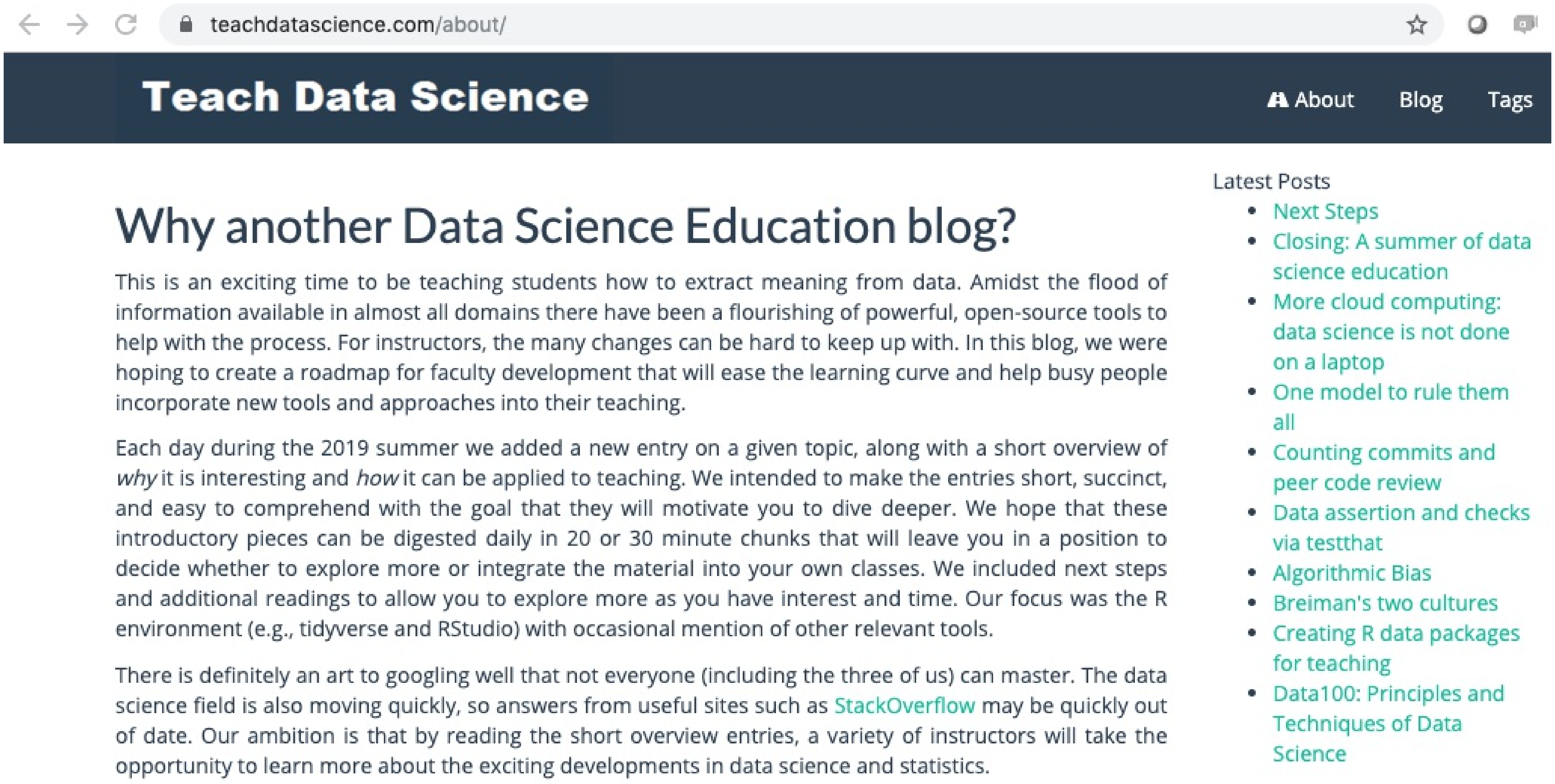
`reticulate`:

running Python within RStudio

Jo Hardin

October 14, 2019

Just say Yes! -- https://teachdatascience.com/



Why another Data Science Education blog?

