



MANGO
SOLUTIONS

Snakes in a Package: combining Python and R

Adnan Fiaz

Data Scientist

✉ afiaz@mango-solutions.com

🐦 [@tapundemek](https://twitter.com/tapundemek)



Introduction

- Data Scientist @ Mango Solutions
- Like to...
 - Make paper helicopters with `library(SixSigma)`
 - Implement bucket brigades
 - Read waitbutwhy.com / what-if.xkcd.com
- Avid user of both Python and R



We're Hiring



Agenda



Python

Reticulate

The mailman package

Bonus



re-tic-u-late (rĭ-tĭk'yə-lĭt, -lāt,)

- Developed by Kevin Ushey & JJ Allaire
- Interoperability Python and R
- Embedded Python session
- Call Python functionality in R



Why reticulate?

- Access solutions already implemented in python
 - Machine Learning: sci-kit learn, tensorflow
 - NLP: nltk, spaCy
 - Image processing: sci-kit image
- Enhance collaboration between developers without re-training



Monty Python's FLYING CIRCUS



What is python?

A way of life

- High-level interpreted ~~statistical~~ *general purpose* programming language
- Comprehensive ~~base~~ *standard* library
- Extendable through packages via ~~CRAN~~ *PyPi*



Zen of Python

- Beautiful is better than ugly.
- ...
- Readability counts.
- ...
- There should be one-- and preferably only one --obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- ...
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.



Python terminology

- **Benevolent Dictator For Life:** Guido van Rossum, creator of Python
- **Pythonista:** a python user
- **Pythoneer:** one who has mastered the Zen of Python
- **Pythonic:** code that is written in the spirit of the Zen of Python



What you really need to know

- Python code is organised as:
 - Package
 - Module(s)
 - Functions
 - Objects
- Zero-based indexing
- White space matters



Code example



A screenshot of a code editor window titled 'test.py'. The editor shows a Python function definition. The code is as follows:

```
1 def this_is_python(spam="spam"):  
2     # this is where you write code  
3     with_eggs = spam + " with eggs"  
4     return with_eggs  
5 |
```

The code is color-coded: 'def' is purple, 'this_is_python' is blue, 'spam="spam"' is blue, ':' is black, '#' is green, 'this is where you write code' is green, 'with_eggs = spam + " with eggs"' is black, 'return' is purple, and 'with_eggs' is black. The cursor is at the end of line 5.





How does it work

How does it work

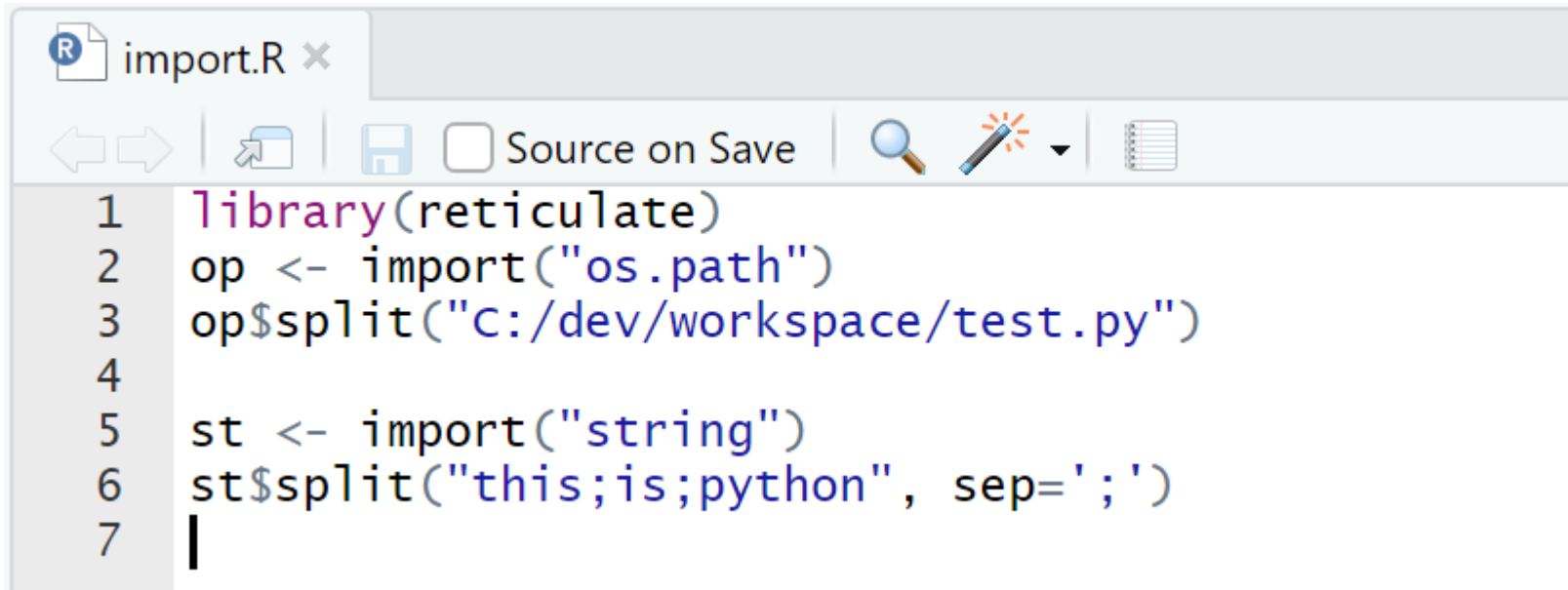
using python modules

- Use `import (<module name>)`
- Returned object of class “module” exposes functionality
- Like any R class access functionality through ‘\$’



How does it work

using python modules



The image shows a screenshot of an RStudio editor window. The title bar at the top reads 'import.R x'. Below the title bar is a toolbar with icons for navigation (back, forward), file operations (open, save), and a checkbox labeled 'Source on Save'. The main editor area contains the following R code, which uses the 'reticulate' package to import Python modules:

```
1 library(reticulate)
2 op <- import("os.path")
3 op$split("C:/dev/workspace/test.py")
4
5 st <- import("string")
6 st$split("this;is;python", sep=';')
7 |
```



