Data Analytics

Yong Zheng

Illinois Institute of Technology
Chicago, IL, 60616, USA

School of Applied Technology
ILLINOIS INSTITUTE OF TECHNOLOGY

Schedule

- Quick Reviews
- Intro: R

Schedule

- Quick Reviews
  - Statistical Applications
  - Data: Population and Sample
  - Data Types
  - Descriptive Statistics
    - For nominal variables
      - By metrics
      - By visualizations (note: be able to interpret plots)
    - For numerical variables
      - By metrics
      - By visualizations(note: be able to interpret plots)

Schedule

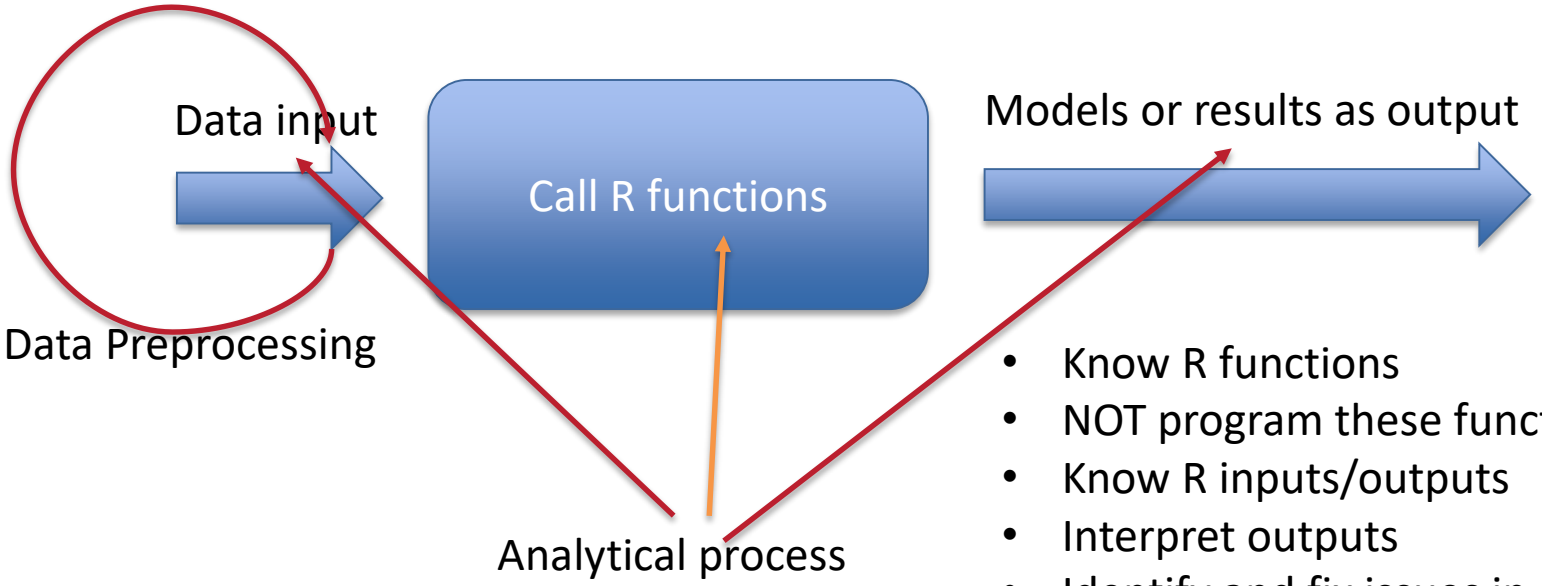
- Quick Reviews
- Intro: R

Introduction to R

- R, https://www.r-project.org/
- Open source, free, light weight
- With supports by many plugins/packages/libraries
- It is available for both Windows/Mac platforms
- R programming: R scripts/commands
- You can download and install either R or R Studio (https://www.rstudio.com/).

