Winter Institute in Data Science and Big Data

Comparative Computing

Le Bao

Massive Data Institute, Georgetown University

January 6, 2022

Plan

- Python
  - Basics
  - Data structures
  - Application: working with web data

- Comparative computing
  - Python, R, Shell
  - Polyglot programming and computing tools
  - Computing environment

- Containers and cloud computing
  - Operating systems and system dependencies
  - Docker
  - Cloud computing with Code Ocean

Data Science Toolbox

- Some most used DS tools:
  - Apache Spark
  - BigML
  - D3.js
  - MATLAB
  - Excel
  - tidyverse
  - Tableau
  - Jupyter
  - ggplot2
  - Matplotlib
  - NLTK
  - Scikit-learn
  - TensorFlow
  - Weka
  - …

Calculating Path to Jupyter Using Excel

CNBC

Search quo

MARKETS BUSINESS INVESTING TECH POLITICS CNBC TV WATCHLIST CRAMER PRO

INVESTING IN SPACE

NASA spacecraft launches toward Jupiter asteroids on an intricate path charted by Excel

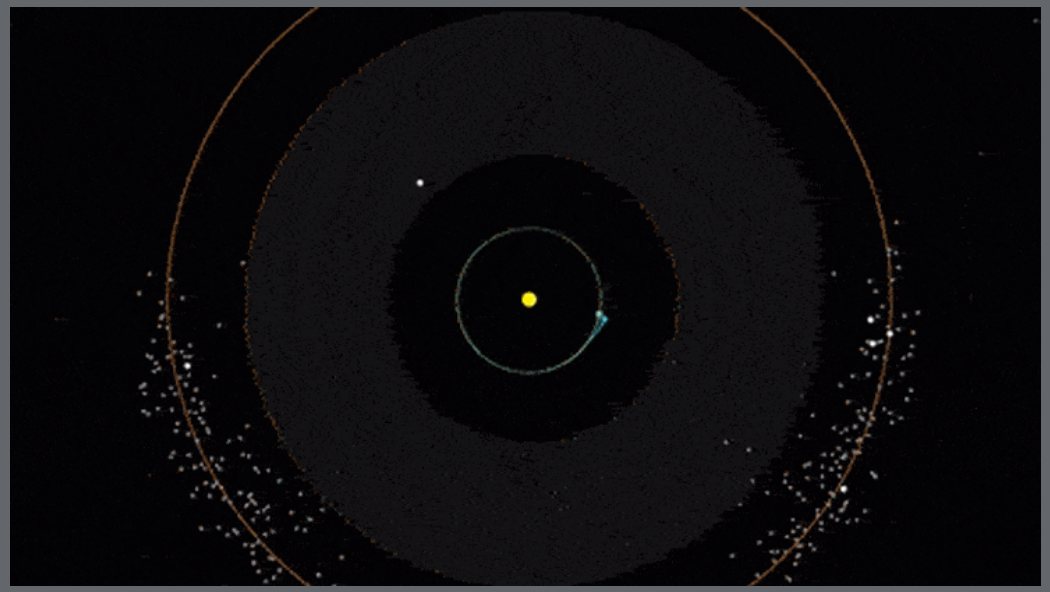
PUBLISHED MON, OCT 18 2021•12:40 PM EDT | UPDATED MON, OCT 18 2021•8:12 PM EDT

Michael Sheetz
@THESHEETZTWEETZ

SHARE

KEY POINTS

- NASA's latest exploration spacecraft launched over the weekend, beginning a 12-year journey to visit Jupiter's Trojan asteroids.
- The mission's intricate path was charted using a tool familiar to Wall Street – Microsoft Excel.
- Lockheed Martin mission architect Brian Sutter explained to CNBC how he used Excel to find which asteroids would be worth visiting, saying he aimed to try "to try to look at as many of these Trojans as we can in a single mission."



