R Markdown Centered Data Analysis Workflow

(bonus: how to use Python in R Markdown using reticulate)

2019-10-16 (updated 2019-10-18)

Contents

1 Introduction

This document describes a data analysis project workflow that uses an R Markdown as a summary (home?) document in which codes written in R and Python are run and findings/narratives follow immediately after each code execution. For general information on R Markdown, see here. For syntax, see cheatsheet.

R Markdown can be used to generate a (portable) document that includes,

- Codes: not just of R, but also other languages as well (e.g., Python)
- Tables/plots: output from source codes
- Narratives/findings: narratives documenting quick findings

Therefore, I think it's a great way to document progress/findings of a data analysis project. (NOTE: maybe add pros and cons of using Rmd vs. Jupyter Notebook)

2 Pre-requisites

- Need to have pandoc installed.
- Need to have several latest R packages installed: e.g., rmarkdown, reticulate (to use Python)

3 Examples

For both R and Python, this document will show examples of working with

- small code snippets embedded within Rmd document
- import/source functions defined in separate scripts then use them within Rmd document
- execute/source separate script files to show outputs from corresponding scripts

In this workflow, I find it helpful to have at least 3 different vim/REPL paired sessions open:

- vim/Rmd for the source .Rmd editing and an interactive R session (let's call it session "Rmd")
- vim/R for .R script file editing and an interactive R session (let's call it session "R")
- vim/Py for .py script file editing and an interactive Python session (let's call it session "Python")

3.1 Working with R codes

```
# R set up (later, there is a separate python set up chunk)

library(ggplot2)
library(dplyr)
library(uncmbb)
library(reticulate) # for Python

# need to specify Python location if planning to use python
# note use_python is an R function, so need to be used in R chunk, not the python one
use_python(python3_path) # python3_path is defined in my .Rprofile
```

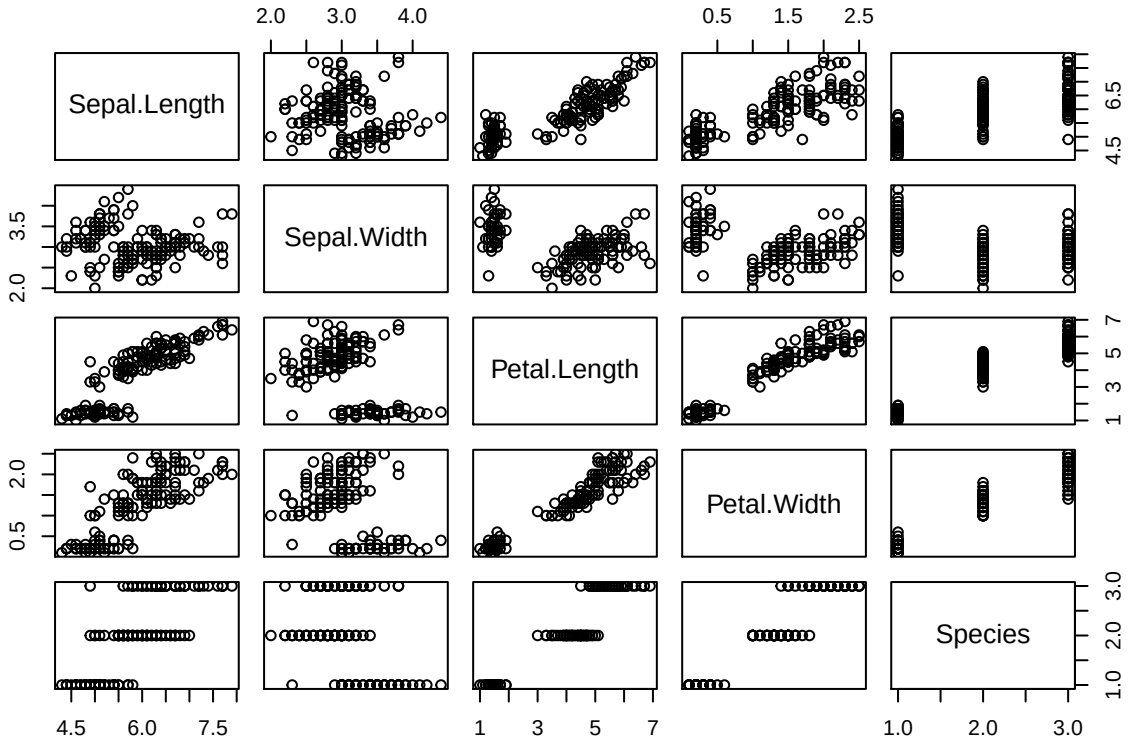
3.1.1 Running R code snippets within Rmd document directly