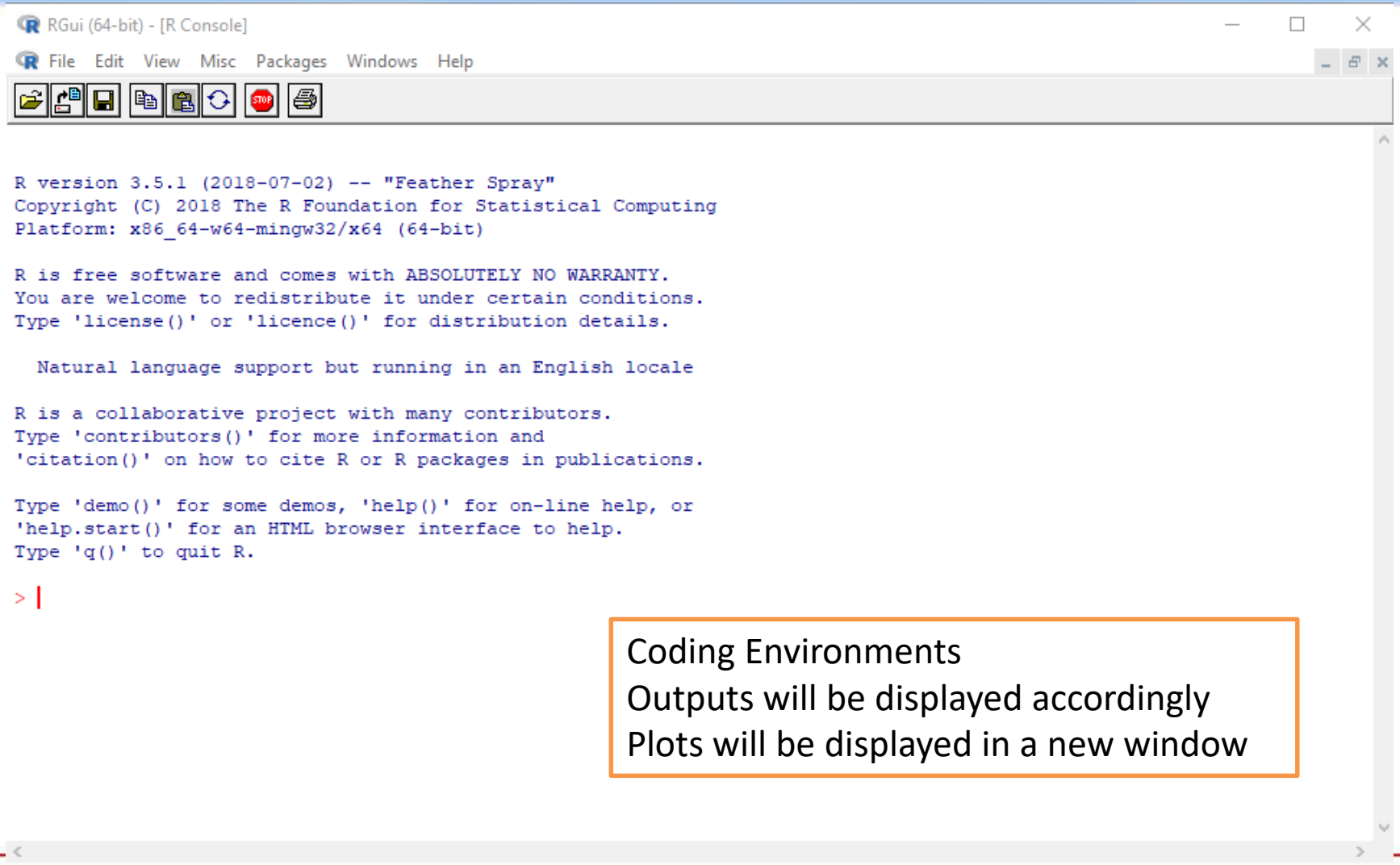
Important Notes About R

- R is considered as a scripting language, not a programming language



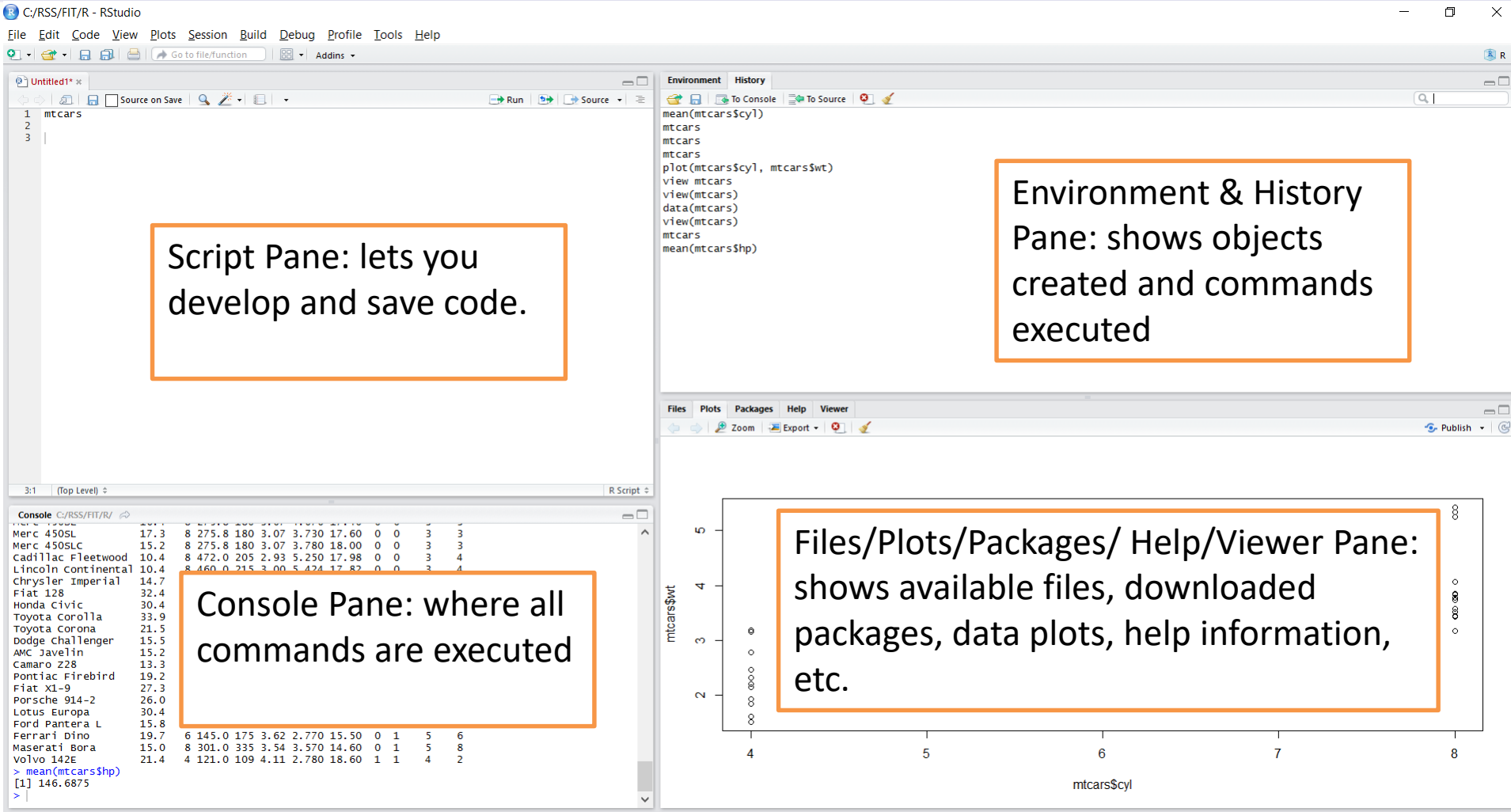
- Know R functions
- NOT program these functions
- Know R inputs/outputs
- Interpret outputs
- Identify and fix issues in outputs

R Workspace



Coding Environments
Outputs will be displayed accordingly
Plots will be displayed in a new window

RStudio Workspace



Schedule

- Quick Reviews
- Intro: R
  - Data Inputs
  - Packages or libraries
  - Descriptive Statistics

13,000+ packages of functions!!!

1. You do not need (or want) to learn all of the function packages.
2. Just focus on the few packages that are most relevant for you.
3. Some popular packages include:
   - dplyr: used for data manipulation
     - https://cran.r-project.org/web/packages/dplyr/vignettes/dplyr.html
   - stringr: used for string manipulation
     - https://cran.r-project.org/web/packages/stringr/vignettes/stringr.html
   - lubridate: used for working with date/time data
     - https://cran.r-project.org/web/packages/lubridate/vignettes/lubridate.html
   - ggplot2: used to generate data visualizations
     - http://www.r-graph-gallery.com/portfolio/ggplot2-package/

Package and Functions

- Example of the R functions


```
mydata = read.table("2009education.csv",
                    header=TRUE,
                    sep=",",
                    colClasses = c("character", rep("numeric",3)))
```

- Components
  - function name and arguments
  - objects as function output, such as mydata
  - Note: output is optional. You can store the output in an object, or just output it to console without storage

Package and Functions

- Functions are case-sensitive.
  - For example, typing “xhwe” will not execute the *XHWE* function
- Functions are organized into “packages”
  - The most commonly used *R* functions are in the “basic” package.
  - To use special functions, you need to install and load packages into the R environment.
- The R user community is continually expanding *R*’s capabilities with new functions and packages
  - For a same purpose (such as descriptive statistics), you may use different functions from different packages. You have multiple choices.

