ISA 480: Data Driven Security

02 - Building your Security Analytics Toolbox

Fadel M. Megahed

Endres Associate Professor
Department of Information Systems and Analytics
Farmer School of Business
Miami University
Email: fmegahed@miamioh.edu
Office Hours: Automated Scheduler for Office Hours

Fall 2022

Outline

Learning Objectives for Today's Class

Learning Objectives

- **Understand basic syntax in both R and Python**
- **Use both platforms to solve the problem from last class**
- **Use either platform to generate reproducible reports through the R Markdown or Jupyter Notebook tools**

As always, please feel free to slow me down, stop me to ask questions, and/or ask me to explain questions differently. My main objective today is to ensure that we all have basic programming commands of both languages. If time allows throughout the semester, I would attempt to provide you with solutions to some of the problems in both programming languages.

The Beauty of Programming Languages[1]

- Programming languages are **languages**.
- **It's just text** – which gives you access to **two extremely powerful techniques**!!!
  - `Ctrl` + `C` & `Ctrl` + `V`
- In addition, programming languages are generally
  - Readable (IMO way easier than trying to figure what someone did in an)
  - Open (so you can it)
  - Reusable and reproducible
  - Diffable (version control is extremely powerful)

[0] **Source:** Content in "The Beauty of Programming Languages" is from Hadley Wickham's You Can't Do Data Science in a GUI

A Note on Both Programming Languages [1]

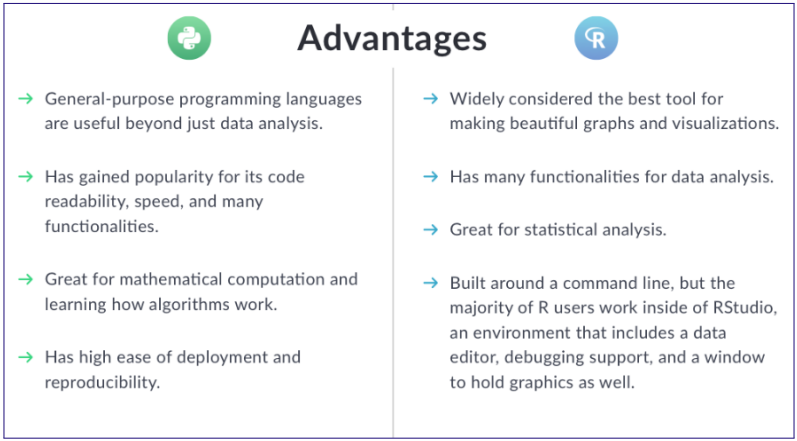


Figure 1: Comparisons Per DataCamp.com

A Note on Both Programming Languages [2]

Figure 2: Comparisons Per DataCamp.com

A Note on Both Programming Languages [3]

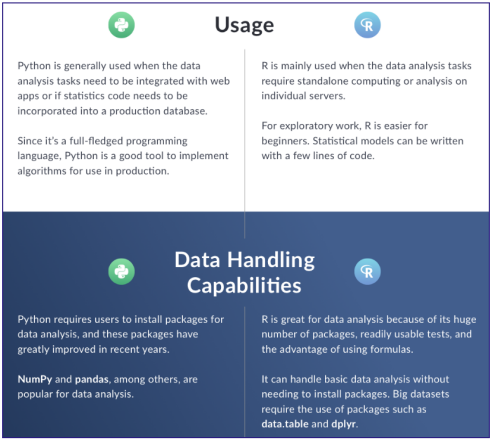


Figure 3: Comparisons Per DataCamp.com

A Note on Both Programming Languages [4]