We can run simple R code snippets like below.

```
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
plot(iris)
```



- Note that 'print' command was not needed to actually print output of the head and plot commands

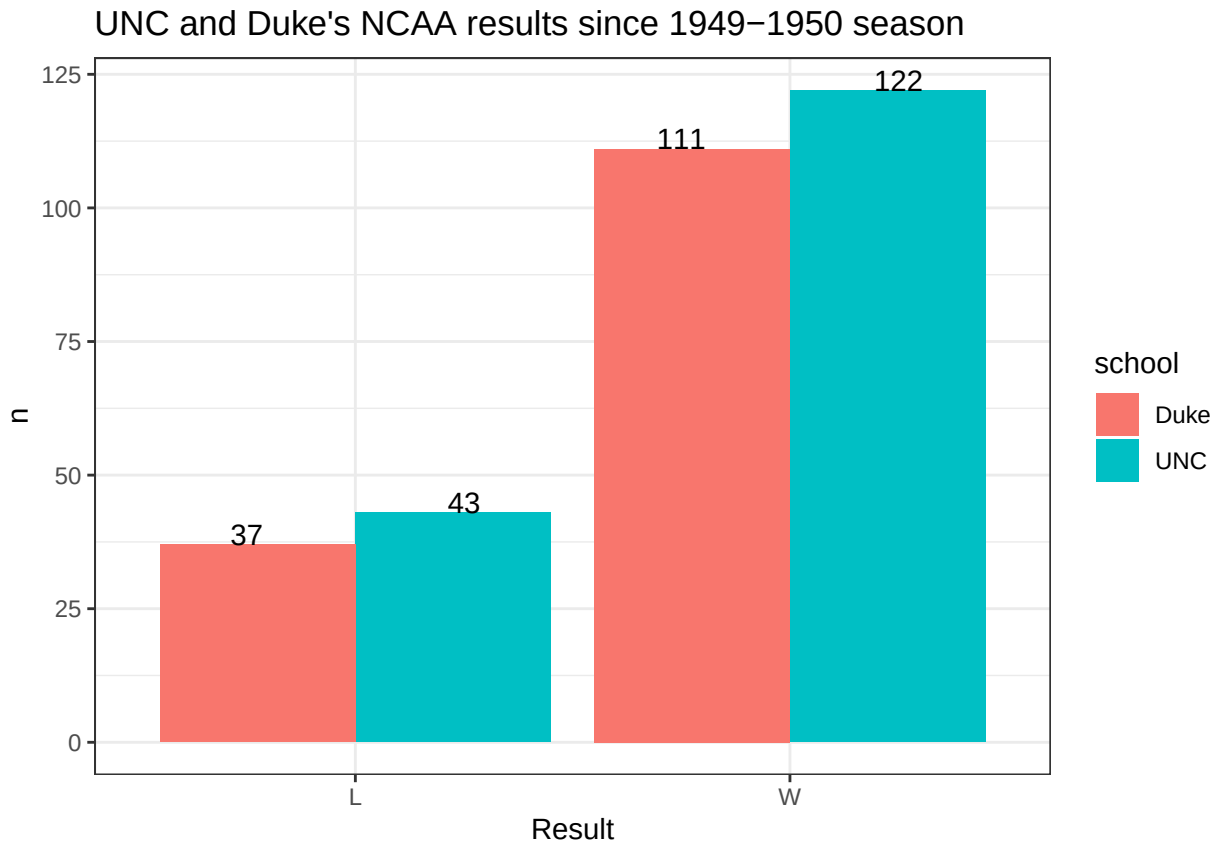
When some R packages are needed, best practice is to load them in setup chunk, generally found in the beginning of the Rmd document.

```
# Using public and personal R packages: load them in setup chunk in the beginning

df <- rbind(unc %>% mutate(school = "UNC"), duke %>% mutate(school = "Duke"))
df <- df %>% dplyr::filter(Type == "NCAA") %>%
             count(school, Result)
head(df)
```

```
## # A tibble: 4 x 3
##   school Result     n
##   <chr>  <chr>  <int>
## 1 Duke   L         37
## 2 Duke   W        111
## 3 UNC    L         43
## 4 UNC    W        122
```

```
df %>%  ggplot(aes(x = Result, y = n)) +
        geom_bar(aes(fill = school), stat = "identity", position = "dodge") +
        geom_text(aes(group = school, label = n), position = position_dodge(width = 1), vjust = 0.01) +
        labs(title = "UNC and Duke's NCAA results since 1949-1950 season") +
        theme_bw()
```