Knowing Your Tools

- Recall that data science is an intersection of
  - statistical practice
  - computational tools
  - substantive knowledge

- The **interdisciplinary nature** of data science
  - Different kinds of statistical practices, computational tools, and substantive focuses

- Goal:
  - Know what you are doing: statistical and substantive knowledge
  - Know how to use your tools
  - Pick up new tools fairly quickly

R vs Python

- Let's forget about Stata.
- **Open-source** programming language
- High-level, expressive, front-end
  - Both were (partially) written in C
  - Both were Turing complete
  - Fewer lines of code to achieve complex functions
  - Closer to human languages
  - Slow (relatively)
- Vibrant online communities
  - Libraries, support, new things, etc.

R vs Python

- **R**
  - Statistical analysis
  - Primarily procedural and functional programming, but can also do OOP
  - Academics, researchers, data scientists
  - Easier to get started but some rough curves on the way
  - Good at data wrangling, exploration, modeling, and visualization
  - Can do data scraping, machine learning (deep learning)
    - `caret`, `tidymodels`, `keras`
  - Fewer packages, slow adaptation beyond statistical methods

- **Python**
  - General purpose
  - Procedural, object-oriented, and functional programming
  - Programmers, developers, engineers
  - Deeper curve for beginners but smooth, linear later
  - Good at data collection, interface with different types of data, machine learning (deep learning)
  - Can do data wrangling, modeling, and visualization
    - `NumPy`, `Pandas`, `Seaborn`/`Matplotlib`/`Plotly`
  - Fewer package on modeling (esp. specific models)

R vs Python

- There is *no* programming language wars
- Both can achieve what you want: “Turing complete”
  - Sometimes, one is easier than the other.
- Learn both!
  - Use one when appropriate and to your advantage!

Using Both: Polyglot Programming

- Integrating different languages: wrapper (interface)
- Tools and facilities in programming
  - IDE, code/text editor, interpreter/console
  - Command line and shell
  - Interactive vs non-interactive programming
- Quarto

Wrapper

- **Wrapper**: interface with other languages
- R interface for python: `reticulate`
- Python interface for R: `rpy2`
- R interface for C++: `Rcpp`
- Python interface for C/C++: Python C-API

Wrapper: Example

- Running Python code in R

```
#install.packages("reticulate")
library(reticulate)
# Specify python location and version
use_python("/usr/bin/python3")
# Run one line of python code
py_run_string("print 'Hello World'")
# Install python library
py_install("pandas")
# Use python in R
np <- import("numpy", convert = TRUE)
np1 <- np$array(c(1:4))
np1
# Source python script
source_python("somepythoncode.py")
```

Tools

- **Integrated Development Environment (IDE)**
  - R: RStudio (dominant), but there are others (VSCode)
  - Python: Spyder, PyCharm, VSCode, and lots of others

- What does it mean by *integrated*?
  - Text/code editor
  - Interpreter/console
  - Other tools: Git, debugging tool, file manager, viewer, etc.

RStudio as an example

RStudio File Edit Code View Plots Session Build Debug Profile Tools Window Help

Flights - RStudio

Go to file/function | Addins | Flights

flights-example.R

Source on Save | Run | Source

```r
library(nycflights13)     ## package containing flights dataset
library(lubridate)
library(dplyr)
library(ggplot2)

head(flights, n = 3)
daily <- flights %>%
  mutate(date = make_date(year, month, day)) %>%
  count(date) %>%
  mutate(wday = wday(date, label = TRUE))
head(daily, n = 3)
ggplot(daily, aes(wday, n)) +
  geom_boxplot(outlier.colour = "hotpink") +
  labs(x = "Weekday", y = "Flights",
       subtitle = "Number of 2013 New York Flights Each Weekday")
```

1:1 (Top Level) R Script

Environment | History | Connections | Tutorial

Import Dataset | 346 MiB | List

R | Global Environment

Data

daily 365 obs. of 3 variables

```
$ date: Date[1:365], format: "2013-01-01" "2013-01-02" …
$ n   : int [1:365] 842 943 914 915 720 832 933 899 902…
$ wday: Ord.factor w/ 7 levels "Sun"<"Mon"<"Tue"<..: 3 …
```