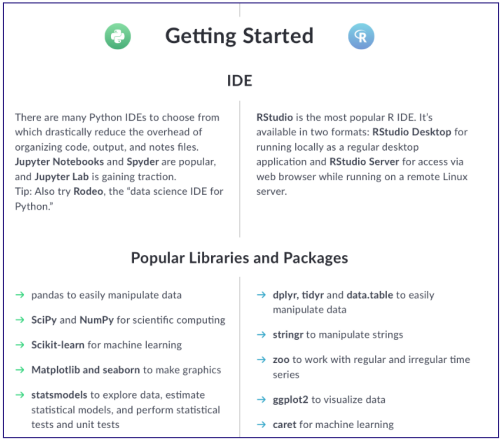


Figure 4: Comparisons Per DataCamp.com

A Note on Both Programming Languages [5]

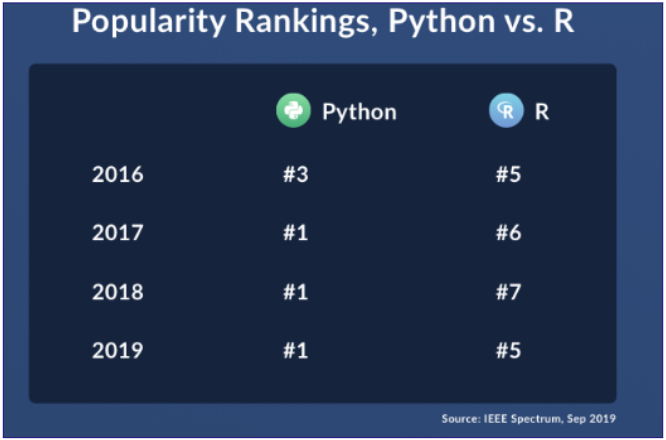


Figure 5: Comparisons Per DataCamp.com

A Note on Both Programming Languages [6]

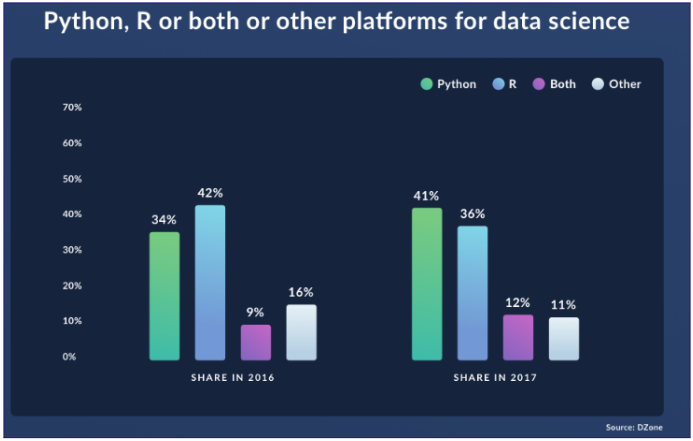


Figure 6: Comparisons Per DataCamp.com

How to Learn Any Programming Language[2]

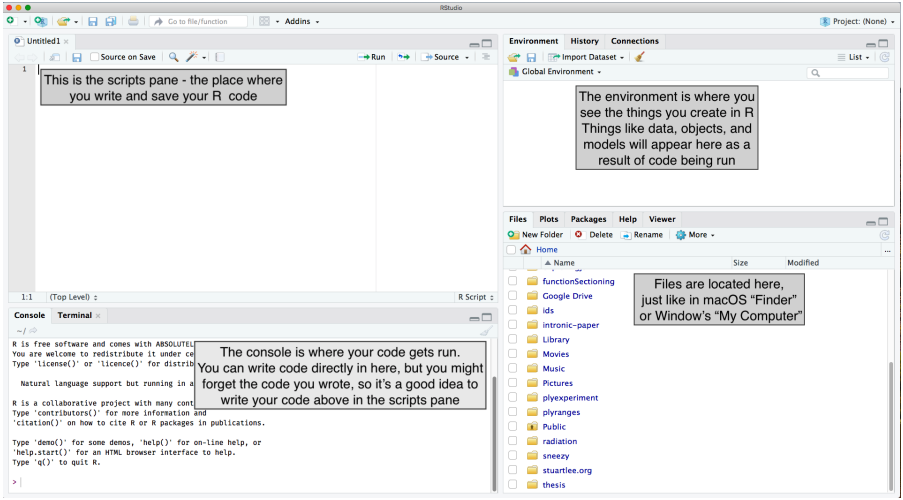
- **Get hands dirty**
- Documentation! Documentation! Documentation!
- (Not surprisingly) Learn to Google: what that error message means (I G a lot)

