virtual7 | CONFERENCE 2019

RRRRRRRRIDICULOUS LANGUAGE WARS

R AND PYTHON BF4EVER

PETER MEISSNER

2019-06-27

(UPDATED: 2019-06-26)

Chapter 1

Where I preach that winning is about not fighting.

Use R for everything

Use Python for everything

BATTLE OF THE NON-COMBATANTS

- performance
- real programming language
- quirky
- learning curve
- ...

R AND PYTHON

- scripting languages
- easy to use
- multi paradigm
- powerful functions
- **rule statistics, analytics, data science, AI**
- come to stay

Chapter 2

Where I ~~preach~~ 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  0.754 0.679  1.36
## 2  0.749 0.343 -0.64
## 3 -2.258 0.014 -0.14
## 4  0.083 0.757  0.49
## 5  0.485 0.970 -0.35
```

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

```
## [1] -0.19
```

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 -0.683684  0.975936  1.951668
## 1 -0.723863 -0.830057 -0.700345
## 2 -1.278534  1.271243 -0.407306
## 3 -0.688779  0.444908 -0.713338
## 4 -0.175957  1.189581  1.587170
```

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

```
## -3.550817029032714
```

PLOTS

R

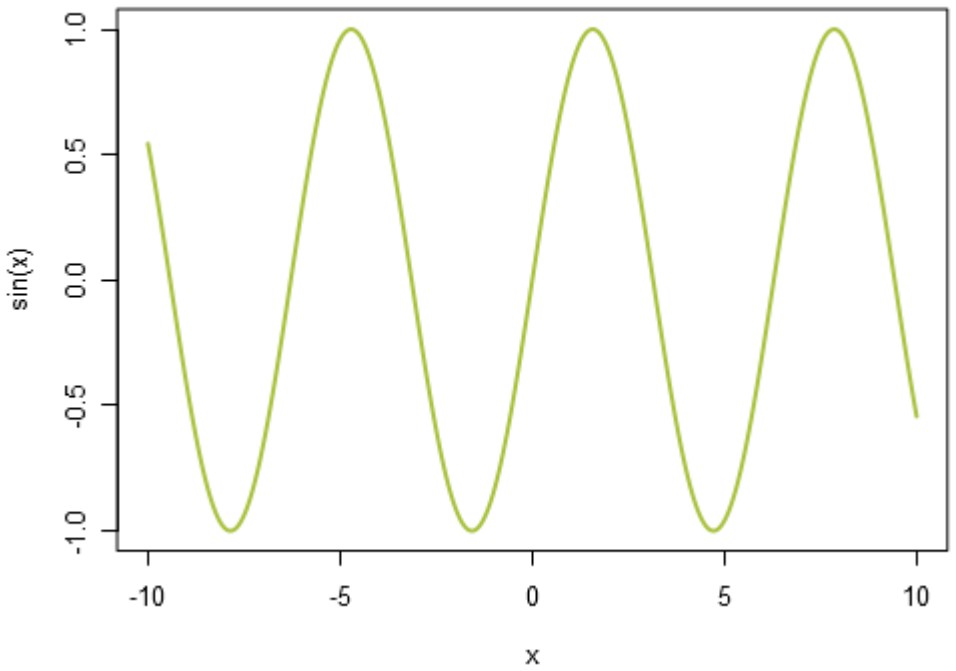
```r
# Sequence of numbers
x = seq(from = -10, to = 10, by = 0.01)

# Mathematical function
y = sin(x)

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

Python

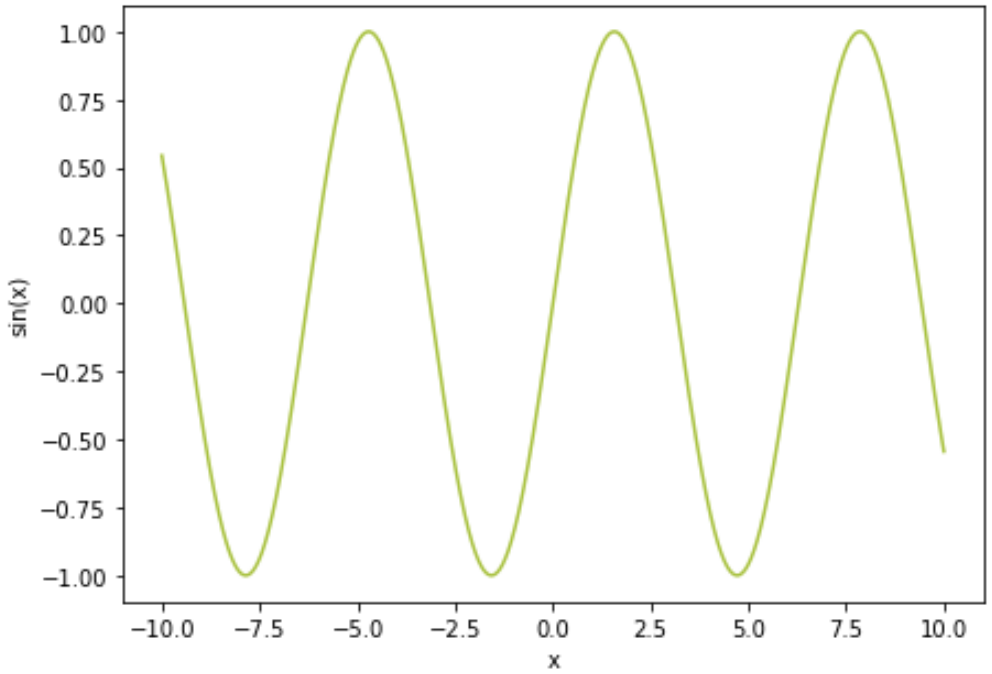
```python
import matplotlib.pyplot as plt
import numpy as np

