My Reticulate documentation

```r
library(reticulate)
use_python("/usr/local/bin/python3")
reticulate::py_config()
```

```
## Warning: Python '/usr/local/bin/python3' was requested but '/Users/dermit01/
## Library/r-miniconda/envs/r-reticulate/bin/python' was loaded instead (see
## reticulate::py_config() for more information)
```

```
## python:         /Users/dermit01/Library/r-miniconda/envs/r-reticulate/bin/python
## libpython:      /Users/dermit01/Library/r-miniconda/envs/r-reticulate/lib/libpython3.8.dylib
## pythonhome:     /Users/dermit01/Library/r-miniconda/envs/r-reticulate:/Users/dermit01/Library/r-minic
## version:        3.8.5 | packaged by conda-forge | (default, Sep 16 2020, 17:43:11)  [Clang 10.0.1 ]
## numpy:          /Users/dermit01/Library/r-miniconda/envs/r-reticulate/lib/python3.8/site-packages/nu
## numpy_version:  1.19.5
```

```r
# `use_virtualenv()` and `use_condaenv()`
#functions enable specification of versions of Python in virtual or conda environments, fe:
#use_virtualenv("myenv")
```

Parametric reports (parameters are passed in Reticulate.R file)

```r
print(params$text)
```

```
## [1] "Hola!"
```

Source a file to access a module

```r
#reticulate::py_run_file('/Users/dermit01/Documents/python/Chapter_5_Python_R/add.py')
source_python('/Users/dermit01/Documents/python/Chapter_5_Python_R/add.py')
add(5, 10)
```

```
## [1] 15
```

```r
suma <-add(5,10)
class(suma)
```

```
## [1] "numeric"
```

Note that once that is source into R, the new object is now an R object!

```r
py_run_string("print('hi there, printing with py_run_string')")
py_run_string("x=100")
py$x
```

```
## [1] 100
```

```r
py_run_string("dic1= {'A':1,'B':2}")
py$dic1
```

```
## $A
## [1] 1
##
## $B
## [1] 2
```

Seamless dictionary creation

Create a dictionary in a more Rish way

```
dic1 = py_dict(keys=c('A','B','C'), values=c(1,2,3))
dic1
```

```
## {'A': 1.0, 'B': 2.0, 'C': 3.0}
```

```
dic1$A
```

```
## 1.0
```

```
py_to_r(dic1)
```

```
## $A
## [1] 1
##
## $B
## [1] 2
##
## $C
## [1] 3
```

Note that the output looks like the pythonic way to represent a dictionary

```
# import numpy and specify no automatic Python to R conversion
np <- import("numpy", convert = FALSE)
#plt <- import("matplotlib.pyplot",convert = FALSE)
# do some array manipulations with NumPy
a <- np$array(c(1:4))
sum <- a$cumsum()
# convert to R explicitly at the end
py_to_r(sum)
```

```
## [1]  1  3  6 10
```

