  1.453211
## 1 -0.382714  0.481290 -0.178083
## 2 -1.266491  0.841089 -0.279913
## 3  1.289428 -1.485019  0.068107
## 4 -0.269928 -0.061806  0.733244
```

```
df.one.sum()
```

```
## 0.5266415015791044
```

PLOTS

R

```
# Sequence of numbers
x = seq(from = -10, to = 10, by = 0.01)

# Mathematical function
y = sin(x)

# Plot
plot(x, y, type = "l")
```

Python

```
import matplotlib.pyplot as plt
import numpy as np

# Sequence of numbers
x = np.arange(-10,10.001, 0.01)

# Mathematical function
y = np.sin(x)

# Plot
plt.plot(x, y)
plt.show()
```
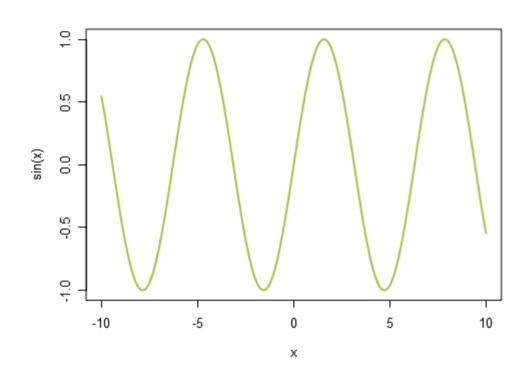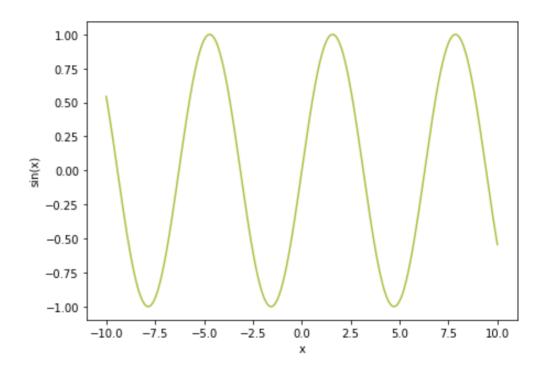
P L O T S

R

Python

DATA IMPORT

R

```r
flights <-
  read.csv(
    "data/flights.csv"
  )

planes <-
  read.csv(
    "data/planes.csv"
  )
```

Python

```python
import pandas as pd

flights = pd.read_csv("data/flights.csv")
planes  = pd.read_csv("data/planes.csv")
```

DATA MANAGEMENT

R

```r
library(dplyr)

# Joins
flight_data <-
  left_join(
    x  = flights,
    y  = planes,
    by = "tailnum"
  ) %>%
# Manipulation
  mutate(
    delay =
      ifelse(dep_delay <= 0, 0, 1)
  )
```

Python

```python
# Joins
flight_data = pd.merge(
  left  = flights,
  right = planes,
  on    = 'tailnum',
  how   = 'left'
)

# Manipualtion
flight_data.loc[
  flight_data.arr_delay <= 0,
  'delay'
  ] = 0

flight_data.loc[
  flight_data.arr_delay > 0,
  'delay'
] = 1
```

MACHINE LEARNING

R

```
# bring data into right shape
```

Python

```python
import numpy as np
import pandas as pd
from statsmodels.api import OLS

# bring data into right shape
dat  = flight_data[['delay', 'seats', 'dep_delay']]
dat.loc[flight_data['manufacturer'] == "AIRBUS", 'AIRBUS'] = 1
dat.loc[flight_data['manufacturer'] == "AIRBUS", 'AIRBUS'] = 0
dat.loc[flight_data['manufacturer'] == "EMBRAER", 'EMBRAER'] = 1
dat.loc[flight_data['manufacturer'] == "EMBRAER", 'EMBRAER'] = 0
dat.loc[flight_data['manufacturer'] == "BOEING", 'BOEING'] = 1
dat.loc[flight_data['manufacturer'] == "BOEING", 'BOEING'] = 0
dat.loc[flight_data['manufacturer'] == "MCDONNELL DOUGLAS", 'MCDoug
dat.loc[flight_data['manufacturer'] == "MCDONNELL DOUGLAS", 'MCDoug
dat = dat.dropna()

X   = dat[['seats', 'dep_delay', 'AIRBUS',
'BOEING', 'EMBRAER', 'MCDoug']]
y   = dat[['delay']]
```

MACHINE LEARNING

R

```
# fit model
lm(
  delay ~ 0 + dep_delay + seats + manufacturer,
  data = flight_data
) %>%
  summary()
```