This is an exciting time to be teaching students how to extract meaning from data. Amidst the flood of information available in almost all domains there have been a flourishing of powerful, open-source tools to help with the process. For instructors, the many changes can be hard to keep up with. In this blog, we were hoping to create a roadmap for faculty development that will ease the learning curve and help busy people incorporate new tools and approaches into their teaching.

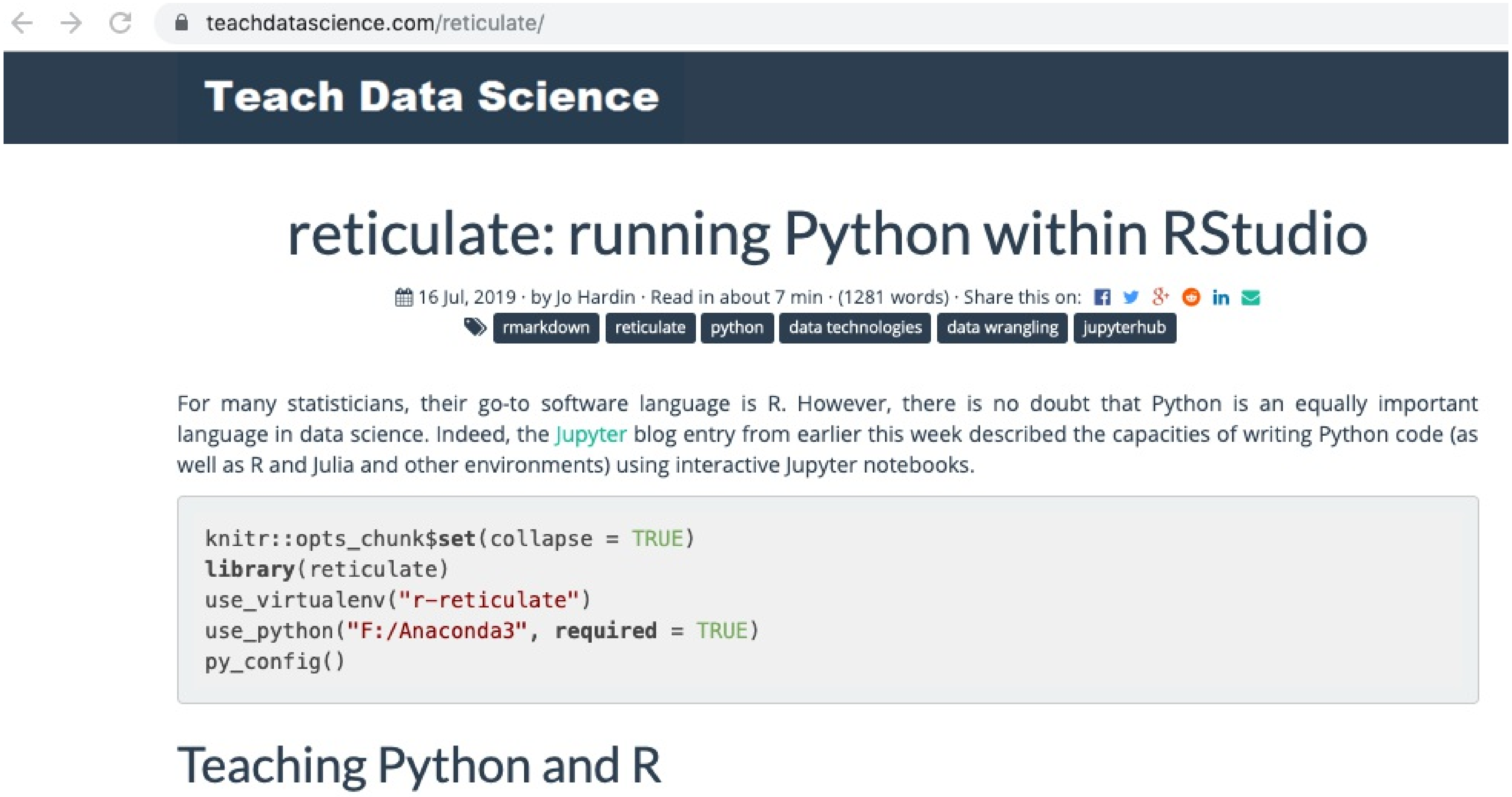
Each day during the 2019 summer we added a new entry on a given topic, along with a short overview of *why* it is interesting and *how* it can be applied to teaching. We intended to make the entries short, succinct, and easy to comprehend with the goal that they will motivate you to dive deeper. We hope that these introductory pieces can be digested daily in 20 or 30 minute chunks that will leave you in a position to decide whether to explore more or integrate the material into your own classes. We included next steps and additional readings to allow you to explore more as you have interest and time. Our focus was the R environment (e.g., tidyverse and RStudio) with occasional mention of other relevant tools.