Install and use a package

- To install a package
  - install.packages("packagename")
- To call a function in a package, you must load it first
  - library(packagename)
- Afterwards, you can call functions in the package

Schedule

- Quick Reviews
- Intro: R
  - Data Inputs
  - Packages or libraries
  - Descriptive Statistics

Data input options

- Input data directly into R by giving file path
- Import data from many common file types

  (.txt, .csv, Excel files, XML, SAS, SPSS, Stata, etc)
- Or, you can also connect to web resources and other remote servers

Data Import in R

```
# output the current working path or folder
getwd()

# setup a new working path or folder
Setwd("D:/Data/")

# load data in your working path or folder
data <- read.csv(file = "allFound.csv", header = TRUE)
data = read.csv(file = "allFound.csv", header = T)
data = read.table(file = "allFound.csv", header = T, sep = ",")
```

Data Import in R

Example of the R functions


```
read.table("2009education.csv",
           header=TRUE,
           sep=",",
           colClasses = c("character", rep("numeric",3)))
```

Data Import in R

- Most likely, we will deal with data with structural tables – either in text file or in csv file

read.table(file, header = FALSE, sep = "", quote = "\'"",
dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
row.names, col.names, as.is = !stringsAsFactors,
na.strings = "NA", colClasses = NA, nrows = -1,
skip = 0, check.names = TRUE, fill = !blank.lines.skip,
strip.white = FALSE, blank.lines.skip = TRUE,
comment.char = "#",
allowEscapes = FALSE, flush = FALSE,
stringsAsFactors = default.stringsAsFactors(),
fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)

Use help function in R for API documentations
help(read.table)

read.csv(file, header = TRUE, sep = ",", quote = "\'"",
dec = ".", fill = TRUE, comment.char = "", ...)

Data Import in R

- More methods to load data into R

https://www.datacamp.com/community/tutorials/r-data-import-tutorial#txt

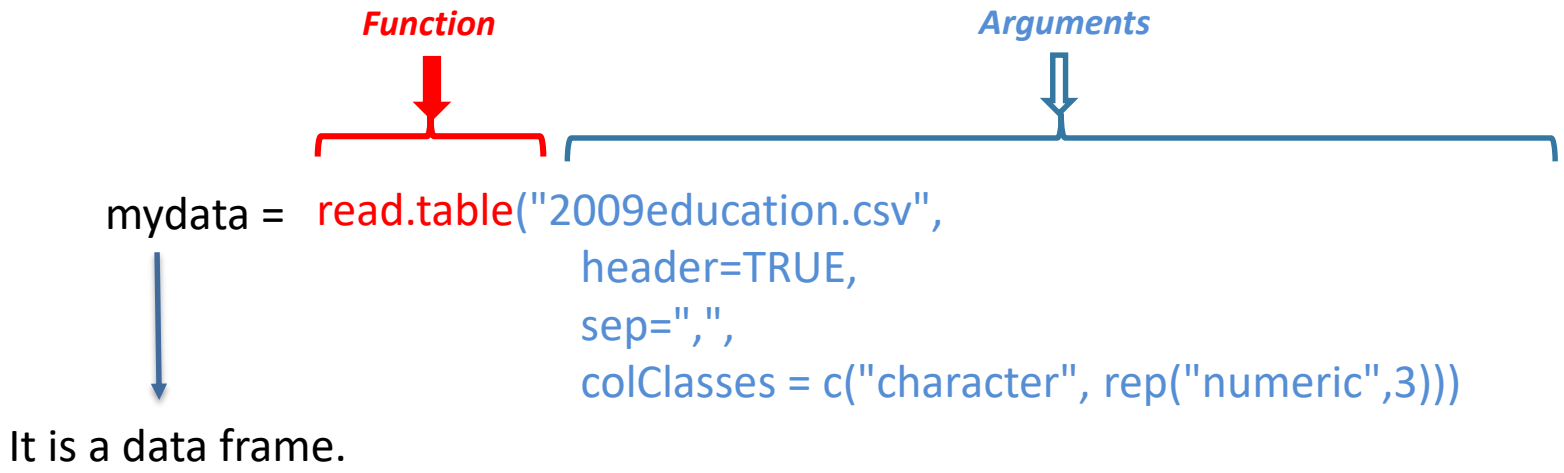
Common *R* Objects

Name	Dimensions	Contents	Example
Vector	1	• Series of values • Single data mode	12.2 9.6 -4.8 2.5
Matrix	2	• Values stored in rows and columns • All values of the same data mode	12.2 9.6 -4.8 2.5 8.3 -7.6 9.3 -2.7 -4.4 17.7 14.7 -6.9 1.7 4.5 53.4 5.2
List	1	• Series of values • Allows multiple modes	Denver 73.4 TRUE
Data Frame	2	• Values stored in rows and columns • Different columns may have different data modes	Denver 73.4 TRUE Topeka 49.8 FALSE