```
##
## Call:
## lm(formula = delay ~ 0 + dep_delay + seats + manufacturer, data = flight_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -7.778 -0.309 -0.263  0.495  0.781
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## dep_delay                  0.0065240  0.0000240  272.22  < 2e-16 ***
## seats                     -0.0001097  0.0000251   -4.37 0.000012 ***
## manufacturerAIRBUS         0.3312337  0.0054662   60.60  < 2e-16 ***
## manufacturerBOEING         0.3936656  0.0046100   85.39  < 2e-16 ***
## manufacturerEMBRAER        0.3077906  0.0020104  153.10  < 2e-16 ***
## manufacturerMCDONNELL DOUGLAS 0.2316267 0.0079968  28.97  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.42 on 197175 degrees of freedom
##   (139595 observations deleted due to missingness)
## Multiple R-squared:  0.58,    Adjusted R-squared:  0.58
## F-statistic: 4.53e+04 on 6 and 197175 DF,  p-value: <2e-16
```

Python

```
# fit model
OLS(y, X).fit().summary()
```

```
## <class 'statsmodels.iolib.summary.Summary'>
## """
##                            OLS Regression Results
## ==============================================================================
## Dep. Variable:                  delay   R-squared:                       0.209
## Model:                            OLS   Adj. R-squared:                  0.209
## Method:                 Least Squares   F-statistic:                 1.040e+04
## Date:                Tue, 25 Jun 2019   Prob (F-statistic):               0.00
## Time:                        17:50:14   Log-Likelihood:            -1.1676e+05
## No. Observations:              196542   AIC:                         2.335e+05
## Df Residuals:                  196536   BIC:                         2.336e+05
## Df Model:                           5
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## seats         4.755e-05   2.62e-05      1.813      0.070   -3.85e-06   9.89e-05
## dep_delay       0.0057   2.52e-05    225.671      0.000       0.006      0.006
## AIRBUS          0.3380      0.006     59.121      0.000       0.327      0.349
## BOEING          0.3158      0.005     65.500      0.000       0.306      0.325
## EMBRAER         0.3614      0.002    171.838      0.000       0.357      0.366
## MCDoug          0.2588      0.008     30.901      0.000       0.242      0.275
## ==============================================================================
## Omnibus:                    12270.124   Durbin-Watson:                   1.728
## Prob(Omnibus):                  0.000   Jarque-Bera (JB):             9547.001
## Skew:                           0.451   Prob(JB):                         0.00
## Kurtosis:                       2.405   Cond. No.                     1.57e+03
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
## [2] The condition number is large, 1.57e+03. This might indicate that there are
## strong multicollinearity or other numerical problems.
## """
```

virtual7

What we need is semantics!
Syntax is a second thought!
Different syntax, same semantics!

And some features.

¯_(ツ)_/¯

ACCEPT AND EMBRACE
R VERSUS PYTHON

⚡ traditions ⚡

biology, social sciences, statistics, math ⚡ physics, computer science

⚡ strengths ⚡

data analytics focused ⚡ everbodies darling

⚡ user base ⚡

focussed ⚡ larger

ACCEPT AND EMBRACE

R AND PYTHON

♥ **Overlap**

data science, stats, AI

♥ **Inspiration**

data.frame, data.table, ggplot, Shiny,
SciKit learn, web scraping, Jupyter notebooks,
Rmarkdown

Chapter III
Where I sound like sales person.

LOOKING ON THE BRIGHT SIDE

A+ TOOLING

- **JuPyteR**-Notebooks
- R, Python, SQL, Scala, Java integration in Spark
- Stan has interfaces for R, Python, shell, MATLAB, Julia, Stata
- wrappers, portations, adaptions
- attempts to have multilanguage support in IDEs
- Rstudio
  - ...

MULTILANGUAGE DATASCIENCE IDES
WHAT DO WE WANT?

must have

- all data science languages
- multiple workflows
  - exploring
  - reporting
  - developing
- responsive, zoomable, good looking

enterprise goodies

- remote developement
- deployment
- authentication and security

MULTILANGUAGE DATASCIENCE IDES

ALL DATA SCIENCE LANGUAGES AND MULTIPLE WORKFLOWS

- support for R, Python, JS, SQL, C++, Stan, D3.js, CSS
- syntax higlighting for many more
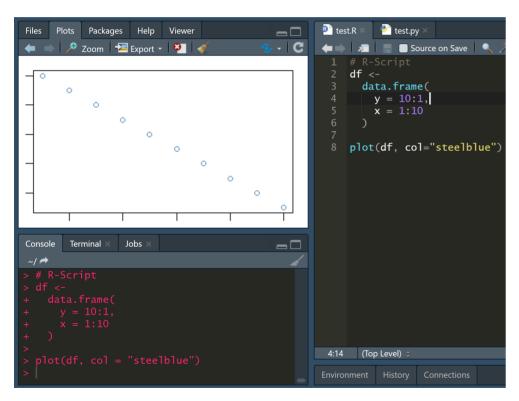- script, notebook, web app, REST API, reports, books, presentations

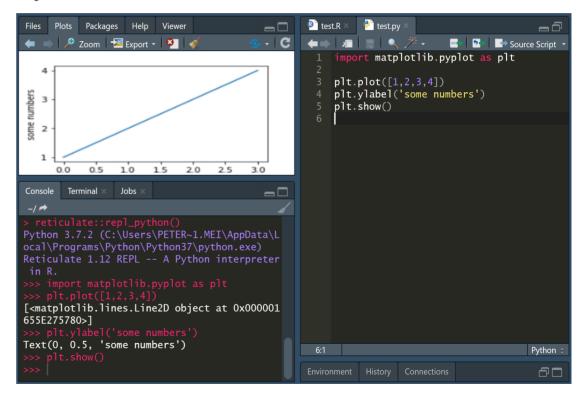|	
R Script	Ctrl+Shift+N
R Notebook	
R Markdown...	
Shiny Web App...	
Plumber API...	
Text File	
C++ File	
Python Script	
SQL Script	
Stan File	
D3 Script	
R Sweave	
R HTML	
R Presentation	
R Documentation	

INTERACT/EXPLORE
REPL

R

Python

IDE

INTEGRATED DEVELOPEMENT ENVIRONMENT

- **integrated** developement **environment**

- plots
- help
- backround jobs
- cluster jobs
- notebooks
- reports
- git
- data base connections
- spark connections
- debugging
- profiling

{RETICULATE}

- a package
- control over Python session
- control over binary and environment (e.g. virtualenv, conda)
- make python objects available within your R session
- gives you a Python-REPL

Chapter IV
Where I talk about standardization and things to come.

DATA.FRAMES

data.frames

=

columnar data

used everywhere in data science

SQL table, R data.frame, Python pandas data.frame, Spark DataFrame

passed around a lot

DATA.FRAMES
APACHE ARROW

- in-memory format
- for columnar data
- with interfaces to
  - R, Python, C++, Java, Rust, Go, …, many more
- fast passing around
- no passing around
- implementation of standard functionality

DATA.FRAMES

APACHE ARROW TO THE RESCUE

- Speedups for Spark
    - Python
    - R

- Data Exchange between R/Python

DATA.FRAMES
APACHE ARROW TO THE RESCUE

R