How does it work

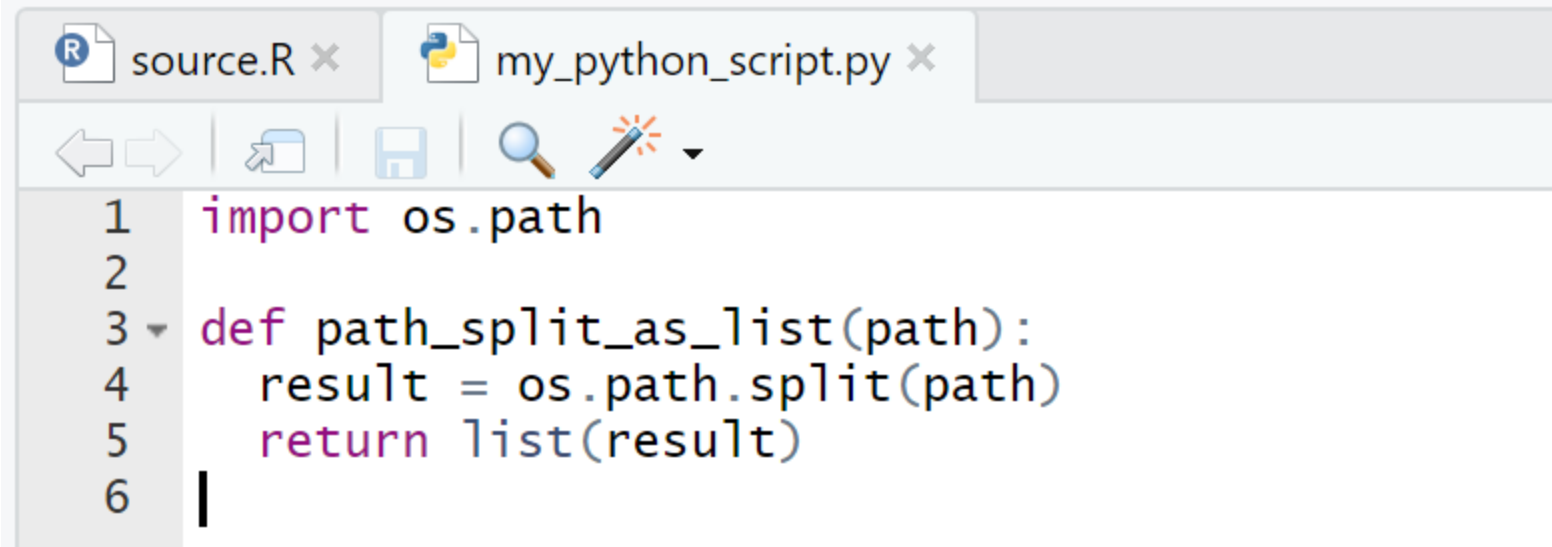
using your own python code

- Use `source_python(<python script>)` to run python code
- All created python objects are loaded into the R environment



How does it work

using your own python code



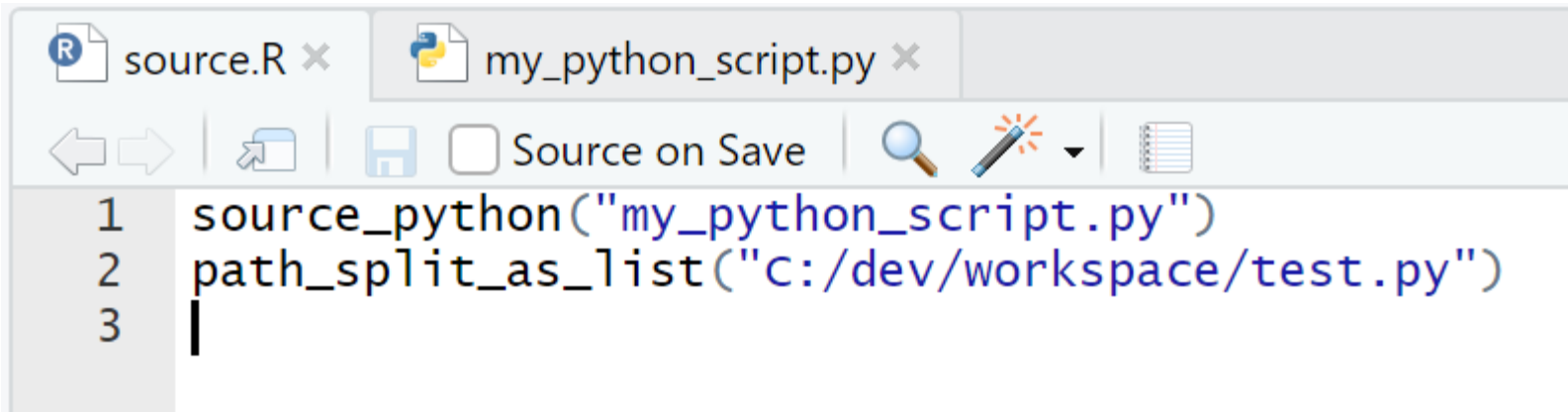
The image shows a screenshot of a code editor interface. At the top, there are two tabs: 'source.R' with an R logo and 'my_python_script.py' with a Python logo. Below the tabs is a toolbar with icons for navigation (back, forward), file operations (open, save), search (magnifying glass), and a drawing tool (pencil). The main area displays a Python script with the following code:

```
1 import os.path
2
3 def path_split_as_list(path):
4     result = os.path.split(path)
5     return list(result)
6 |
```



How does it work

using your own python code



The screenshot shows the RStudio interface with a script editor open. The tab is labeled 'my_python_script.py'. The script contains three lines of R code:

```
1 source_python("my_python_script.py")
2 path_split_as_list("c:/dev/workspace/test.py")
3 |
```



The screenshot shows the RStudio console with the following output:

```
Console C:/dev/workspace/
> source_python("my_python_script.py")
> path_split_as_list("c:/dev/workspace/test.py")
[1] "c:/dev/workspace" "test.py"
> |
```



How does it work

a more complex example

- R has `base::duplicated`, Python has `pandas.drop_duplicates`
- <http://wesmckinney.com/blog/filtering-out-duplicate-dataframe-rows/>
- Let's reproduce research!