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

# Mathematical function
y = np.sin(x)

# Plot
plt.plot(x, y)
plt.show()
```

PLOTS

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

```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

```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

```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:                Wed, 26 Jun 2019   Prob (F-statistic):               0.00
## Time:                        10:39:27   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.
## """
```

WHAT REALLY MATTERS

virtual7

What we need is semantics!

Syntax is a second thought!

Different syntax, same semantics!

And some essential data science features.

¯\(ツ)/¯

Chapter 3

Where I finaly do a R versus Python like everybody else .

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

Chapter 4

Where I sound like a sales person for Rstudio.

LOOKING ON THE BRIGHT SIDE

A+ TOOLING

- **J**u**P**yte**R**-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

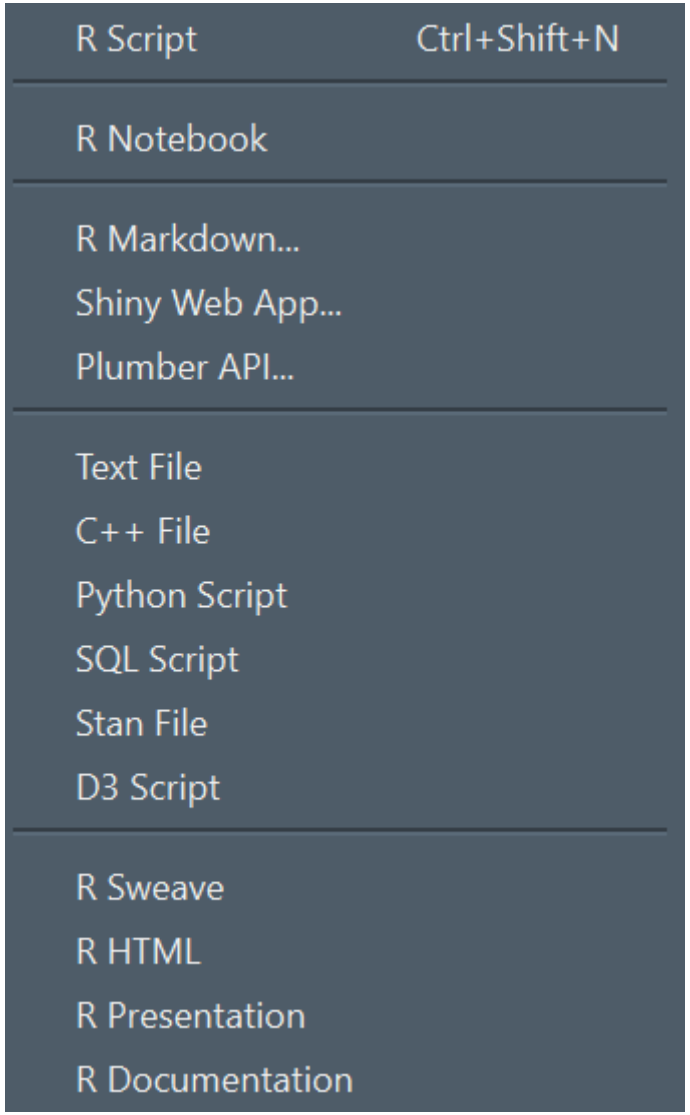
enterprise goodies

- remote developement
- deployment
- authentication and security

MULTILANGUAGE DATASCIENCE IDES

ALL DATA SCIENCE LANGUAGES AND MULTIPLE WORKFLOWS

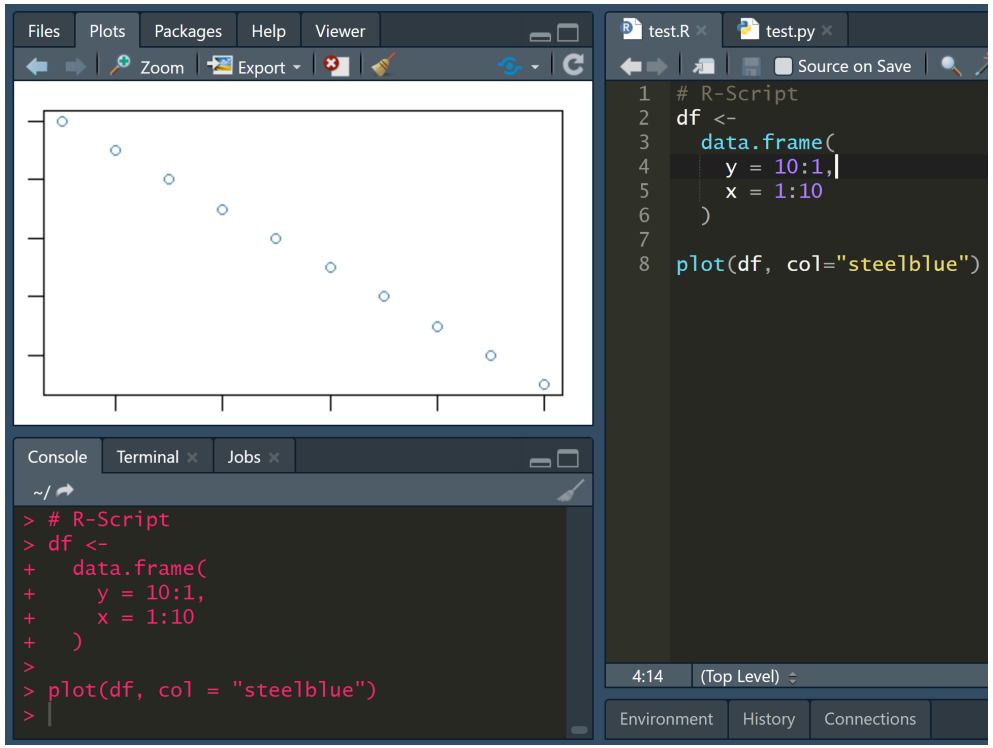
- support for R, Python, JS, SQL, C++, Stan, D3.js, CSS
- syntax higlighting for many more
- script, notebook, web app, REST API, reports, books, presentations



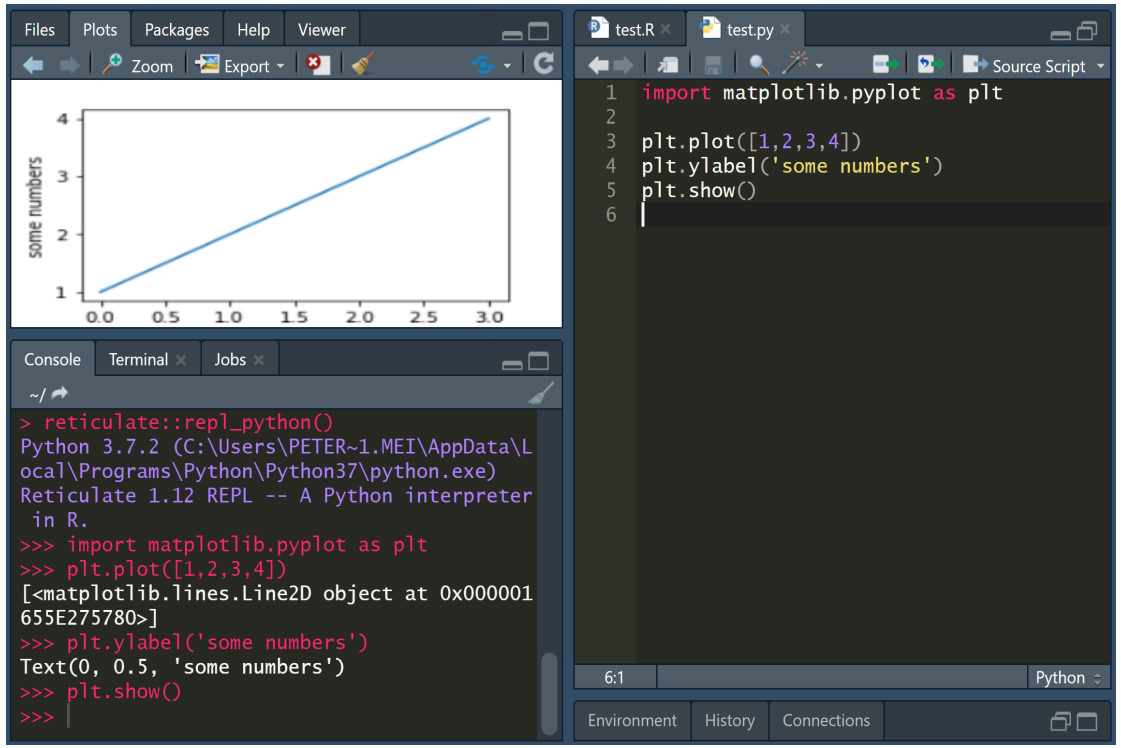
INTERACT/EXPLORE

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 5

Where I talk about standardization and things to come.

DATA SCIENCE, ML, KI

=

- data import/export
- data management
- plots
- **data.frames**
- **algorithms for models**

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