```
library(feather)

system.time({
  write.csv(nycflights13::flights, "data/flights.csv")
  tmp <- read.csv("data/flights.csv")
})
```

```
##     user  system elapsed
##    10.45    0.39    11.02
```

```
system.time({
  write_feather(nycflights13::flights, "data/flights.fth")
  read_feather("data/flights.fth")
})
```

```
##     user  system elapsed
##     0.08    0.06     0.14
```

```
fs::file_info(
  c("data/flights.csv", "data/flights.fth")
)$size
```
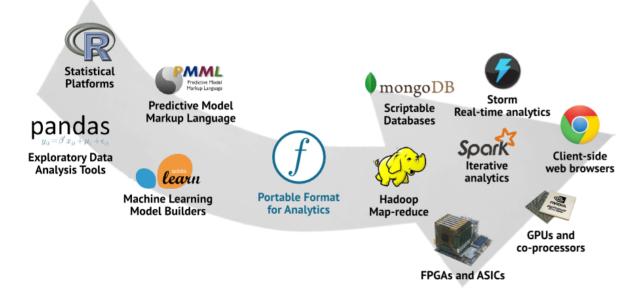
```
## 35M 38.1M
```

Python

```python
import feather
import pandas as pd
import time

start_time = time.time()
df = pd.read_csv("data/flights.csv")
df.to_csv("data/flights.csv")
print("%.2f s" % (time.time()-start_time))
```

```
## 12.00 s
```

```python
start_time = time.time()
df.to_feather("data/flights.fth")
df = pd.read_feather("data/flights.fth")
print("%.2f s" % (time.time()-start_time))
```

```
## 0.37 s
```

MACHINE LEARNING MODELS

LANGUAGE AGNOSTIC MODEL DESCRIPTIONS

- nothing like Apache Arrow - yet
- but there are formats
  - MLeap (Spark)
  - PFA (Data Mining Group)
  - ???
- now that we can pass around data this is the next frontier

Chapter V

Where I close up with wise words and a final joke.

LETS START OVER

Demand Features

data.frames, data.management, algebra, plots, algorithms

Expect Tooling

exploration, reporting, developing

Look Forward to

Infrastructure for Core Functionality

data, machine learning algortihms

Make data ❤ science not language wars.

THE ENEMY IS

~~Java~~

~~JavaScript~~

~~C++~~

~~Pearl~~

All the problems we have not solved - yet!

But
In 5 Years Time

EVERYTHING IS
JAVASCRIPT
anyways

NOTES

- Using the pipe operator (`%>%`, `Ctrl+Shift+M`) needs magrittr to be loaded - like that:

```
library(magrittr)
```

- REPL: Read-Eval-Print-Loop (Running you code in a interactively in a terminal)

- **number of recompilations:** 113