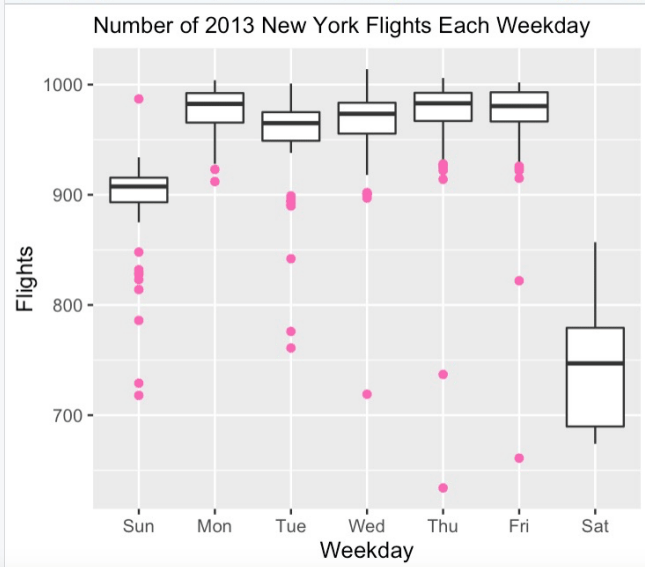
Console | Terminal | Jobs

~/Documents/Flights/

```
# A tibble: 3 x 19
   year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
  <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>     <dbl> <chr>
1  2013     1     1      517            515         2      830            819        11 UA
2  2013     1     1      533            529         4      850            830        20 UA
3  2013     1     1      542            540         2      923            850        33 AA
# … with 9 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
#   distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>
> daily <- flights %>%
+   mutate(date = make_date(year, month, day)) %>%
+   count(date) %>%
+   mutate(wday = wday(date, label = TRUE))
> head(daily, n = 3)
# A tibble: 3 x 3
  date           n wday
  <date>     <int> <ord>
1 2013-01-01   842 Tue
2 2013-01-02   943 Wed
3 2013-01-03   914 Thu
> ggplot(daily, aes(wday, n)) +
+   geom_boxplot(outlier.colour = "hotpink") +
+   labs(x = "Weekday", y = "Flights",
+        subtitle = "Number of 2013 New York Flights Each Weekday")
>
```

Files | Plots | Packages | Help | Viewer

Zoom | Export



Command Line Interface (CLI)

- Command line tools
  - Mac/Linux: Terminal
  - Windows: Command Prompt, PowerShell

- Command-line interface (CLI).
  - CLI vs GUI (Graphical User Interface)

R and Python with CLI

- Mac/Linux:

```
R
python
python3
```

- Windows

```
# Command Prompt
"C:\Program Files\R\R-4.1.2\bin\R.exe"
"C:\Users\[Your User Name]\anaconda3\python.exe"
# PowerShell
& "C:\Program Files\R\R-4.1.2\bin\R.exe"
# For python, Anaconda Prompt
python
```

Non-Interactive Mode

- Interactive vs non-interactive mode
  - Batch execution

- R: source methods

```r
source("script.R")
```

- Python

```python
exec(open("script.py").read())
```

Non-Interactive Mode using Command Line

- Mac/Linux:

```
R CMD BATCH script.R
Rscript script.R
Rscript -e "rnorm(100)"
python script.py
python -c "print('hello world')"
```

- Windows

```
# Command Prompt
"C:\Program Files\R\R-4.1.2\bin\R.exe" CMD BATCH script.R
"C:\Program Files\R\R-4.1.2\bin\Rscript.exe" script.R
"C:\Program Files\R\R-4.1.2\bin\Rscript.exe" -e "rnorm(100)"
#### For PowerShell, remember to add &
"C:\Users\[Your User Name]\anaconda3\python.exe" script.py
"C:\Users\[Your User Name]\anaconda3\python.exe" -c "print('hello world')"
```