There is definitely an art to googling well that not everyone (including the three of us) can master. The data science field is also moving quickly, so answers from useful sites such as StackOverflow may be quickly out of date. Our ambition is that by reading the short overview entries, a variety of instructors will take the opportunity to learn more about the exciting developments in data science and statistics.

Latest Posts

- Next Steps
- Closing: A summer of data science education
- More cloud computing: data science is not done on a laptop
- One model to rule them all
- Counting commits and peer code review
- Data assertion and checks via testthat
- Algorithmic Bias
- Breiman's two cultures
- Creating R data packages for teaching
- Data100: Principles and Techniques of Data Science

Teach Data Science: `reticulate`



reticulate: running Python within RStudio

16 Jul, 2019 · by Jo Hardin · Read in about 7 min · (1281 words) · Share this on:

rmarkdown reticulate python data technologies data wrangling jupyterhub

For many statisticians, their go-to software language is R. However, there is no doubt that Python is an equally important language in data science. Indeed, the Jupyter blog entry from earlier this week described the capacities of writing Python code (as well as R and Julia and other environments) using interactive Jupyter notebooks.

```
knitr::opts_chunk$set(collapse = TRUE)
library(reticulate)
use_virtualenv("r-reticulate")
use_python("F:/Anaconda3", required = TRUE)
py_config()
```

Teaching Python and R

What is `reticulate`??

(much of the talk taken from: https://rstudio.github.io/reticulate/)

R Interface to Python

The **reticulate** package provides a comprehensive set of tools for interoperability between Python and R. The package includes facilities for:

- Calling Python from R in a variety of ways including R Markdown, sourcing Python scripts, importing Python modules, and using Python interactively within an R session.
- Translation between R and Python objects (for example, between R and Pandas data frames, or between R matrices and NumPy arrays).
- Flexible binding to different versions of Python including virtual environments and Conda environments.

Reticulate embeds a Python session within your R session, enabling seamless, high-performance interoperability. If you are an R developer that uses Python for some of your work or a member of data science team that uses both languages, reticulate can dramatically streamline your workflow!