```r
library(feather)

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

```
##    user  system elapsed
##    8.80    0.19    9.05
```

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

```
##    user  system elapsed
##    0.06    0.06    0.13
```

```r
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))
```

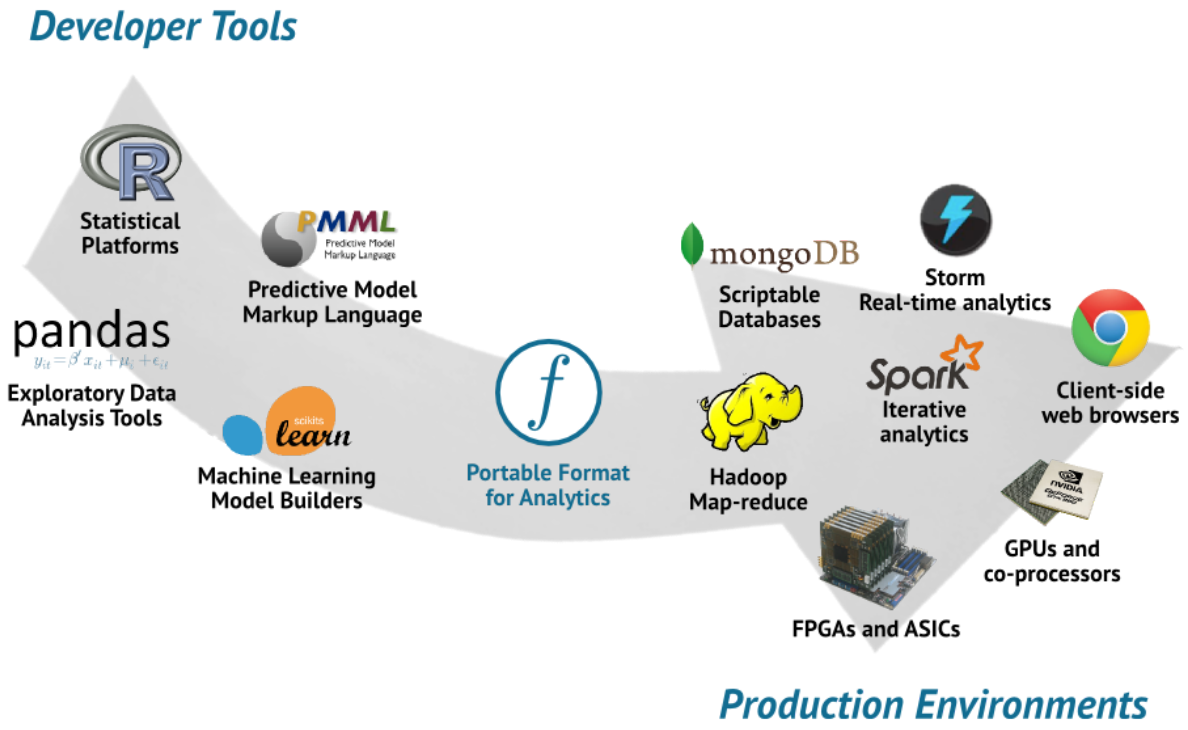
```
## 5.92 s
```

```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))
```

```
## 0.30 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 6

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

LETS START OVER

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:** 129