After loading data into R, it will automatically be shaped into a data frame

Data Import in R

Example of the R functions



Data types within data frames

- Numeric variable = ***numeric*** or ***integer***
  Ex: 1, 1.5, 200000, 3.14159
- Text variable = ***character***
  Ex: a, b, hello, 3b
- Nominal variable = ***factor***
  Ex: cat, dog, pig, rhino, horse
- Ordinal variable = ***ordered factor***
  Ex: xsmall, small, medium, large, xlarge
- True/false = ***logical***
  Ex: TRUE, FALSE

After Loading Data into R

- Some useful functions
  - str(obj) ➔ summarize the structure of an object

```
> str(rates)
'data.frame':   37791 obs. of  9 variables:
 $ zip          : chr  "35218" "35219" "35214" "35215" ...
 $ eiaid        : chr  "195" "195" "195" "195" ...
 $ utility_name: Factor w/ 145 levels "Alabama Power Co",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ state        : Factor w/ 50 levels "AK","AL","AR",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ service_type: Factor w/ 3 levels "Bundled","Delivery",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ ownership    : Factor w/ 1 level "Investor Owned": 1 1 1 1 1 1 1 1 1 1 ...
 $ comm_rate    : num  0.106 0.106 0.106 0.106 0.106 ...
 $ ind_rate     : num  0.0603 0.0603 0.0603 0.0603 0.0603 ...
 $ res_rate     : num  0.115 0.115 0.115 0.115 0.115 ...
```

  - head(data), tail(data) ➔ output the top/bottom rows
  - class(obj) ➔ get the object type, such as data frame
  - names(data) ➔ get the column names
  - dim(data) ➔ get the number of rows and columns

After Loading Data into R

```
> str(rates)
'data.frame':   37791 obs. of  9 variables:
 $ zip          : chr  "35218" "35219" "35214" "35215" ...
 $ eiaid        : chr  "195" "195" "195" "195" ...
 $ utility_name: Factor w/ 145 levels "Alabama Power Co",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ state        : Factor w/ 50 levels "AK","AL","AR",..: 2 2 2 2 2 2 2 2 2 2 ...
 $ service_type: Factor w/ 3 levels "Bundled","Delivery",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ ownership    : Factor w/ 1 level "Investor Owned": 1 1 1 1 1 1 1 1 1 1 ...
 $ comm_rate    : num  0.106 0.106 0.106 0.106 0.106 ...
 $ ind_rate     : num  0.0603 0.0603 0.0603 0.0603 0.0603 ...
 $ res_rate     : num  0.115 0.115 0.115 0.115 0.115 ...
```

```
> class(rates)
[1] "data.frame"
```

```
> names(rates)
[1] "zip"          "eiaid"        "utility_name"
[4] "state"        "service_type" "ownership"
[7] "comm_rate"    "ind_rate"     "res_rate"
```

```
> dim(rates)
[1] 37791     9
```

Retrieve Data from Data Frame

- Retrieve your data by column index
  - data[,1]: retrieve the 1st column
  - Data$grade: retrieve column by column name
- Retrieve your data by row index
  - data[3,]: retrieve the 3rd row
- If there are NA or empty strings in your column
  - grade = na.omit(data[,1])

Retrieve Data from Data Frame

- In addition, you may want to get subset of the data
- How to select specific variables

```
# select variables by column names
myvars <- c("v1", "v2", "v3")
newdata <- mydata[myvars]

# select variables by column index
# select 1st and 5th thru 10th variables
newdata <- mydata[c(1,5:10)]

# select 5th thru 10th variables
newdata <- mydata[, 5:10]
```

The c function is a function
Used for combinations

- How to exclude specific variables

```
# exclude 3rd and 5th variable
newdata <- mydata[c(-3,-5)]
```

Retrieve Data from Data Frame

- In addition, you may want to get subset of the data
- How to select specific rows

```
# the observations from row 5 to row 10
newdata <- mydata[5:10,]


# using subset function
# select all rows that have male students with age no less than 20, We keep the ID and
Weight columns.
newdata <- subset(mydata, age >= 20 & sex=="male", select=c(ID, Weight))
```