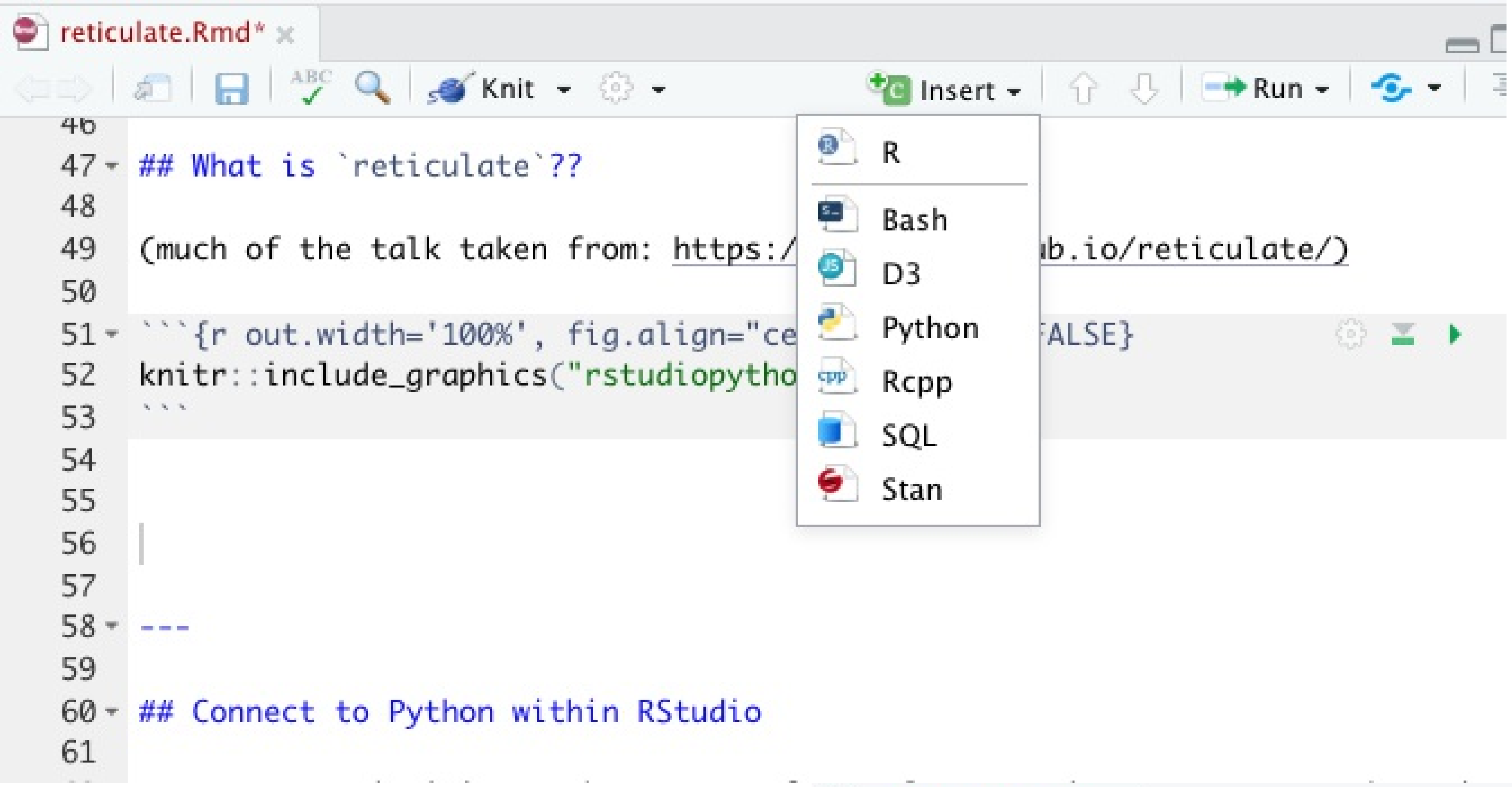
Connect to Python within RStudio

For many statisticians, the go-to software language is R. However, there is no doubt that Python is a very important language in data science. Why not do both??

```r
library(reticulate)
use_virtualenv("r-reticulate")
import("statsmodels")
```

```
## Module(statsmodels)
```

I can run Python inside R??

