Using Both R and Python to Manipulate Data

Kathleen Strybos

2020-05-10

Why learn how to use multiple coding languages? First of all, they offer unique benefits of their own. For instance, R is popular for statistical analysis and for creating great visualizations of data. Python is a more general language for data science and has a large user base, making it easy to get help within the community, although the R community is vast as well.

Sometimes you may want to bounce back and forth between the two when working on a project, and it is easy to work with both of the languages simultaneously using the reticulate package in R. By using this package, you are incorporating bits of each language at once.

library(reticulate)

We can then import our data using Python, with the use of the Pandas as a substitute for R’s dplyr. We can use the filter function to select which variables we want to include in a modified table and can use the sort\_values() function to arrange the observations by increasing life expectancy.

import pandas  
joindata = pandas.read\_csv("~/Desktop/Website/content/joindata.csv")  
joindatamod=(joindata.filter(['border\_status','lifeexpectancy', 'uninsuredchildrenrate'])  
 .sort\_values(by=('lifeexpectancy')).dropna())  
joindatamod

## border\_status lifeexpectancy uninsuredchildrenrate  
## 157 Non-Border 72.3 10  
## 5 Non-Border 72.4 12  
## 171 Non-Border 72.9 11  
## 202 Non-Border 73.1 9  
## 186 Non-Border 73.3 12  
## 227 Non-Border 73.3 12  
## 65 Border 73.4 9  
## 0 Non-Border 73.8 11  
## 167 Non-Border 73.9 12  
## 180 Non-Border 74.0 6  
## 145 Non-Border 74.1 11  
## 201 Non-Border 74.1 12  
## 73 Non-Border 74.1 13  
## 247 Non-Border 74.2 12  
## 137 Non-Border 74.2 15  
## 113 Non-Border 74.3 10  
## 11 Non-Border 74.3 10  
## 64 Non-Border 74.3 15  
## 209 Non-Border 74.5 15  
## 41 Non-Border 74.5 11  
## 187 Non-Border 74.5 11  
## 237 Non-Border 74.5 11  
## 96 Non-Border 74.5 15  
## 59 Non-Border 74.6 12  
## 106 Non-Border 74.6 11  
## 68 Border 74.6 17  
## 251 Non-Border 74.7 13  
## 229 Non-Border 74.7 11  
## 241 Non-Border 74.7 15  
## 116 Non-Border 74.8 12  
## .. ... ... ...  
## 47 Non-Border 79.9 12  
## 93 Non-Border 79.9 9  
## 239 Border 80.1 11  
## 238 Non-Border 80.2 12  
## 34 Non-Border 80.3 16  
## 159 Non-Border 80.3 23  
## 9 Non-Border 80.3 13  
## 104 Non-Border 80.5 9  
## 85 Non-Border 80.6 19  
## 70 Border 80.6 9  
## 179 Non-Border 80.8 9  
## 30 Border 80.8 13  
## 129 Non-Border 80.9 10  
## 102 Non-Border 81.1 17  
## 20 Non-Border 81.1 9  
## 147 Non-Border 81.4 20  
## 198 Non-Border 81.4 9  
## 135 Border 81.4 11  
## 208 Non-Border 81.6 16  
## 217 Border 81.6 13  
## 60 Non-Border 81.7 6  
## 226 Non-Border 81.8 9  
## 245 Non-Border 82.0 7  
## 107 Border 82.1 11  
## 42 Non-Border 82.7 7  
## 54 Border 82.9 15  
## 78 Non-Border 83.0 7  
## 114 Border 86.4 15  
## 188 Border 90.0 14  
## 121 Border 90.6 19  
##   
## [237 rows x 3 columns]

If we want to see how our modified dataset from Python can play with R functions, we can make a plot using that data with the ggplot function. The geom\_point() function helps us see if we can eyeball any trends in the data.

library(ggplot2)  
ggplot(py$joindatamod, aes(x=uninsuredchildrenrate, lifeexpectancy, color=border\_status))+geom\_point()+ggtitle("Life Expectancy by Rate of Uninsured Children")

As you can see, the data modifications made in Python played nicely with the functions in R. This way, we can use Python code without losing the benefits of ggplot, as we can use both simulatenously.