Schedule

- Quick Reviews
- Intro: R
  - Data Inputs
  - Packages or libraries
  - Descriptive Statistics

Describe Qualitative Data

- Describe qualitative data Numerically
  - By class frequency
  - By class relative frequency
- Describe qualitative data by visualizations
  - By bar graph
  - By pie chart

Describe Nominal Data by Using Case Study 1

```
# set environment and load data into R
setwd("D:/GoogleDrive/Courses/IIT/2019 Spring/ITMD 527 - Data Analytics/Case Studies")
data=read.table("case1_student Grades_regular.csv",header=T, sep=',')

# get a summary of the data
str(data)

# let's focus on the variable Nationality
nat=data$Nationality
class(nat)

# install and load library plyr
install.packages('plyr')
library(plyr)

# call the function count in plyr
count(nat)
```

```
# If you want to get class relative frequency
table(data$Nationality)/nrow(data)
```

Describe Nominal Data by Using Case Study 1

```
> data=read.table("case1_student Grades_regular.csv",header=T, sep=',')
> str(data)
'data.frame':   600 obs. of  12 variables:
 $ ID                  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ Nationality         : Factor w/ 4 levels " China"," France",..: 1 2 2 3 3 3 3 3 2 2 ...
 $ Gender              : int  1 0 0 1 1 1 1 1 0 0 ...
 $ Age                 : int  21 26 20 18 18 18 18 20 19 20 ...
 $ Degree              : Factor w/ 3 levels " BS"," MS"," PHD": 1 3 3 2 2 1 1 1 1 1 ...
 $ Hours.on.Readings   : int  12 0 0 1 1 0 0 0 1 1 ...
 $ Hours.on.Assignments: int  10 6 6 6 6 5 5 5 13 13 ...
 $ Hours.on.Games      : int  1 9 9 12 12 11 11 11 0 0 ...
 $ Hours.on.Internet   : int  12 4 4 5 5 0 0 0 0 0 ...
 $ Exam                : num  97.2 77.5 65.7 60.4 46.9 ...
 $ Grade               : num  85.3 70.5 72.9 63.8 62.3 ...
 $ GradeLetter         : Factor w/ 4 levels "A","B","C","F": 2 3 3 3 3 3 4 3 1 1 ...
> nat=data$Nationality
> class(nat)
[1] "factor"
```

```
> library(plyr)
Warning message:
package 'plyr' was built under R version 3.5.2
> count(nat)
        x freq
1   China  176
2  France   96
3   India  133
4   Spain  195
```

```
> table(data$Nationality)/nrow(data)

    China    France     India     Spain
0.2933333 0.1600000 0.2216667 0.3250000
>
```

Describe Nominal Data by Using Case Study 1

```
# prepare the data for plots
opt=count(nat)
cf=opt$freq
labels=opt$x
crf=table(data$Nationality)/nrow(data)

# produce Pie Chart
pie(crf,labels)

# produce bar graph by using class frequency
barplot(cf,names.arg=labels)
```

Save outputs

```
> library(plyr)
Warning message:
package 'plyr' was built under R version 3.5.2
> count(nat)
        x freq
1   China  176
2  France   96
3   India  133
4   Spain  195
```