[1] **Source:** Slide is based on Kia Ora's How I Learn a Technology

Outline

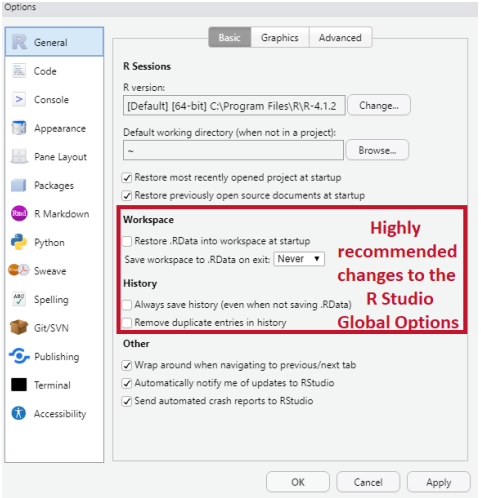
Outline

RStudio Interface

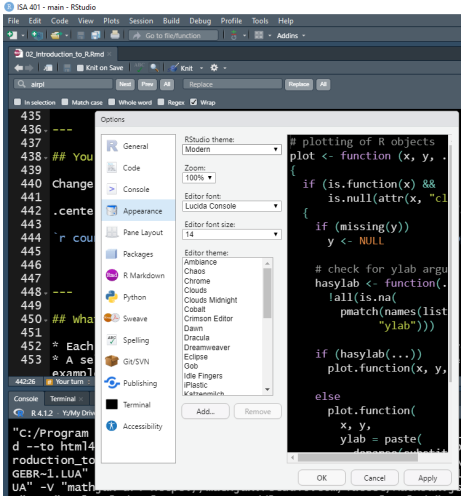


[2] **Image credit:** Stuart Lee

Setting Up RStudio (Do This Once)



Change the Appearance of RStudio to your Taste



What is an .RProject?

- Each university course is a project, and get your work organised.
- A self-contained project is a folder that contains all relevant files, for example my `ISA 419/` includes:
  - `ISA419.Rproj`
  - `figures/`
  - `lectures/`
    - `01_intro_cyber_sec/`
    - `02_software_intro/`
- All working files are **relative** to the **project root** (i.e. `isa419/`).
- The project should just work on a different computer.

Outline

Assignment

The first operator we have talked about in R is the assignment operators <-, =, and ->, which can be used as follows.

```
x1 <- 5

x2 = 5

5 -> x3

print(paste0("The values of x1, x2 and x3 are ", x1, ", ", x2, ", and ", x3, "
```

```
## [1] "The values of x1, x2 and x3 are 5, 5, and 5 respectively. As you can s
```

Note that the operator <- can be written using the shortcut "Alt" + "-" on a Windows computer.

Retrieval

We can retrieve/call the object using its name as follows:

```
x1
```

```
## [1] 5
```

```
x3
```

```
## [1] 5
```

Arthimetic

While we will not specifically talk about doing math in R, these operators are helpful for you. The table below is scrapped from Quick-R: Operators (click here for more details).

Operator	Description
+	addition
-	subtraction
*	multiplication
/	division
^ or **	exponentiation
x %% y	modulus (x mod y) 5%%2 is 1
x %/% y	integer division 5%/%2 is 2

Logical

As with the previous slide, the table below is scrapped from Quick-R: Operators (click here for more details).