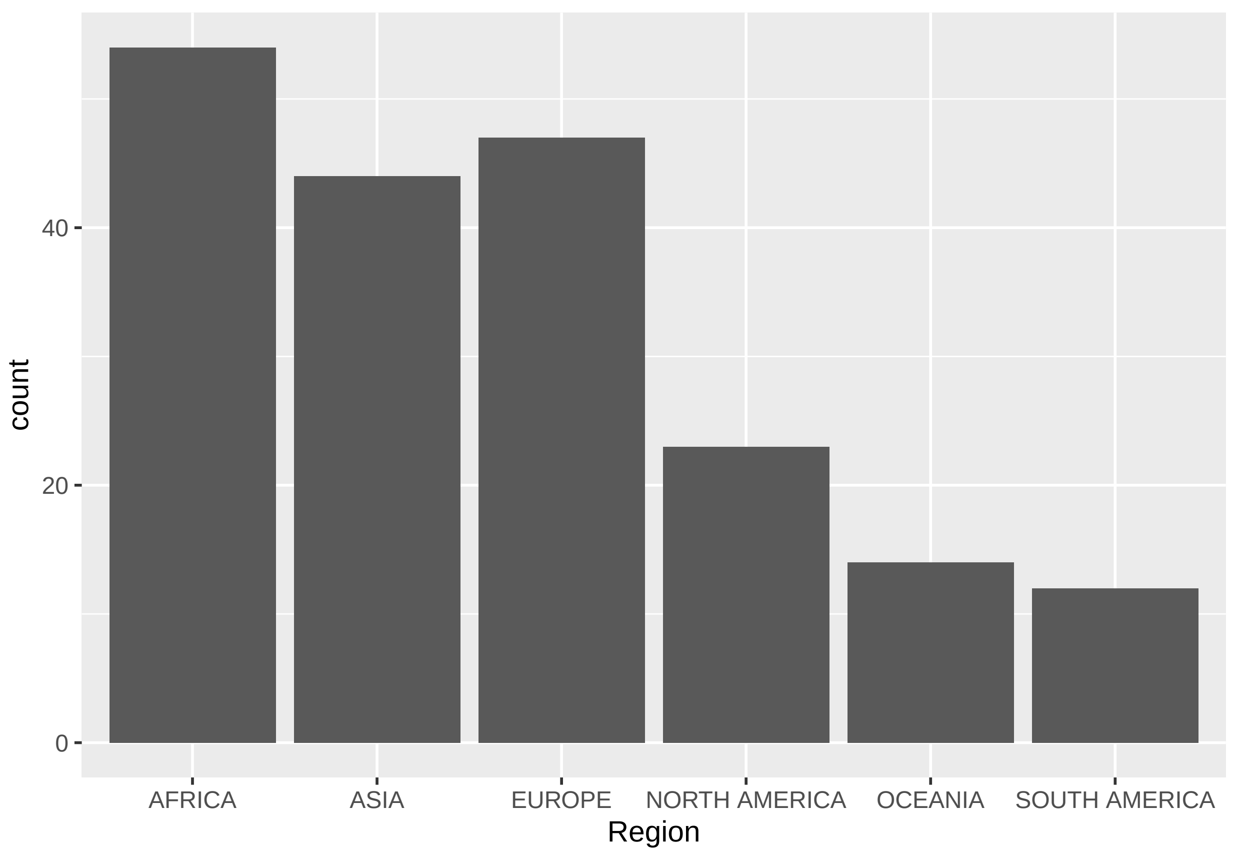
```
import pandas as pd
import io
import requests
```

```
url="https://raw.githubusercontent.com/cs109/2014_data/master/countries.csv"
s=requests.get(url).content
df=pd.read_csv(io.StringIO(s.decode('utf-8')))
print(df.head(3))
```

```
##    Country  Region
## 0  Algeria  AFRICA
## 1   Angola  AFRICA
## 2    Benin  AFRICA
```

Access the python environment with py$

```
library(ggplot2)
ggplot(py$df, aes(x=Region))+
    geom_bar()
```



Subsetting data in python

#See 43 min https://www.youtube.com/watch?v=U3ByGh8RmSc

```
#to select columns
print(df[["Region"]])
```

```
##             Region
## 0           AFRICA
## 1           AFRICA
## 2           AFRICA
## 3           AFRICA
## 4           AFRICA
## ..             ...
## 189  SOUTH AMERICA
## 190  SOUTH AMERICA
## 191  SOUTH AMERICA
## 192  SOUTH AMERICA
## 193  SOUTH AMERICA
##
```

```
## [194 rows x 1 columns]
```

Interestingly only 10 rows will be printed.

```
#to filter row
print(df[df['Region']=="AFRICA"][0:3])
```

```
##    Country  Region
## 0  Algeria  AFRICA
## 1   Angola  AFRICA
## 2    Benin  AFRICA
```

```
african_regions= df[df['Region']=="AFRICA"].shape[0]
```

Interestingly all rows are printed (I guess is to be explict?) but I can limit the number of printed statement by doing subset. Remember python starts with 0 index. We can do inline python code with ** py$**. For example the number of afircan regions is 54

For additional exaples see https://github.com/ttimbers/breast_cancer_predictor/blob/108694bc60bfb4d0be94a93b4d0a0bc1
src/reticulate_fit_breast_cancer_predict_model.R