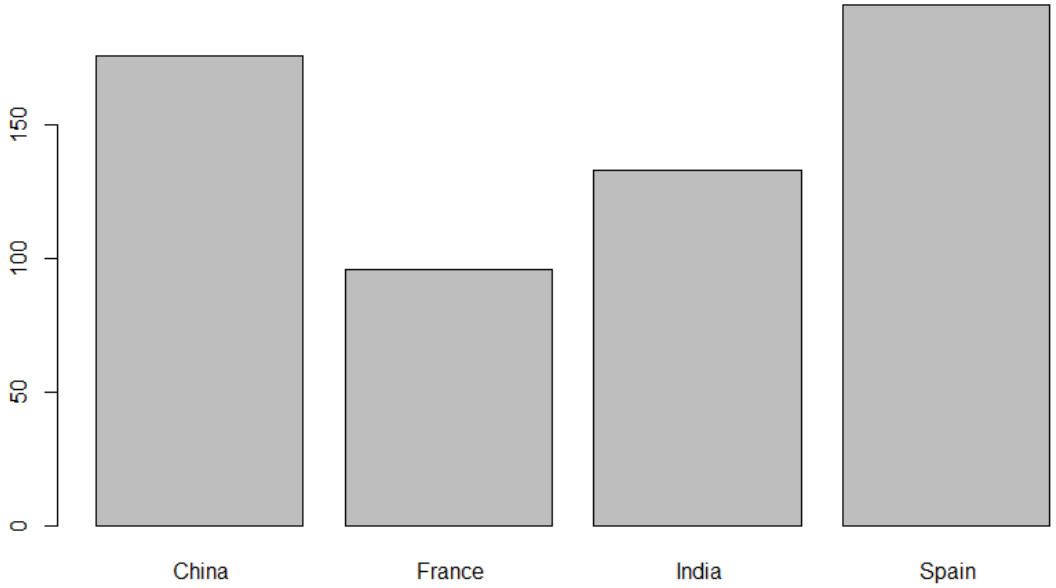
Save outputs

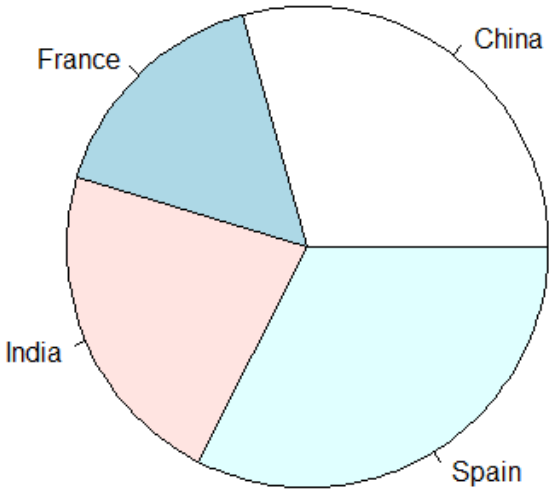
```
> table(data$Nationality)/nrow(data)

    China    France     India     Spain
0.2933333 0.1600000 0.2216667 0.3250000
>
```

```
# if you would like to display percentages on the Pie Chart
# see examples, https://www.tutorialspoint.com/r/r_pie_charts.htm
```

Describe Nominal Data by Using Case Study 1





Describe Quantitative Data

- Describe quantitative data Numerically
  - By range, min, max, mean, median, mode
  - By variance, standard deviation
  - By q1, q2, q3
- Describe quantitative data by visualizations
  - ~~By stem-and-leaf~~
  - By histogram
  - By box plot
  - By probability distribution

Describe Numerical Data by Using Case Study 1

```
# Let's focus on the variable Grade
g=data$Grade

# note that if you have missing values in it, you cannot produce numerical metrics
# you need to use na.omit to ignore the missing values
summary(g)

# use describe function in package 'psych'
install.packages('psych')
library(psych)
describe(g)
```

```
> summary(g)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  50.29   70.80   81.20   79.55   90.52   99.96

> describe(g)
   vars   n  mean    sd median trimmed   mad   min   max range  skew kurtosis   se
X1    1 600 79.55 12.91   81.2   80.23 14.49 50.29 99.96 49.67 -0.38     -0.8 0.53
```

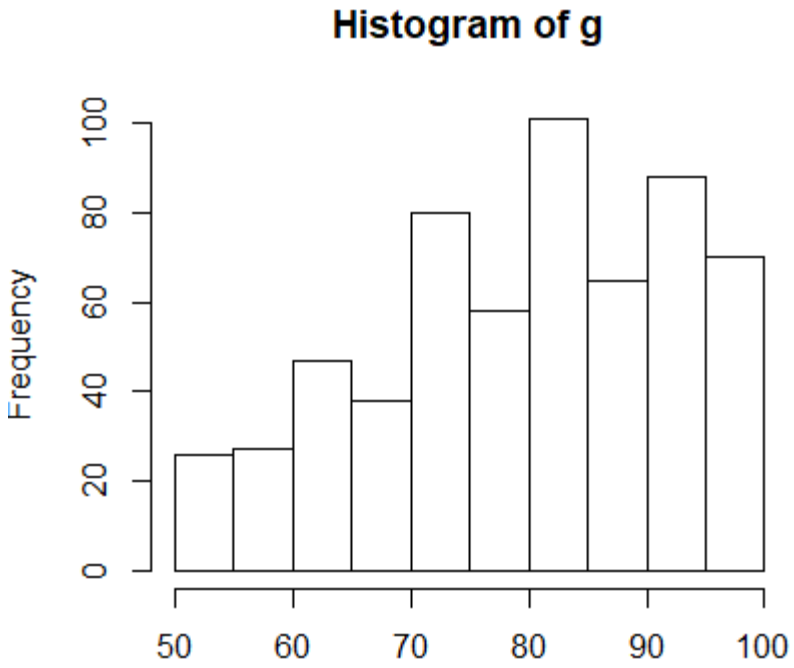
Describe Numerical Data by Using Case Study 1