Operator	Description
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to
==	exactly equal to
!=	not equal to
!x	Not x
x | y	x OR y
x & y	x AND y
isTRUE(x)	test if X is TRUE

Outline

R is a Vector Language: Sample Output

Learning from a **sample 10 obs. from the Z distribution** (i.e., $x \sim \mathcal{N}(0, 1)$)

```
x_vec = rnorm(n=10, mean = 0, sd = 1) # generating std normal dist data

x_vec > 0 # finding which elements in x are larger than 0
```

```
##  [1] FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
```

```
sum(x_vec > 0) # summing the number of elements (i.e., how many are > 0)
```

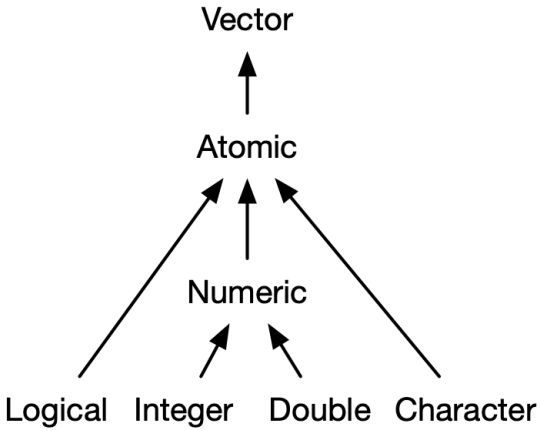
```
## [1] 5
```

If we focus on the obtained outputs, we can see that **both** are vectors:
- `x_vec > 0` returns a vector of size 10 (TRUE/FALSE for each element)
- `sum(x_vec > 0)` returns a vector of size 1

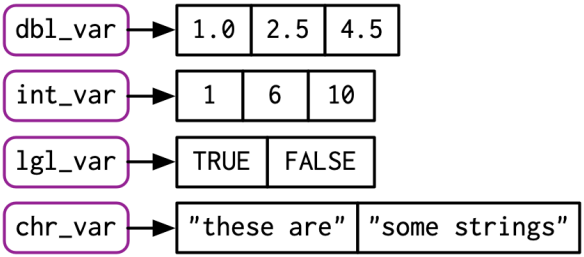
There are no scalars in R !!!

R is a Vector Language: Atomic Vectors

- Vectors come in two flavors, which differ by their elements' types:
- atomic vectors – all elements must have the same type
- lists (dataframes are a special case)– elements can be different



Data Types: A Visual Introduction [1]



- To check the **type of** an object in R, you can use the function `typeof`:

```
typeof(x_vec)
```

```
## [1] "double"
```

Data Types: A Visual Introduction [2]

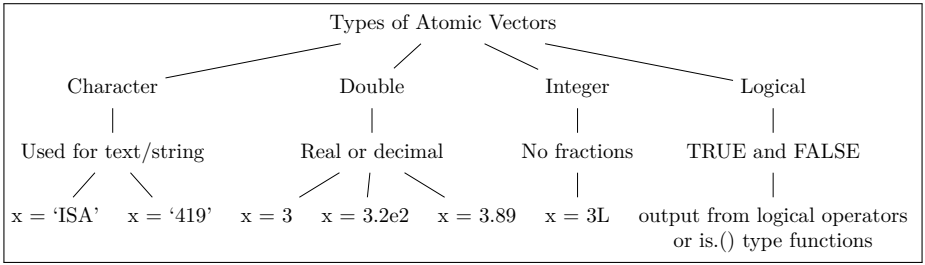


Figure 7: The four data types that we will utilize the most in our course.

Data Types: A Visual Introduction [3][3]