- Again, output from 'head' and 'ggplot' commands were displayed without manual printing

Objects from above R code snippets can be accessed/used throughout this Rmd document, typically when describing the findings from the snippet. E.g., below findings are calculated inline using objects from above chunk.

Since 1949-1950 season,

- UNC has played total 165 games and won 122 games, and lost 43.
- Duke has played total 148 games and won 111 games, and lost 37.

Findings follow:

- Note again that if snippets are run within Rmd, then no manual print command is needed to print plot
- The home (Rmd) document can become clogged easily

So we've seen how we can use R code snippets in Rmd document, which can be effective for small operations. For more complex operations that require a bit more lines of codes, working directly with code snippets within Rmd document may not be ideal for various reasons. In this case, we can save those (many) lines of codes in separate script files, then source them in Rmd document. First, a hybrid of the two:

3.1.2 Importing functions from R script files then use them within Rmd document

We can source .R scripts within an R chunk and use functions defined in .R scripts.

```
# hello function from R/r_udfs.R
source("R/r_udfs.R")
hello("Jay")
```

```
## [1] "Hello, Jay! 'hello' function is defined in r_udfs.R!"
```

- This allows functions to be defined in another .R script, then used in the home (Rmd) document
- The home Rmd document can become clogged easily still

3.1.3 Displaying outputs from R script files within Rmd document

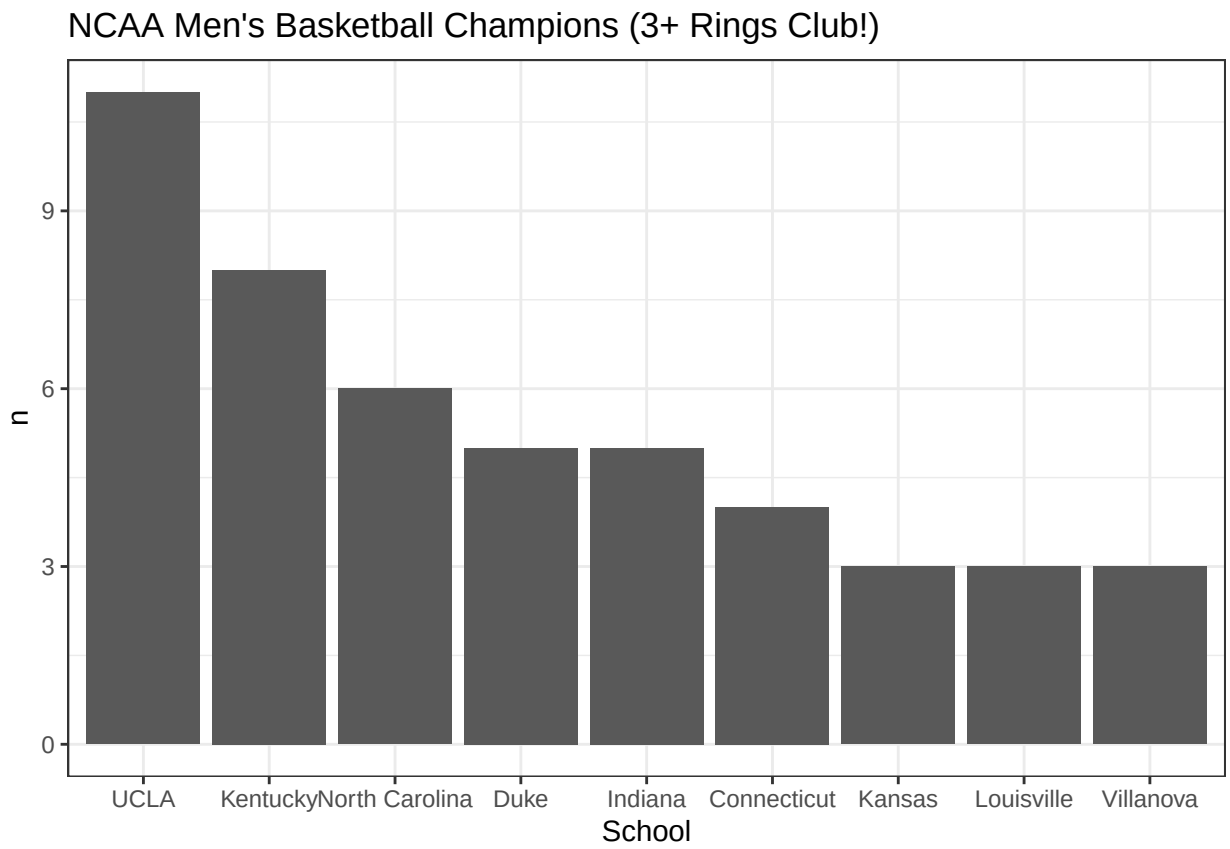
Another common (more realistic?) scenario in data analysis project is to start analysis by reading in data stored in a separate directory (e.g., data/), perform data transformation, then generate some tables/plots.