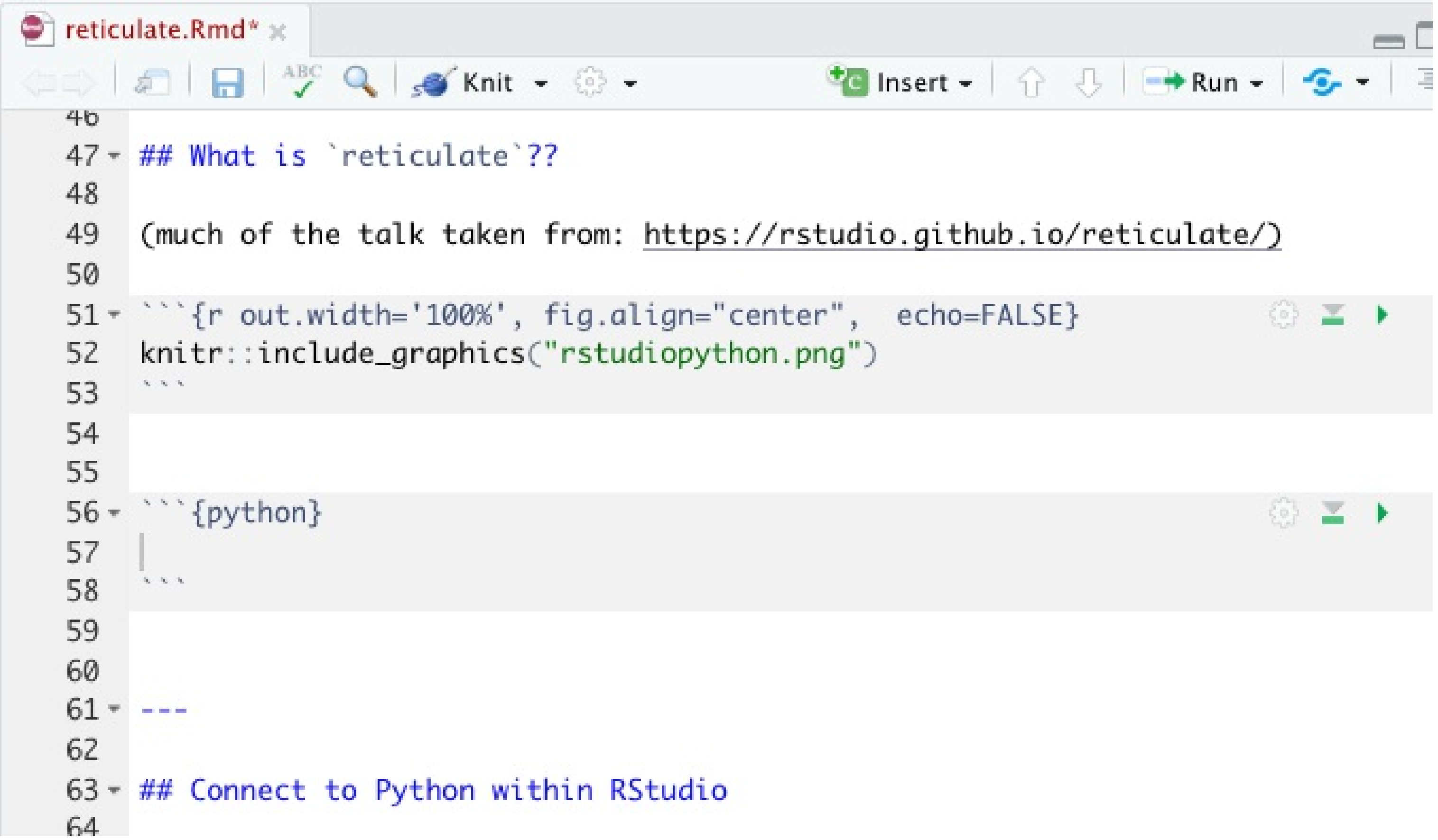


~~~
## What is `reticulate`??

(much of the talk taken from: https:/                    ub.io/reticulate/)

```{r out.width='100%', fig.align="ce                    FALSE}
knitr::include_graphics("rstudiopytho
~~~

Connect to Python within RStudio
```


reticulate.Rmd*
Knit
Insert
Run

```
## What is `reticulate`??

(much of the talk taken from: https://rstudio.github.io/reticulate/)

```{r out.width='100%', fig.align="center",  echo=FALSE}
knitr::include_graphics("rstudiopython.png")
```

```{python}

```

Connect to Python within RStudio
```
```

Python in R

- `pandas` for data wrangling.
- In R, the chunk is specified to be a Python chunk (RStudio is now running Python).

~~~
```{python}
import pandas
flights = pandas.read_csv("flights.csv")
flights = flights[flights["dest"] == "ORD"]
flights = flights[['carrier', 'dep_delay', 'arr_delay']]
flights = flights.dropna()
```
~~~

A view of the Python chunk which is actually run:

```python
import pandas
flights = pandas.read_csv("flights.csv")
flights = flights[flights["dest"] == "ORD"]
flights = flights[['carrier', 'dep_delay', 'arr_delay']]
flights = flights.dropna()
```

Learn about the dataset

~~~
```{python}
flights.shape
flights.head(3)
flights.describe()
```
~~~

```
flights.shape
```

```
## (12590, 3)
```

```
flights.head(3)
```

```
##    carrier  dep_delay  arr_delay
## 4       UA       -4.0       12.0
## 5       AA       -2.0        8.0
## 22      AA       -1.0       14.0
```