How does it work

a more complex example

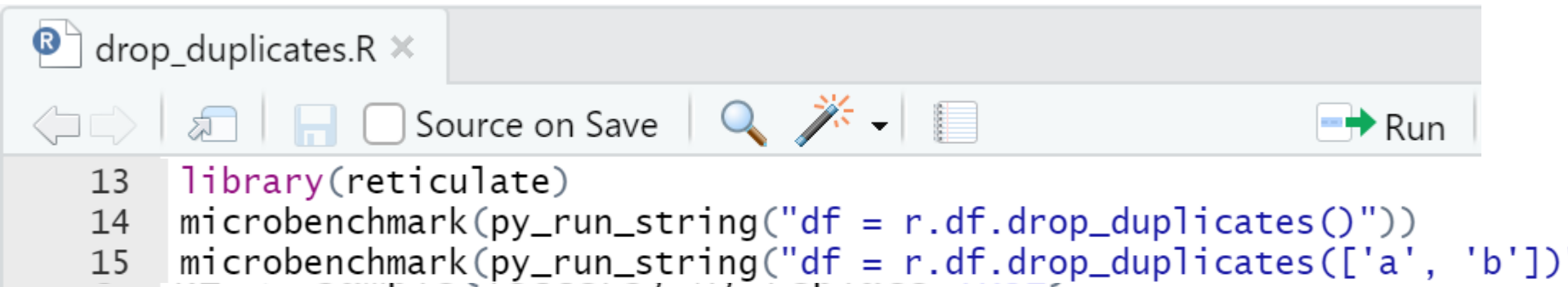
```
drop_duplicates.R x
← → | 📁 | 💾 ☐ Source on Save | 🔍 ✏️ ▼ | 📝 | 🏠 ➡ Run

1 set.seed(1234)
2 N <- 100000
3 k1 <- sample(letters, N, replace=TRUE)
4 k2 <- sample(letters, N, replace=TRUE)
5 df <- data.frame(a=k1, b=k2, c=rep(1:100, N / 100),
6                   stringsAsFactors = FALSE)
7
8 library(microbenchmark)
9 microbenchmark(df[!duplicated(df)],)
10 microbenchmark(df[!duplicated(df[,c("a", "b")])],)
11
12
```



How does it work

a more complex example



```
drop_duplicates.R x
Source on Save
Run

13 library(reticulate)
14 microbenchmark(py_run_string("df = r.df.drop_duplicates()"))
15 microbenchmark(py_run_string("df = r.df.drop_duplicates(['a', 'b'])"))
```

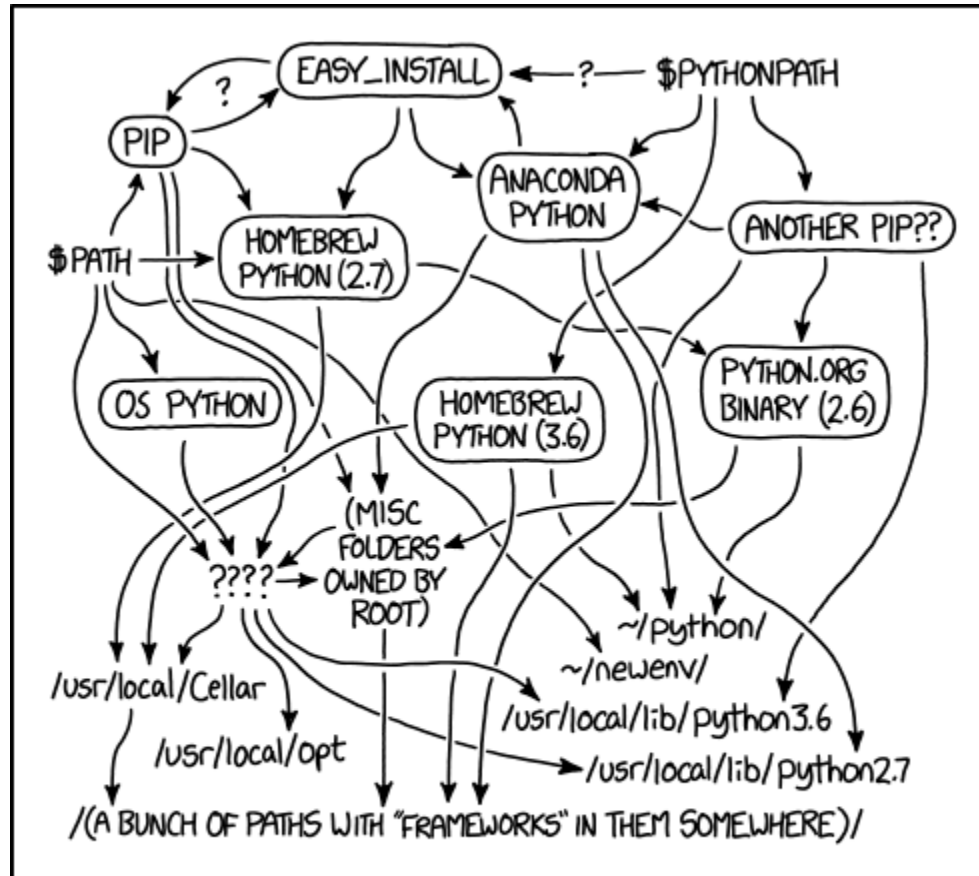
- Automatic conversion from `data.frame` to `pandas.DataFrame` (more later)
- Access Python/R objects with **r** and **py**





Python, I choose you!

Python Version Discovery



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED
THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