Below, we source "src_r_script.R" file in an R chunk, which reads in a data, summarizes it, and outputs a table and a plot. While writing the src_r_script.R code, I'd made sure each line works as intended from a separate R session. This ensures

- the source script works, and
- the findings/narratives can be added to home (Rmd) document without having to render the whole Rmd document

```
source("R/src_r_script.R")
```

```
## # A tibble: 14 x 3
##    CHAMPION       COACH               n
##    <fct>          <fct>           <int>
##  1 UCLA           John Wooden        10
##  2 Duke           Mike Krzyzewski     5
##  3 Kentucky       Adolph Rupp         4
##  4 Connecticut    Jim Calhoun         3
##  5 Indiana        Bob Knight          3
##  6 North Carolina Roy Williams        3
##  7 Cincinnati     Ed Jucker           2
##  8 Florida        Billy Donovan       2
##  9 Indiana        Branch McCracken    2
## 10 Louisville     Denny Crum          2
## 11 North Carolina Dean Smith          2
## 12 Oklahoma State Henry Iba           2
## 13 San Francisco  Phil Woolpert       2
## 14 Villanova      Jay Wright          2
```



Findings/narratives from the above R script follow immediately, and also more importantly without having to render the whole Rmd document, as the source R script was run and output was checked in a separate session (session "R").

- Note any objects in the src_r_script.R that I want to be displayed in the rendered document had to be saved to a variable and printed manually. Check out R/src_r_script.R file. Without such manual print command, the plot wouldn't display in rendered Rmd document.
- I can't wait to see how Duke handles coach K's retirement (when it happens)

3.2 Working with Python codes

We can also work with Python codes from within Rmd document, using reticulate package!