```
flights.describe()
```

```
##           dep_delay     arr_delay
## count  12590.000000  12590.000000
## mean      11.709770      2.917951
## std       39.409704     44.885155
## min      -20.000000    -62.000000
## 25%       -6.000000    -22.000000
## 50%       -2.000000    -10.000000
## 75%        9.000000     10.000000
## max      466.000000    448.000000
```

Computations

~~~
```{python}
flights = pandas.read_csv("flights.csv")
flights = flights[['carrier', 'dep_delay', 'arr_delay']]
flights.groupby("carrier").mean()
```
~~~

```python
flights = pandas.read_csv("flights.csv")
flights = flights[['carrier', 'dep_delay', 'arr_delay']]
flights.groupby("carrier").mean()
```

```
##          dep_delay  arr_delay
## carrier
## AA        8.586016   0.364291
## AS        5.804775  -9.930889
## DL        9.264505   1.644341
## UA       12.106073   3.558011
## US        3.782418   2.129595
```

From Python chunk to R chunk

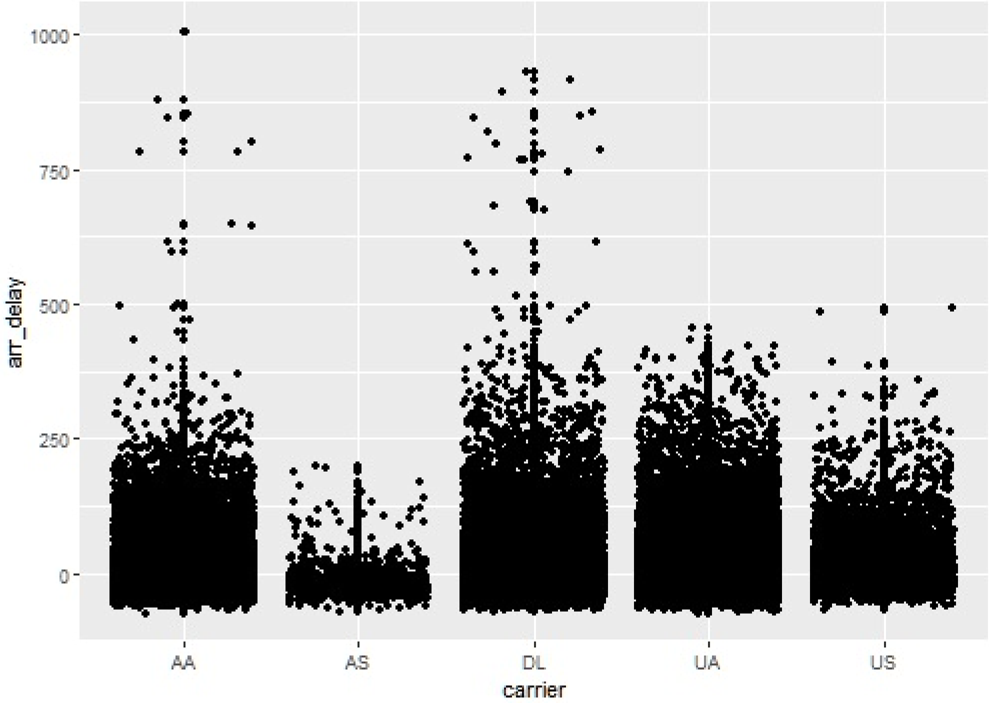
- `py$x` accesses an `x` variable created within Python from R
- `r.x` accesses an `x` variable created within R from Python

~~~
```{r}
library(ggplot2)
ggplot(py$flights, aes(x=carrier, y = arr_delay)) +
  geom_point() + geom_jitter()
```
~~~

From Python chunk to R chunk

- `py$x` accesses an `x` variable created within Python from R
- `r.x` accesses an `x` variable created within R from Python

```
library(ggplot2)
ggplot(py$flights,
       aes(x=carrier,
           y=arr_delay)) +
  geom_point() +
  geom_jitter()
```



From R chunk to Python chunk