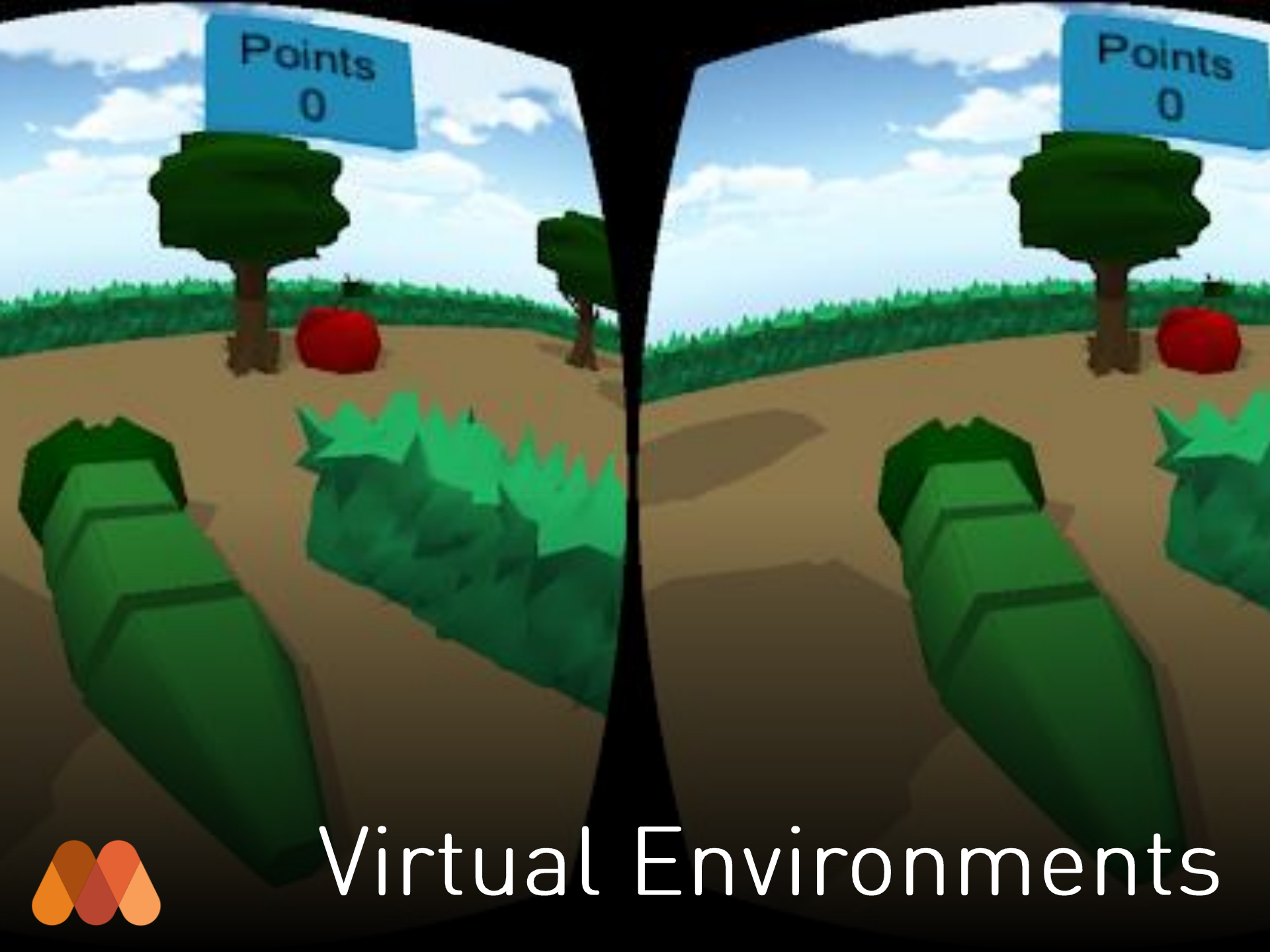
<https://xkcd.com/1987/>



The reticulated answer

- Activated by first call to python and then whichever version has the module
- In this order:
 1. RETICULATE_PYTHON environment var.
 2. User specified (through `use_python`)
 3. Virtual envs with imported module name
 4. As specified on PATH
 5. Other (e.g. `/usr/local/bin/python`)





Virtual Environments

Virtual Environments

- Isolated python environment
- Different setups for different projects
- Easy to create, maintain, throw away
- Two flavours:
 - `virtualenv` (standard library)
 - `conda` (new & improved)



Virtually reticulated

- `use_virtualenv/use_conda` for existing environments
- `virtualenv_create/conda_create` to create new ones
- Note installing packages with `py_install` installs in a virtual env “r-reticulate”





The mailman package

Rationale

- “Publish blogposts on a Tuesday” – Gabor Csardi
- Gmail clips e-mails larger than 102Kb