3.2.1 Running Python code snippets within Rmd document

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt

x1 = ['a', 'b', 'c', 'c', 'c', 'b', 'b', 'a', 'a', 'c']
x2 = range(10)
y = [True, False, False, False, True, True, True, False, True, False]
```

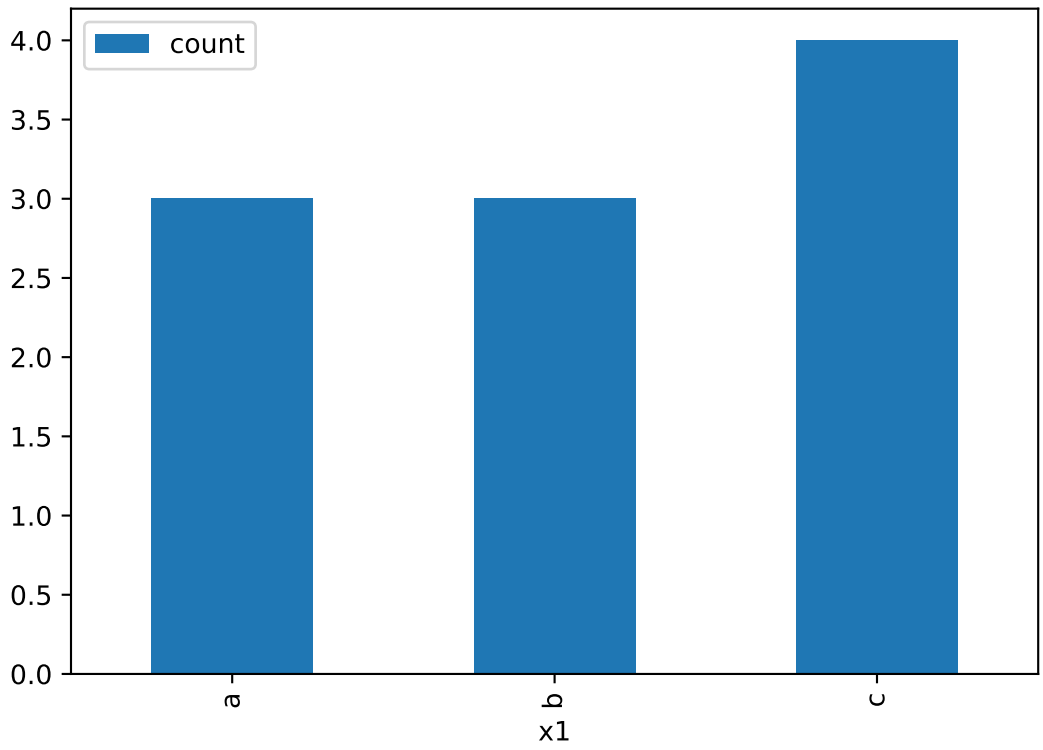
```
df = pd.DataFrame(zip(x1, x2, y), columns = ['x1', 'x2', 'y'])

# View the data frame
df.head()

# View plot
```

```
##   x1  x2      y
## 0  a   0   True
## 1  b   1  False
## 2  c   2  False
## 3  c   3  False
## 4  c   4   True
```

```
df.groupby('x1').size().reset_index(name = 'count').plot(kind = 'bar', x = 'x1', y = 'count')
plt.show()
```



Finding follows:

- A toy dataframe is hard(er) to put together in Python than in R!

We can check this Python code snippet works from a separate **Python** session.

It's preferable to first create a setup chunk specifically for Python codes, like that for R.

```
import os
import sys
import pickle
import numpy as np
import pandas as pd
import importlib
from matplotlib import pyplot as plt
plt.switch_backend('agg') # switch to 'agg' bakend, not X11, in order to disable interactive plot print
```

It seems once a python chunk is called, a separate python session is created and maintained throughout the live/active source Rmd document lifetime, therefore allowing the use of imported functions in the subsequent python chunks.

3.2.2 Importing functions from Python script files then use them within Rmd document

```
# hello function from py/py_udfs.py
from py.py_udfs import *
hello("Jay")
```