```
data(diamonds)
head(diamonds)
```

```
## # A tibble: 6 x 10
##   carat cut       color clarity depth table price     x     y     z
##   <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl>
## 1 0.23  Ideal     E     SI2      61.5    55   326  3.95  3.98  2.43
## 2 0.21  Premium   E     SI1      59.8    61   326  3.89  3.84  2.31
## 3 0.23  Good      E     VS1      56.9    65   327  4.05  4.07  2.31
## 4 0.290 Premium   I     VS2      62.4    58   334  4.2   4.23  2.63
## 5 0.31  Good      J     SI2      63.3    58   335  4.34  4.35  2.75
## 6 0.24  Very Good J     VVS2     62.8    57   336  3.94  3.96  2.48
```

A Python chunk

```
print(r.diamonds.describe())
```

```
##               carat         depth  ...             y             z
## count  53940.000000  53940.000000  ...  53940.000000  53940.000000
## mean       0.797940     61.749405  ...      5.734526      3.538734
## std        0.474011      1.432621  ...      1.142135      0.705699
## min        0.200000     43.000000  ...      0.000000      0.000000
## 25%        0.400000     61.000000  ...      4.720000      2.910000
## 50%        0.700000     61.800000  ...      5.710000      3.530000
## 75%        1.040000     62.500000  ...      6.540000      4.040000
## max        5.010000     79.000000  ...     58.900000     31.800000
##
## [8 rows x 7 columns]
```

A Python chunk

```
import statsmodels.formula.api as smf
model = smf.ols('price ~ carat', data = r.diamonds).fit()
```

```
## C:\PROGRA~3\ANACON~1\lib\site-packages\statsmodels\compat\pandas.py:49: FutureWarning: The Panel class is removed from pa
##   data_klasses = (pandas.Series, pandas.DataFrame, pandas.Panel)
```

```
print(model.summary())
```

```
##                             OLS Regression Results
## ==============================================================================
## Dep. Variable:                  price   R-squared:                       0.849
## Model:                            OLS   Adj. R-squared:                  0.849
## Method:                 Least Squares   F-statistic:                 3.041e+05
## Date:                Mon, 14 Oct 2019   Prob (F-statistic):               0.00
## Time:                        10:14:32   Log-Likelihood:            -4.7273e+05
## No. Observations:               53940   AIC:                         9.455e+05
## Df Residuals:                   53938   BIC:                         9.455e+05
## Df Model:                           1
## Covariance Type:            nonrobust
## ==============================================================================
##                  coef    std err          t      P>|t|      [0.025      0.975]
## ------------------------------------------------------------------------------
## Intercept  -2256.3606     13.055   -172.830      0.000   -2281.949   -2230.772
## carat       7756.4256     14.067    551.408      0.000    7728.855    7783.996
## ==============================================================================
## Omnibus:                    14025.341   Durbin-Watson:                   0.986
## Prob(Omnibus):                  0.000   Jarque-Bera (JB):           153030.525
## Skew:                           0.939   Prob(JB):                         0.00
## Kurtosis:                      11.035   Cond. No.                         3.65
## ==============================================================================
##
## Warnings:
## [1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Bells and whistles

- autocomplete
- Python script

Importing Python modules

`import()` will import any Python module and call it from R. [The `os` module provides functionality for navigating the operating system.]

```
os <- import("os")
os$listdir(".")
```

```
##  [1] "custom.css"            "flights.csv"          
##  [3] "pychunk1.png"          "pychunk2.png"         
##  [5] "PyScript.png"          "reticulate.html"      
##  [7] "reticulate.Rmd"        "reticulatePyScript.py"
##  [9] "reticulate_files"      "rstudiopython.png"    
## [11] "ST47Scopy.png"         "teachDS.png"          
## [13] "teachreticulate.png"
```

Full disclosure

- Python versions (@#$%#$%@#$????)
- module versions (@%#$%@#$%#$????)

Learn more

- RStudio R Interface to Python

https://rstudio.github.io/reticulate/

- RStudio blog on Reticulated Python

https://blog.rstudio.com/2018/10/09/rstudio-1-2-preview-reticulated-python

Thank you!

- jo.hardin@pomona.edu
- @jo_hardin47
- https://github.com/hardin47
- http://research.pomona.edu/johardin/