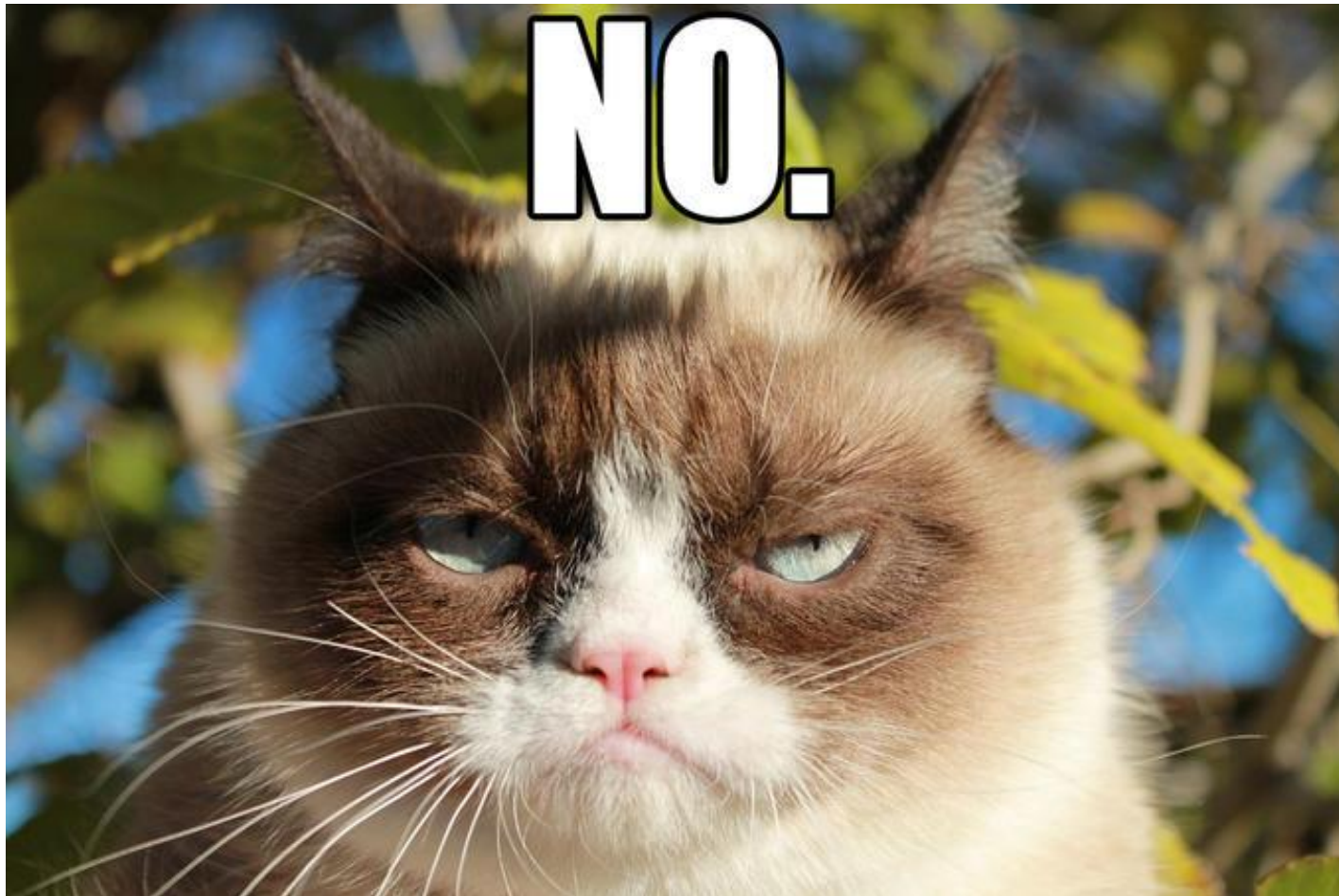


How did I get data?

- Google takeout: takeout.google.com
- Export your data from most Google services
- Gmail exported in mbox format



How to read mbox in R



Enter python



Creating a hybrid package

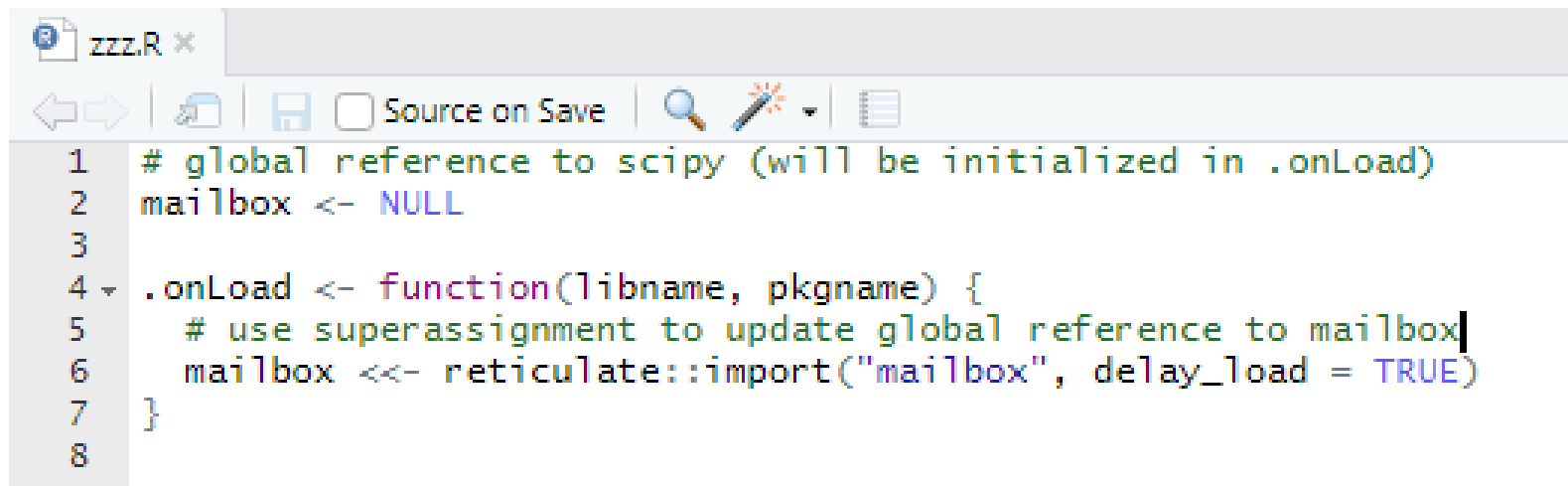
Some considerations

- Python module must be loaded
 - Implement `.onLoad()`
- But only after python version discovery
 - Use `import (<module name>, delay_load=TRUE)`



Creating a hybrid package

Some considerations



```
1 # global reference to scipy (will be initialized in .onLoad)
2 mailbox <- NULL
3
4 .onLoad <- function(libname, pkgname) {
5   # use superassignment to update global reference to mailbox
6   mailbox <<- reticulate::import("mailbox", delay_load = TRUE)
7 }
8
```



Creating a hybrid package

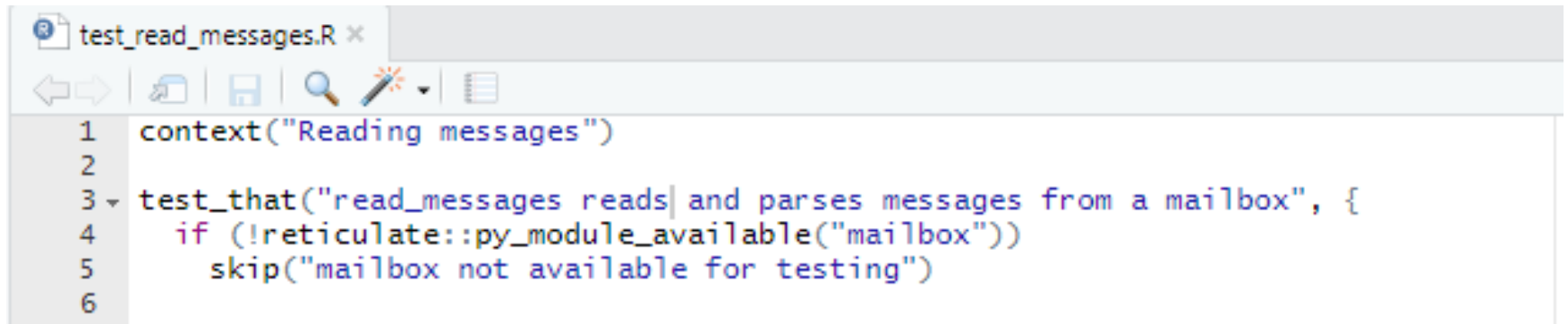
Some considerations

- Package must pass CRAN checks
 - Run `devtools::check()`
 - Run `goodpractice::gp()`
- Skip tests if no python/module present
 - Use `py_module_available(<module name>)`



Creating a hybrid package

Some considerations



```
1 context("Reading messages")
2
3 test_that("read_messages reads and parses messages from a mailbox", {
4   if (!reticulate::py_module_available("mailbox"))
5     skip("mailbox not available for testing")
6   ...
```