```
## Hello, Jay! 'hello' function is defined in py_udfs.py!
```

Note that we can run functions from the imported module as well, e.g., we imported all functions from py_udfs.py and used one of its functions 'hello' above.

3.2.3 Displaying outputs from Python script files within Rmd document

Again, a typical data analysis starts by reading in data from a separate directory, performs data transform, and outputs tables/plots.

First I tried using reticulate::source_python() function, but unfortunately, the outputs from the source Python script file didn't display at all in the rendered document (regardless of backends I used: 'agg' and 'x11'). I have yet to find the answer as to why.

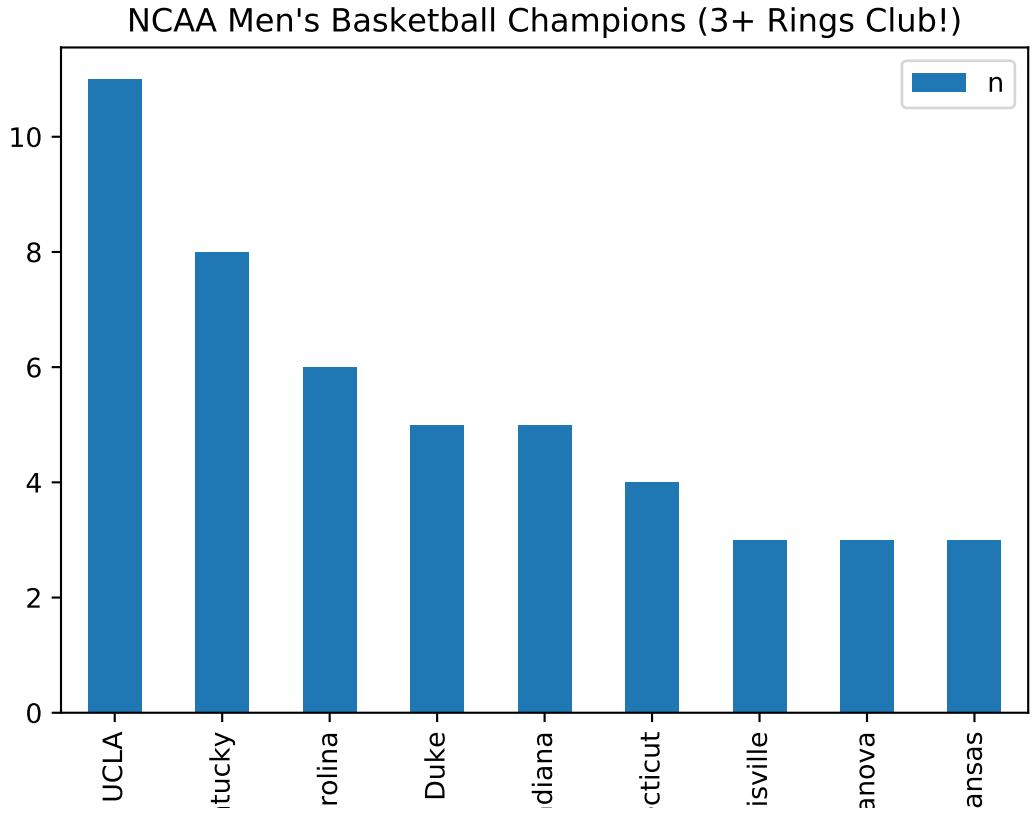
```
# note language engine here is R, not Python, although we're running Python code!
source_python("py/src_python_script.py") # without "scratch/" in filename
```

Instead, using execfile within a Python chunk prints the outputs from the Python script files.

```
# execfile seems to be the closest thing to R's source command
# execfile("py/src_python_script.py") # for Python2
exec(open("py/src_python_script.py").read()) # for Python3
```

```
##           CHAMPION             COACH   n
## 42            UCLA       John Wooden  10
## 7             Duke   Mike Krzyzewski   5
## 16        Kentucky       Adolph Rupp   4
## 32  North Carolina      Roy Williams   3
## 5      Connecticut       Jim Calhoun   3
## 11         Indiana        Bob Knight   3
```

