# Comments on Applications of R vs Python

These comments are limited to applications of R and Python to our specific assignment and are not intended to cover an overall functionality available in R and Python for the purposes of exploratory data analysis.

## Data Load

Both R and Python provide functionality for reading a comma-separated file. R provides a native method that loads data from *.csv* file into a data frame. Python does not provide methods for loading data into a data frame out of the box (instead a *.csv* file can be opened for a sequential reading). However with the use of an open source libraries *pandas* and *numpy* we can load a *.csv* file directly into a data frame in Python.

|  |  |  |
| --- | --- | --- |
|  | **R Example** | **Python Example** |
| Load data | cuny9 <- read.table  ('C:/CUNY/cuny9.csv',header=TRUE, sep=",") | import pandas as pd  import numpy as np  cuny9 = pd.read\_csv('C:/CUNY/cuny9.csv') |
| Access column x1 | cuny9[,'x1'] | cuny9.x1 |

## Basic Summary Statistics

R, as the language designed for statistical analysis, provides an extensive native set of basic summary statistics capabilities. Basic summary statistics for all columns of a data set can be generated with one command *summary()*. Python has very limited number of stat functions out of the box. In order to access more extensive functionality we need to load package *scipy* and use sub-package *stats*. Still in Python every element of summary statistics should be accessed/printed separately via procedural code.

|  |  |  |
| --- | --- | --- |
|  | **R Example** | **Python Example** |
| Descriptive Statistics | summary(cuny9) | import scipy as sp  from scipy import stats  n, min\_max, mean, var, skew, kurt = stats.describe(cuny9.y1)  print("Minimum: {0:8.6f} Maximum: {1:8.6f}".format(min\_max[0], min\_max[1]))  print("Mean: {0:8.6f}".format(mean)) |

## Basic Plots

Basic plots can be created in R using plot() function. Python does not provide similar functionality out of the box.

|  |  |  |
| --- | --- | --- |
|  | **R Example** | **Python Example** |
| Basic Plots | x1 <- cuny9[,'x1']  y1 <- cuny9[,'y1']  plot(y1~x1) | Not available |

## Advanced Plots

Advanced plotting functionality is available in both R and Python trough external packages. In R – package *ggplot2*. In Python – package *matplotlib*.

Package *ggplot2* provides two main functions:

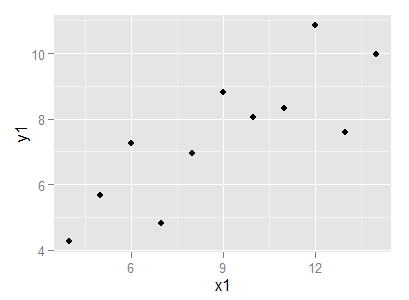
* *qplot* – for quick plots without extensive customization
* *ggplot* – allows to customize plot appearance. Function *ggplot* can be used with a number of parameters in the following categories:
  + type of plot
  + aesthetics (mapping of variables)
  + scales (mapping between data and aesthetics)
  + coordinate systems
  + faceting
  + position adjustments
  + annotation
  + themes

|  |  |  |
| --- | --- | --- |
|  | **R Example** | **Python Example** |
| Basic scatter plot | library("ggplot2")  x1 <- cuny9[,'x1']  y1 <- cuny9[,'y1']  qplot(x1,y1) | import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  %matplotlib inline  x = cuny9.x1  y = cuny9.y1  cuny9.plot(x ='x1',y ='y1',kind = 'scatter') |
| Plot with regression line | ggplot(cuny9,aes(y=y1,x=x1))+  geom\_smooth(method='lm') | m,b = np.polyfit(x,y,1)  plt.plot(x,y,'yo',x,m\*x+b, '--k')  plt.xlabel("x1")  plt.ylabel("y1") |
| Box Plot | ggplot(cuny9, aes(y=y1,x=x1))+geom\_boxplot() | fig = plt.figure()  ax = fig.add\_subplot(111)  ax.boxplot([y]) |

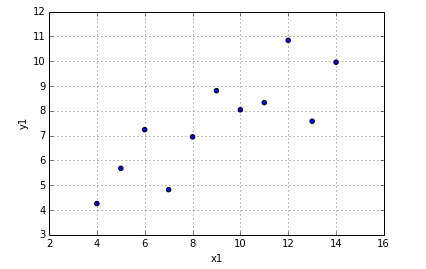
### Appearance of Graphs

If we compare similar graphs generated by ggplot2 and matplotlib, the difference is not very significant but ggplot2 graphs look more professional and ready-to-be-published look.

Graph generated by *ggplot2*:



Graph generated by *matplotlib*:



## Conclusion

As a language created specifically for statistical analysis, R outperforms Python in areas of data manipulation and exploratory data analysis. However with an introduction of *matplotlib* library Python developers received a powerful graphing tool that is comparable to R library *ggplot2* and allows Python developers to produce highly customizable graphs without leaving Python environment.