Creating a hybrid package

<demo>

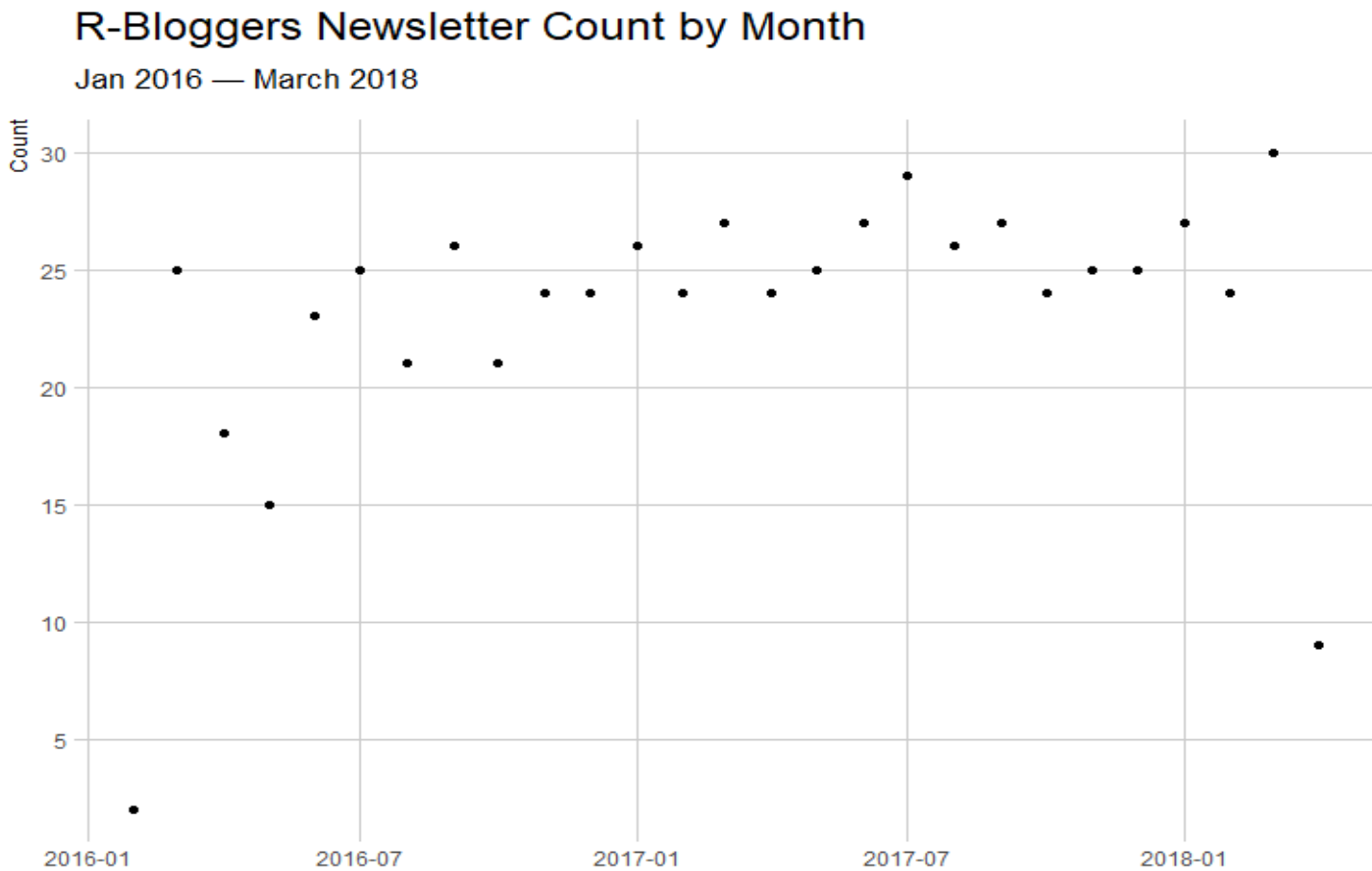


Prior Art

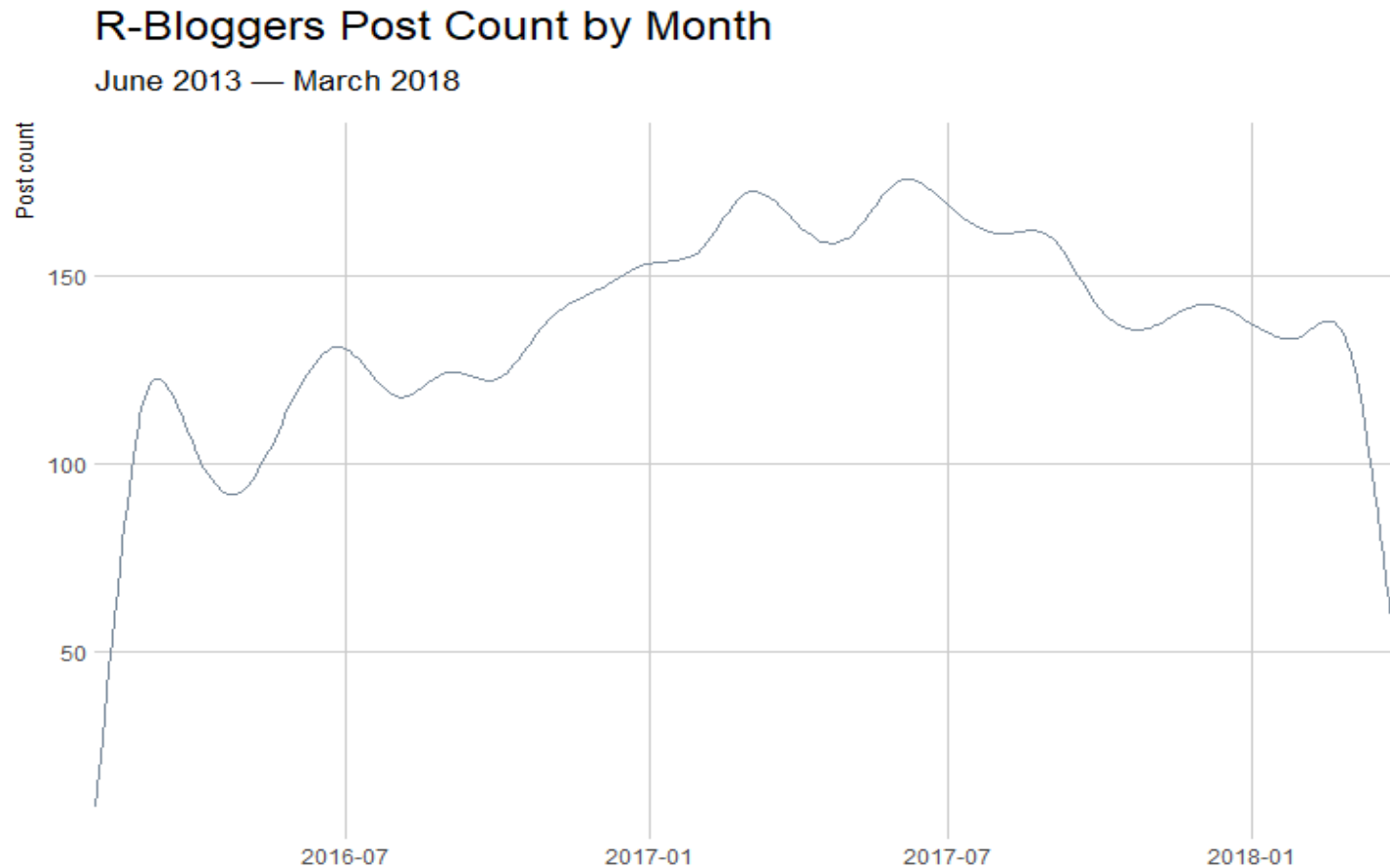
Exploring R-Bloggers Posts with the Feedly API (<https://rud.is/b/2018/04/04/exploring-r-bloggers-posts-with-the-feedly-api/>)