```
## 46       Villanova        Jay Wright   2
## 30  North Carolina        Dean Smith   2
## 22      Louisville       Denny Crum   2
## 4       Cincinnati         Ed Jucker   2
## 8           Florida     Billy Donovan   2
## 38   San Francisco     Phil Woolpert   2
## 12          Indiana  Branch McCracken   2
## 36  Oklahoma State         Henry Iba   2
```



- It's fun coding in different languages!

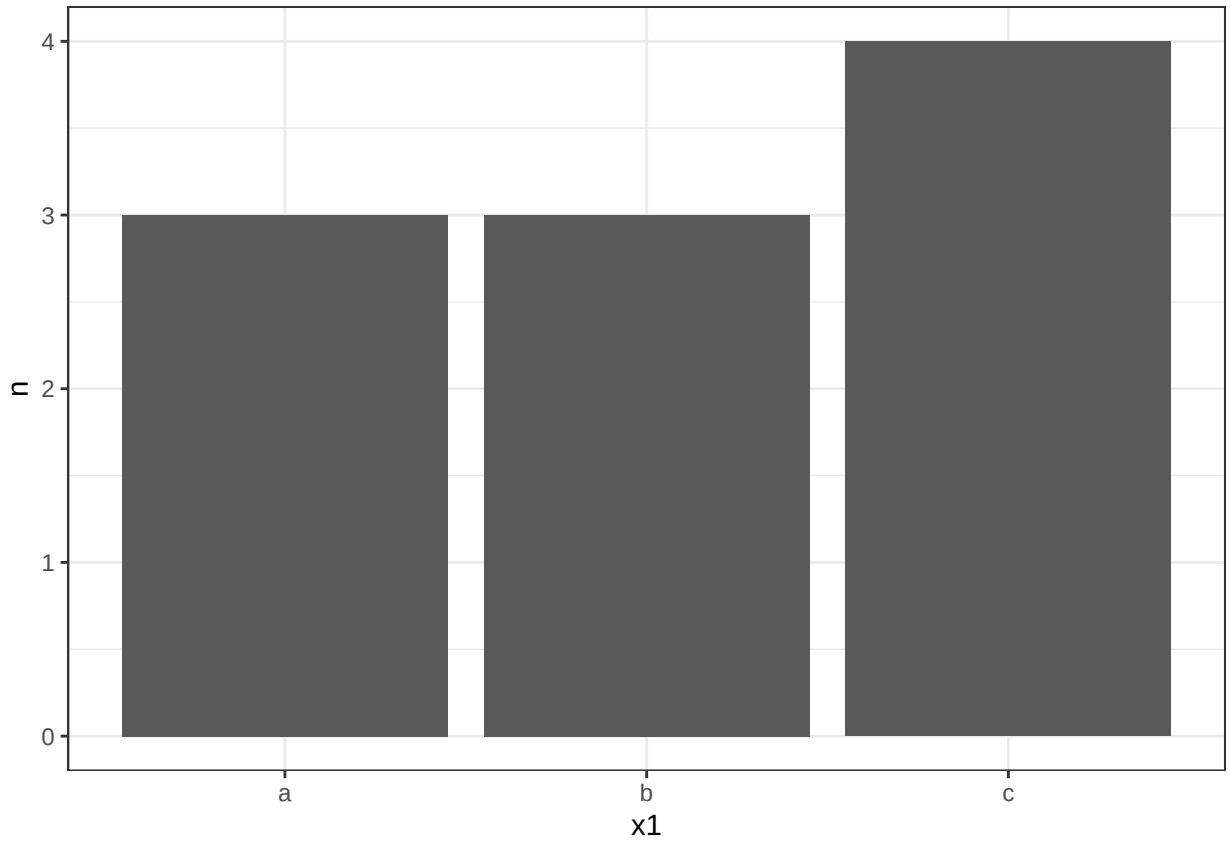
3.3 Working R and Python together

Lastly, we can also run R code snippets using Python objects that are defined earlier in R Markdown (and vice versa, i.e., run Python code snippets using R objects).

If you are more comfortable working with ggplot2 in R than say pyplot in Python, then you can use Python objects in an R chunk like below.

```
# note 'df' is a Python object defined in the earlier py_snippet_1 chunk
# and is accessible by 'py' with $
df_r = py$df
df_r %>% count(x1) %>%
         ggplot(aes(x = x1, y = n))  +
```