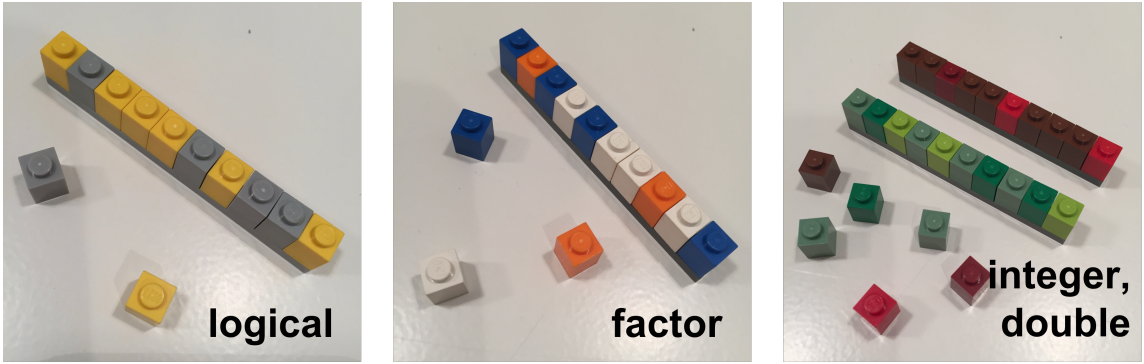


Figure 8: A visual representation of different types of atomic vectors

[2]The images are from the excellent lego-rstats GitHub Repository by Jenny Bryan

Data Structures: Atomic Vector (1D)[4]

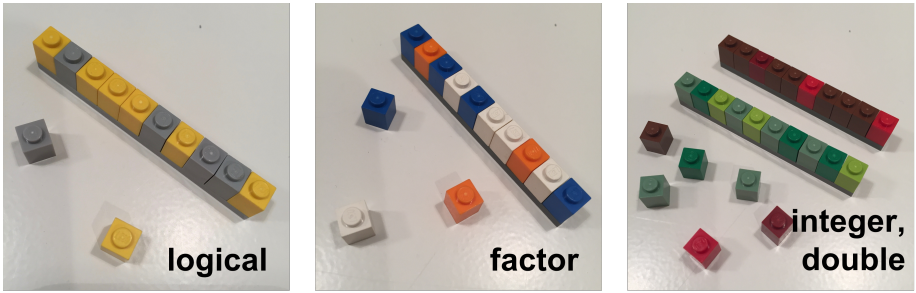
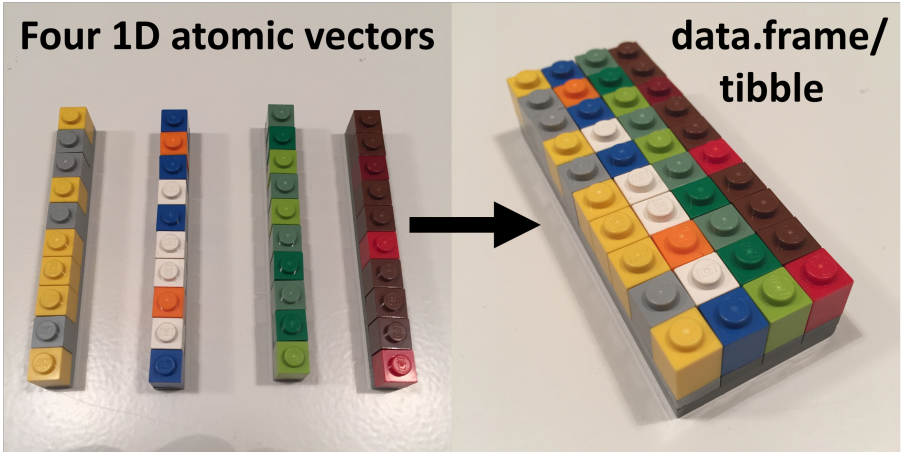


Figure 9: Keeping the visual representation of different types of atomic vectors in your head!!

```
dept = c('ACC', 'ECO', 'FIN', 'ISA', 'MGMT')
nfaculty = c(18L, 19L, 14L, 25L, 22L)
```

[3]The images are from the excellent lego-rstats GitHub Repository by Jenny Bryan

Data Structures: 1D to 2D [Visually][5]



[4]The images are from the excellent lego-rstats GitHub Repository by Jenny Bryan

Data Structures: 1D to 2D [In Code]

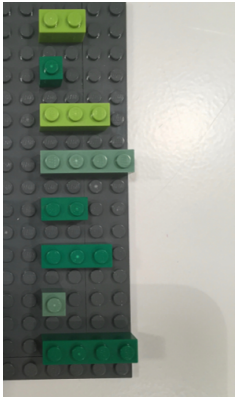
```
library(tibble)

fsb_tbl <- tibble(
  department = dept,
  count = nfaculty,
  percentage = count / sum(count))
fsb_tbl
```

```
## # A tibble: 5 x 3
##   department count percentage
##   <chr>      <int>      <dbl>
## 1 ACC           18      0.184
## 2 ECO           19      0.194
## 3 FIN           14      0.143
## 4 ISA           25      0.255
```

Data Structures: Lists [1][6]

An object contains elements of **different data types**.