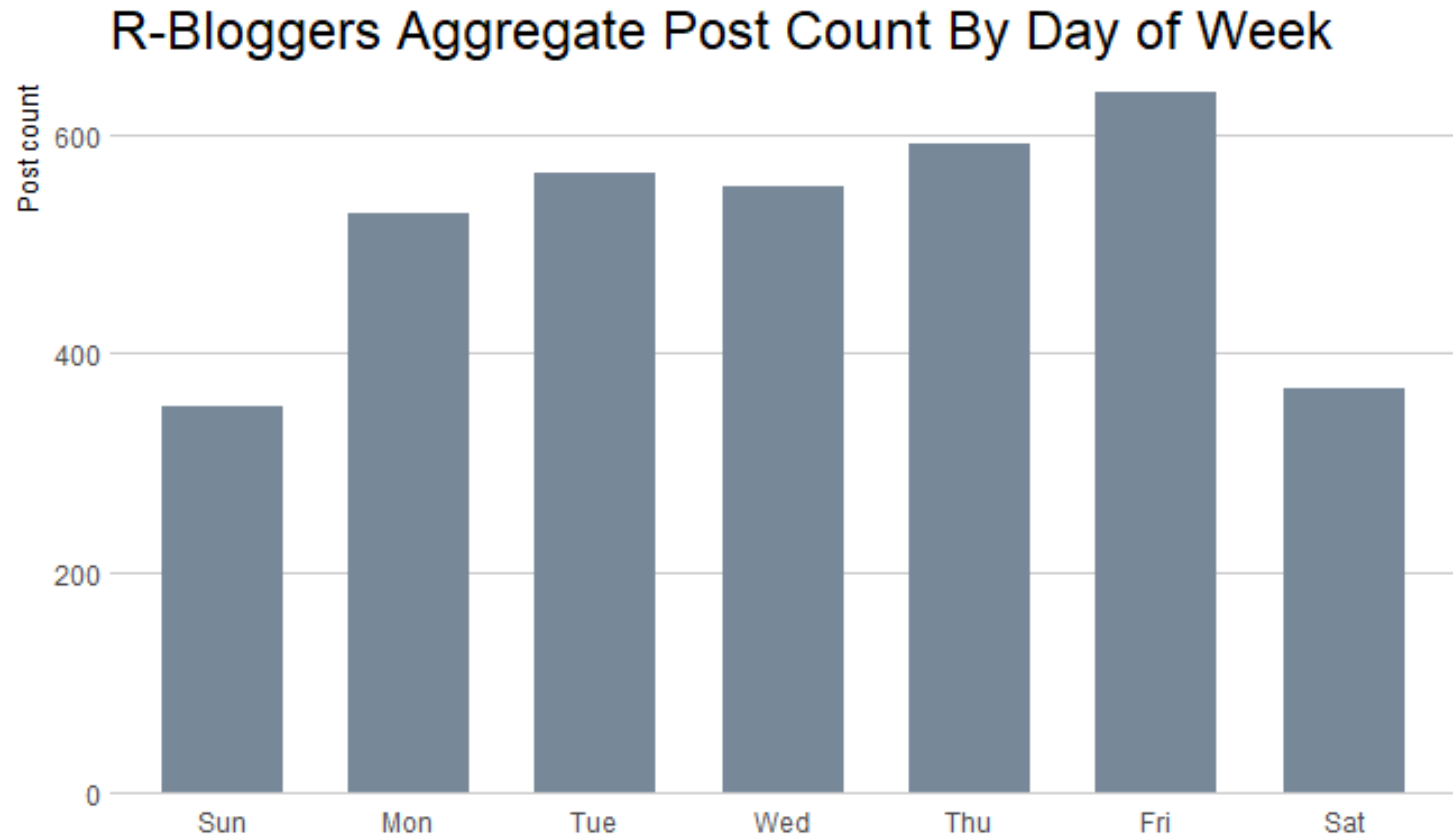
Bonus: Analysis



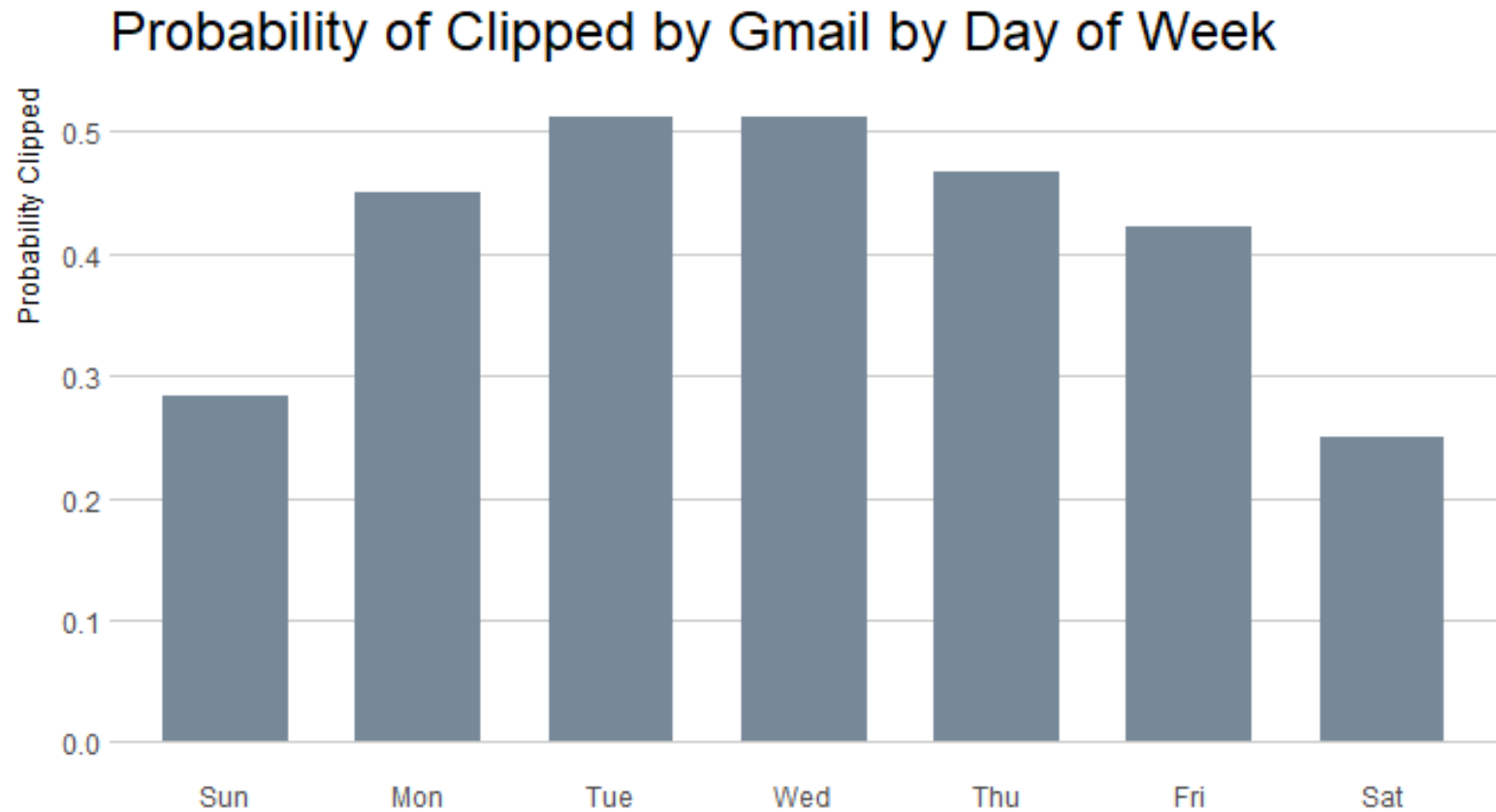
Bonus: Analysis



Bonus: Analysis



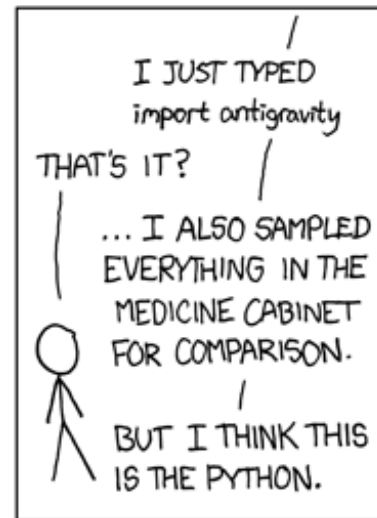
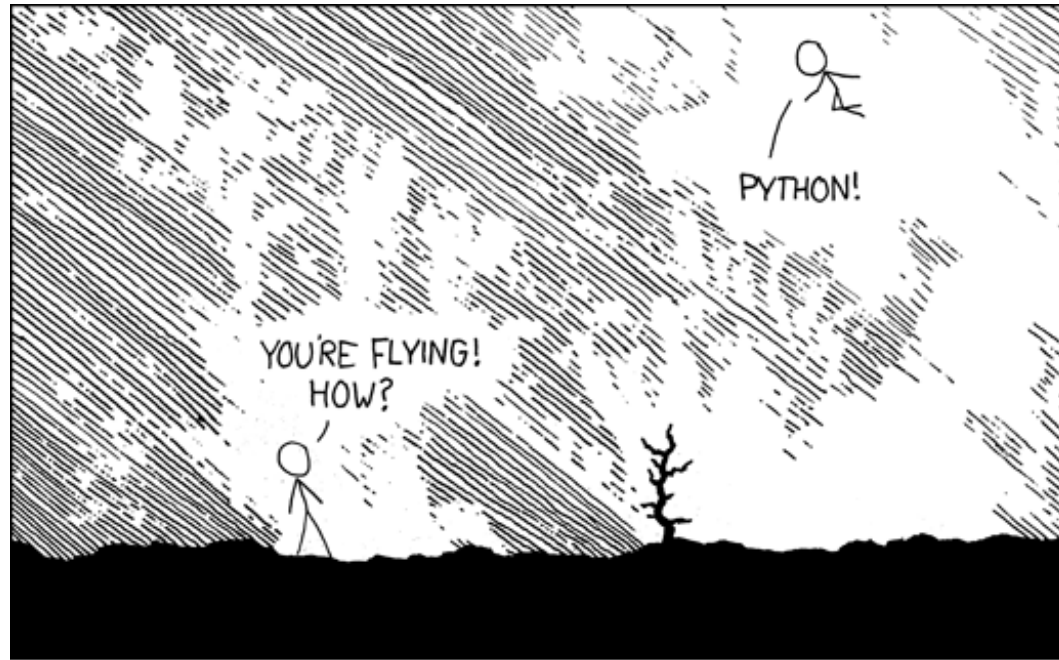
Bonus: Analysis



The answer

- Most posts occur on Friday
- Tuesday's newsletter has highest probability of being clipped
- Gabor was wrong?





<https://xkcd.com/353/>



afiaz@mango-solutions.com

Resources

- <https://rstudio.github.io/reticulate/>
- <https://github.com/MangoTheCat/mailman>
- <http://www.mango-solutions.com/>