```
        geom_bar(stat = "identity") +
        theme_bw()
```



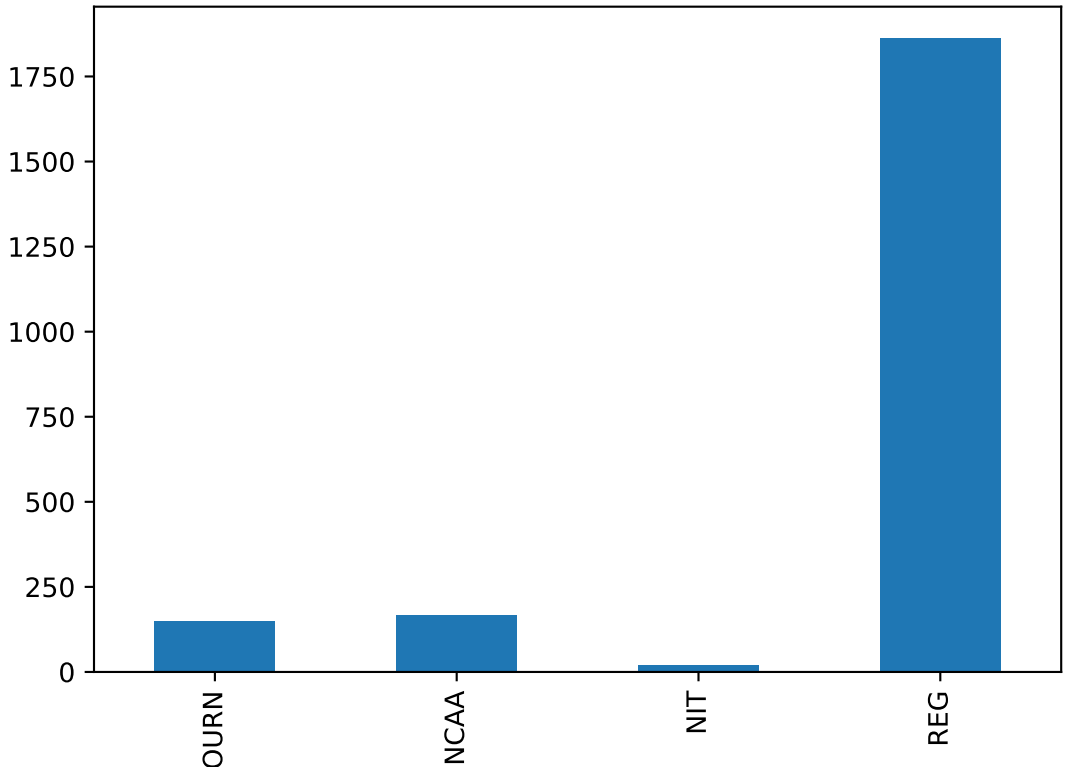
- Note how Python objects are referred to in an R chunk (by py$).

The opposite case is to use R objects in a Python chunk.

```
# note 'unc' is an R object defined in the earlier r_snippet_2 chunk
# and is accessible by 'r' object with a dot
unc_py = r.unc
unc_py.head()
```

```
##   Season   Game_Date Game_Day Type Where    Opponent_School Result    Tm   Opp  OT
## 0   1950  1949-12-01      Thu  REG     H               Elon      W  57.0  39.0  NA
## 1   1950  1949-12-03      Sat  REG     A           Richmond      W  58.0  50.0  NA
## 2   1950  1949-12-05      Mon  REG     A      Virginia Tech      L  48.0  62.0  NA
## 3   1950  1949-12-07      Wed  REG     A       Lenoir-Rhyne      L  78.0  79.0  OT
## 4   1950  1949-12-09      Fri  REG     H  George Washington      L  44.0  54.0  NA
```

```
unc_py.groupby('Type').size().plot(kind = 'bar')
plt.show()
```



Findings follow:

- Interactive checking becomes a bit hard (not sure if it's even doable), because R (Python) objects have to be read in from Python (R).
- We can start a Python interpreter using reticulate::repl_python() function from an R session (session Rmd).
- It feels like magic to me, but data conversion between R and Python seems to work just fine.

4 Report output

Once all source codes are run and findings/narratives are documented, we can go ahead and generate a summary document. Whether html or pdf, I find having an intermediate summary document of a data analysis project helps me understand the progression/findings of the project more efficiently, especially after some time off the project.

There are several options to view the output document.

1. If no file transfer is required (e.g., all files are already in local machine), open the output in a browser
2. If file transfer is needed (e.g., some files are generated in remote machine), then
    - send email from R with attachment
    - SimpleHTTPServer from Python

5 R Markdown-centered data analysis workflow

1. Start three vim/REPL paired sessions

   - session Rmd: for Rmd document (call it "home" document?) editing and interactive R session
   - session R: for R script editing and interactive R session
   - sesion py: for py script editing and interactive Python session

2. Start documenting project meta data, e.g., introduction, context, etc., in the home (Rmd) document

3. Iteratively work on codes (R/Python), generate plots, and document findings

   - Make sure codes run successfully by checking outputs, e.g., plots from terminal (ssh -YC)
   - If findings are worth noting, document them in home (Rmd) document, following corresponding code chunks

4. Render to html document and enjoy the output!