[5]The images are from the excellent lego-rstats GitHub Repository by Jenny Bryan

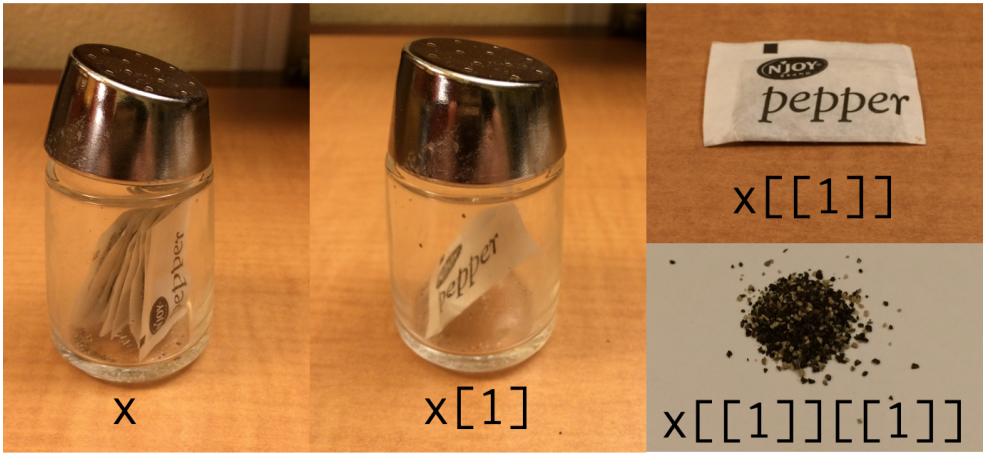
Data Structures: Lists [2]


```
lst <- list( # list constructor/creator
  1:3, # atomic double/numeric vector  of length = 3 #<<
  "a", # atomic character vector of length = 1 (aka scalar) #<<
  c(TRUE, FALSE, TRUE), # atomic logical vector of length = 3 #<<
  c(2.3, 5.9) # atomic double/numeric vector of length =3 #<<
)
writeLines( paste0(lst, collapse = '\n') ) # printing the list
```

```
## 1:3
## a
## c(TRUE, FALSE, TRUE)
## c(2.3, 5.9)
```

[6] This is inspired from Hadley Wickham's Advanced R Chapter 2 on Vectors

Data Structures: Lists [3]



[6] The image is from Hadley Wickham's Tweet on Indexing lists in R

Data Structure: Matrices

A matrix is a **2D data structure** made of **one/homogeneous data type**.

```
x_mat = matrix( sample(1:10, size = 4), nrow = 2, ncol = 2 )
str(x_mat) # its structure?
```

```
##  int [1:2, 1:2] 5 7 4 9
```

```
x_mat # printing it nicely
print('-----------------')
x_mat[1, 2] # subsetting #<<
```

```
##      [,1] [,2]
## [1,]    5    4
## [2,]    7    9
## [1] "-----------------"
## [1] 4
```

Data Structure: Data Frames [1]

```
df1 <- data.frame(x = 1:3, y = letters[1:3])

typeof(df1) # showing that its a special case of a list
```

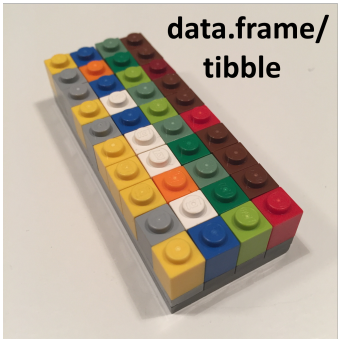
```
## [1] "list"
```

```
attributes(df1) # but also is of class data.frame
```

```
## $names
## [1] "x" "y"
##
## $class
## [1] "data.frame"
##
## $row.names
## [1] 1 2 3
```

Data Structure: Data Frames [2][7]

As noted in the creation of `df1`, columns in a data frame can be of different types. Hence, it is more widely used in data analysis than matrices.