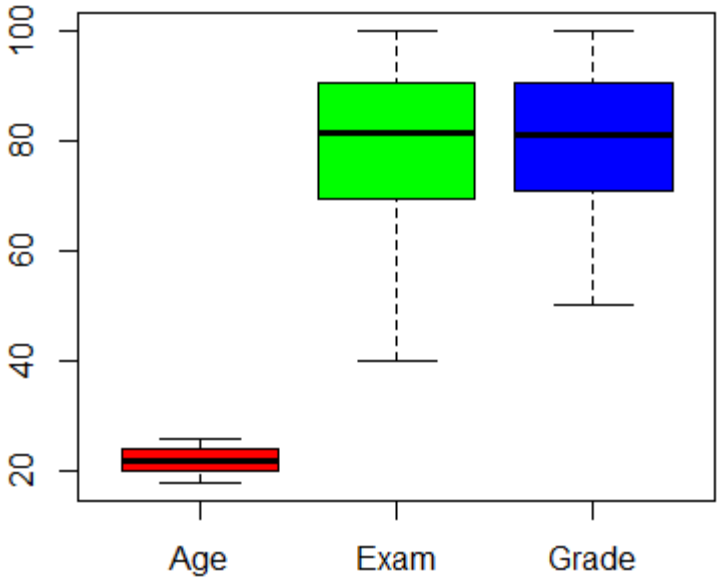
```
# histogram
hist(g)

# boxplot to compare Age, Grade and Exam
# prepare your data
values=data[,c('Age','Exam','Grade')]
boxplot(values,col=rainbow(ncol(values)))
```

Give rainbow colors to the plots

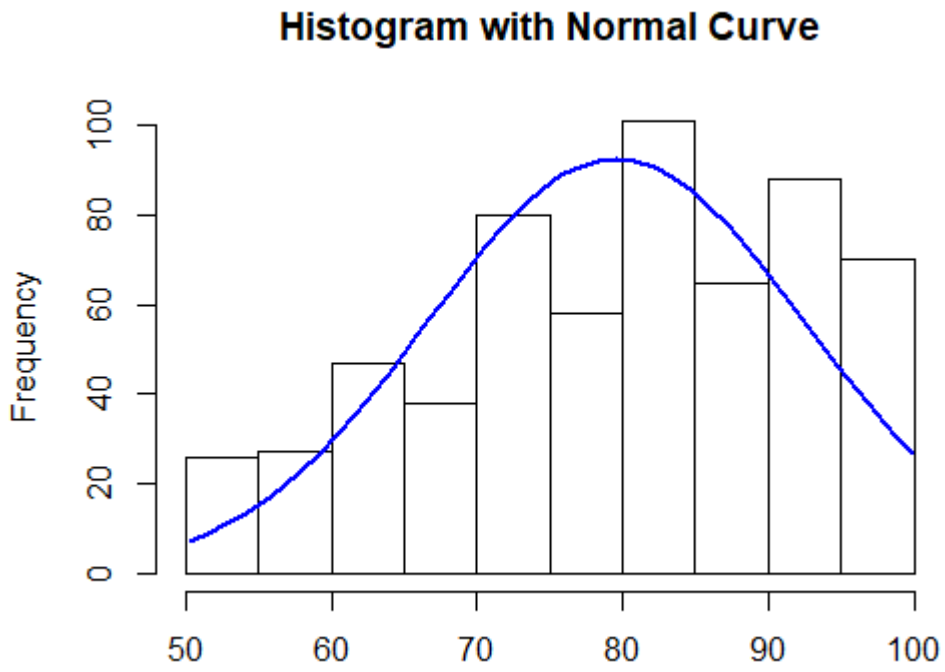
Describe Numerical Data by Using Case Study 1





Describe Numerical Data by Using Case Study 1

```
# plot histogram with normal curve
h<-hist(g, main="Histogram with Normal Curve")
xfit<-seq(min(g),max(g),length=40)
yfit<-dnorm(xfit,mean=mean(g),sd=sd(g))
yfit <- yfit*diff(h$mids[1:2])*length(g)
lines(xfit, yfit, col="blue", lwd=2)
```



Schedule

- Practice by yourself
- Next Class: Inferential Statistics