Python with ggplot

Jupyter Notebooks combining Python and R

Cristian Cardellino Follow

Dec 3, 2018 · 4 min read

Disclaimer: This post assumes you have some familiarity with **ggplot2** (and, of course, Python, R, and Jupyter). If you need a quick catch up with the ggplot2 library I recommend **ZevRoss cheatsheet**.

A little back story (feel free to skip ahead)

I love Python, it's my language of choice when I am working in different data science projects.

That said, in the last few weeks I have been trying to introduce myself into the world of R. Not because I think is better for data science, nor I think it can replace Python in any way, but mostly because I usually get bored when I am working too much in the same programming language, and I like to explore my options to *keep it fresh,* so to speak. For that I have started to use more and more **RStudio** and try to develop reports on **RMarkdown** for work.

However, this is not my first introduction to R, during my PhD. I had the opportunity to use it, specially in the first couple of years, but I never developed a real reason to use it, except for **ggplot**.

You see, though I did almost all the experiments of my thesis in Python, I didn't use **matplotlib** or even **seaborn**, cause I learned how to plot in R before how to do so in Python. First it was the native R plotting functions but then I gradually moved to ggplot and never looked back.

However, until last year, I was kind of struggling since I needed to switch between my **Jupyter** notebooks (both in Python and R) while I experimented, calculated the metrics and finally plotting them. I know there is a Python implementation of ggplot by **yhat**, still I prefer the R implementation (feels more natural to me).

Last year, at the **ESSLLI 2017**, there was a course on "Unsupervised methods for linguistic data" give by **Aaron Steven White**. Sadly, the material of the course was taken out from GitHub (Aaron's website as well), and I wasn't able to download all of it. But if there is something that I learned there and didn't lose was how to make Python and R (specially ggplot) co-exist on the same notebook, which is what I want to show in this post.

Setting up the environment

First we will setup the environment. For this we need Python 3 and **conda** (you can either go with **miniconda** or **Anaconda**). We need to install a couple of libraries, we will create a conda environment for that:

```
$ conda create --name python-r python=3 anaconda tzlocal
$ conda activate python-r
$ conda install -c r r-tidyverse rpy2
```

So, line by line:

- The first line creates the Python 3 environment for conda and installs the `anaconda` packages and the `tzlocal` library as well (needed by `rpy2`).
- The second loads the recently created environment.
- The third line installs the necessary packages for R: `r-tidyverse` installs the **tidyverse** packages and `rpy2` installs the library that is in charge of all the magic to work.

Lock and load (or how to use R and Python in the same notebook)

Note: The notebook that accompanies this article is available on **GitHub**. Feel free to browse it.

The examples I use are based on the **titanic dataset** (which is available in the same repository of the code). The first part of the notebooks import the needed libraries:

```
import pandas as pd
import rpy2
```

```
# the base of rpy2 plotting is matplotlib, thus we need to
declare
# it inline in order to see the plots in the notebook
%matplotlib inline
```

```
# we need to activate the automatic conversion for pandas
from rpy2.robjects import pandas2ri
pandas2ri.activate()


# load the needed extension for the %%R cell magic
%load_ext rpy2.ipython
```

There's nothing too difficult in these lines. We import `pandas` and `rpy2`. Then we activate `matplotlib` to see the plots inline (since `rpy2` uses it under the hood for plotting). Third, we need to activate the automatic conversion for pandas (this has been recently added to `rpy2`), and finally we load the extension that gives us the `%%R` cell magic.

The next cell is also pretty self explanatory, just loading the ggplot library to R:

```
%%R
```

```
# load the ggplot2 library
library(ggplot2)
```

Here the `%%R` cell magic needs to be the first line of the cell so Jupyter knows how to interpret the code that follows. The `%%R` cell magic has some optional arguments:

- `-i <variable>` is the variable from Python that will be imported to R. In the case of the 4th cell is the `titanic` pandas data frame that is imported into R as a dataframe.
- `-o <variable>` is the output varible that goes from R into Python. That variable is a Vector, and even if it is one value only (like the case of the 4th cell) needs to be accessed through indexation (like in the 5th cell).
- `-w <num>` is used by Jupyter to set the width of the plot drawn by the R cell.
- `-h <num>` like the previous option, this is used to set the height of the plot.
- `-u <unit>` this option sets the unit for width and height (of the two previous options), e.g. `in` for inches, `cm` for centimeters, `mm` for millimeters, and `px` for pixels.

And that's it. The notebook has a couple of examples of how to use the cell magic and how it works with different ggplot graphics. Feel free to explore and work your own examples. Thanks for reading!