[6]The images are from the excellent lego-rstats GitHub Repository by Jenny Bryan

Functions

A function call consists of the **function name** followed by one or more **argument** within parentheses.

```
temp_high_forecast = c(37, 37, 26, 22, 37, 27, 29, 40)
mean(x = temp_high_forecast)
```

```
## [1] 31.875
```

- function name: `mean()`, a built-in R function to compute mean of a vector
- argument: the first argument (LHS `x`) to specify the data (RHS `temp_high_forecast`)

Check the function's help page with `?mean`

Please take 2 minutes to investigate the help page for ***`mean`*** *in R Studio.*

```
mean(x = temp_high_forecast, trim = 0, na.rm = FALSE, ...)
```

- Read **Usage** section
  - What arguments have default values?
- Read **Arguments** section
  - What does `trim` do?
- Run **Example** code

Function Arguments

Match by **positions**

```
mean(temp_high_forecast, 0.1, TRUE)
```

```
## [1] 31.875
```

Match by **names**

```
mean(x = temp_high_forecast, trim = 0.1, na.rm = TRUE)
```

```
## [1] 31.875
```

Packages (i.e. Using Other People's Functions)

An awesome (yet challenging) part of the R echo system is that there are tens of thousands of package that can be easily accessed. These packages allow you to easily access a large variety of functions that are useful for data scraping, cleaning, exploration, and modeling. While there are several ways to install and load packages in R, I will use the pacman approach whenever possible due to its simplicity.

If you are not a fan of this approach, please feel free to use other approaches.

Pacman

One reason that I am a fan of the pacman approach is that it allows me to load the packages into R (and install them if they are installed, through the `p_load()`). Thus, it saves a step from the traditional `install.packages()` and then `library()` approach.

```
if(require(pacman)==FALSE) install.packages("pacman")
pacman::p_load(tidyverse, magrittr, DataExplorer, skimr)
```

Live Demo

Now, let us explore the dataset from last class.

For the data in lines 1.1M-1.2M, please answer the following questions:

- What is the timestamp of observation (i.e. row number) 1,200,000?
- What are the number of unique users for the aforementioned 100,000 observations?
- What is the number of unique computers used by U12 for the aforementioned 100,000 observations?

Outline

Python Primer

For the sake of time, let us jump into Anaconda for a live demo.

Now, let us explore the dataset from last class.

For the data in lines 1.1M-1.2M, please answer the following questions:

- What is the timestamp of observation (i.e. row number) 1,200,000?
- What are the number of unique users for the aforementioned 100,000 observations?
- What is the number of unique computers used by U12 for the aforementioned 100,000 observations?

Outline

Learning Objectives for Today's Class

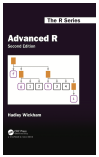
Learning Objectives

- **Understand basic syntax in both R and Python**
- **Use both platforms to solve the problem from last class**
- **Use either platform to generate reproducible reports through the R Markdown or Jupyter Notebook tools**

Things to Do [1]

- Go over your notes, read the references below, and complete this self-paced R tutorial.
- To improve your R skills, please read:

  - Workflow: basics
  - Workflow: scripts
  - Workflow: basics



  - Names and values
  - Vectors
  - Subsetting

- To improve your Python skills, please read 10 minutes to pandas
- Please complete Assignment 01

ISA 480: Data Driven Security

02 - Building your Security Analytics Toolbox

Fadel M. Megahed

Endres Associate Professor
Department of Information Systems and Analytics
Farmer School of Business
Miami University
Email: fmegahed@miamioh.edu
Office Hours: Automated Scheduler for Office Hours

Fall 2022