WEEK 2: R

Gates

TOPICS

1. Working with data and dataframes in R:
- Reshaping data in R: row-wise and column-wise joining, row manipulations, column manipulations, sorting, merging, subsetting, binning.

2. The apply() Family
3. Cleaning and preparing data in R: regular expression, updating data, removing data, dealing with missing or NA data
4. Packages in R - these will be part of various examples and will not be exhaustive.
5. Data partitioning in R: kmeans, PCA
6. Visualization in R: plot, ggplot, qplot, leaflet

INSTRUCTIONS

- This PowerPoint Guide will cover many core R topics.
- As you review the Guide, type all examples into R and practice.
- Use the Internet, resources, or books to fill in the blanks and extra detail.

NEXT TOPIC

1. Working with data and dataframes in R:
- Reshaping data in R: row-wise and column-wise joining, row manipulations, column manipulations, sorting, merging, subsetting, binning.

2. The apply() Family
3. Cleaning and preparing data in R: regular expression, updating data, removing data, dealing with missing or NA data
4. Packages in R - these will be part of various examples and will not be exhaustive.
5. Data partitioning in R: kmeans, PCA
6. Visualization in R: plot, ggplot, qplot, leaflet

MORE WITH DATAFRAMES

1) It is critical to become an expert with using dataframes.

2) Using dataframes, you can add features, you can remove features, you can add and remove rows, you can blend datasets into one, you can clean data, etc.

3) The next several slides will look at examples using dataframes and related methods.

4) Throughout this PowerPoint, I will also include Packages as needed. Whenever I include a new Package – take the time to look it up and to learn more about it.

REMINDER: CREATE A DATAFRAME

```
394  ##############################
395  ##MORE ON DATA FRAMES
396  ##############################
397  ##
398  ##Make a dataframe
399  NewDataFrame <- data.frame(name=c("John", "Sally", "Fred"),
400                             gender=c("M","F","M"), age=c(23,45,67))
401  NewDataFrame
402
```

397:3 Reading data R S

Console ~/RStudioFolder/

```
> NewDataFrame
   name gender age
1  John      M  23
2 Sally      F  45
3  Fred      M  67
>
```

ACCESSING DF – SEE RESULTS ON NEXT SLIDE

```
##Accessing portions of dataframe ##columns by name
NewDataFrame$name #creates a factor
class(NewDataFrame$name)
NewDataFrame["name"] ##creates a dataframe
class(NewDataFrame["name"])
##columns and certain rows #Row 1 and 2 and column 2 and 3
NewDataFrame[c(1,2),c(2,3)]
##access one element
NewDataFrame[2,2]
##access a column and then an element in the col
NewDataFrame$gender[1]
```

ACCESSING DATAFRAMES

Be sure to practice.
Try these methods.
Create your own dataframes and/or read data into a dataframe and try the different methods.

```
> NewDataFrame$name #creates a factor
[1] John  Sally Fred
Levels: Fred John Sally
> class(NewDataFrame$name)
[1] "factor"
> NewDataFrame["name"] ##creates a dataframe
   name
1  John
2 Sally
3  Fred
> class(NewDataFrame["name"])
[1] "data.frame"
> ##columns and cetain rows
> #Row 1 and 2 and column 2 and 3
> NewDataFrame[c(1,2),c(2,3)]
  gender age
1      M  23
2      F  45
> ##access one element
> NewDataFrame[2,2]
[1] F
Levels: F M
> ##access a column and then an element in the col
> NewDataFrame$gender[1]
[1] M
Levels: F M
```

DATAFRAMES: MEASURES, UPDATING ROWS, GETTING NAMES

```
##measures
(AVG <- mean(NewDataFrame$age))
##column names
names(NewDataFrame)
rownames(NewDataFrame)
##Name the rows
rownames(NewDataFrame) <- c("one","two","three")
NewDataFrame
#Place a df col into a list
AgeList <- NewDataFrame$age
class(AgeList)
(MED <- median(AgeList))
```

```
> ##measures
> (AVG <- mean(NewDataFrame$age))
[1] 45
> ##column names
> names(NewDataFrame)
[1] "name"   "gender" "age"
> rownames(NewDataFrame)
[1] "1" "2" "3"
> ##Name the rows
> rownames(NewDataFrame) <- c("one","two","three")
> NewDataFrame
       name gender age
one    John      M  23
two   Sally      F  45
three  Fred      M  67
> #Place a df col into a list
> AgeList <- NewDataFrame$age
> class(AgeList)
[1] "numeric"
> (MED <- median(AgeList))
[1] 45
>
```

ADDING A NEW COLUMN TO DF

This adds a new column called "height" and the associated data.

```
436  ##Add a column to df
437  NewDataFrame$height <- c(67,69,60)
438  NewDataFrame
439
```

437:35 Reading data

Console ~/RStudioFolder/

```
> ##Add a column to df
> NewDataFrame$height <- c(67,69,60)
> NewDataFrame
      name gender age height
one    John      M  23     67
two   Sally      F  45     69
three  Fred      M  67     60
>
```

RBIND AND CBIND: RESULTS ON NEXT SLIDE

```
##Using rbind to add rows to df
Otherdf <- data.frame(name=c("Harry", "Alana"), gender=c("M","F"),
                      age=c(45,21), height=c(67,65))
rownames(Otherdf) <- c("four","five")
Bigger_df <- rbind(NewDataFrame,Otherdf)
Bigger_df
##Using cbind to bind columns together ## Sizes need to match when binding
(Anotherdf <- data.frame(weight=c(126,123,98,107,111), rank=c(1,2,2,1,1)))
(Bigger_df <- cbind(Bigger_df,Anotherdf))
```

RESULTS FOR RBIND AND CBIND

Remember – you must type all of these examples in. You must practice and see what each method does. Recall that methods like rbind (binding rows) and cbind (binding columns) require matched sizes.

```
> ##Using rbind to add rows to df
> Otherdf <- data.frame(name=c("Harry", "Alana"), gender=c("M","F"),
+                       age=c(45,21), height=c(67,65))
> rownames(Otherdf) <- c("four","five")
> Bigger_df <- rbind(NewDataFrame,Otherdf)
>
> Bigger_df
       name gender age height
one    John      M  23     67
two   Sally      F  45     69
three  Fred      M  67     60
four  Harry      M  45     67
five  Alana      F  21     65
>
> ##Using cbind to bind columns together
> (Anotherdf <- data.frame(weight=c(126,123,98,107,111), rank=c(1,2,2,1,1)))
  weight rank
1    126    1
2    123    2
3     98    2
4    107    1
5    111    1
> (Bigger_df <- cbind(Bigger_df,Anotherdf))
       name gender age height weight rank
one    John      M  23     67    126    1
two   Sally      F  45     69    123    2
three  Fred      M  67     60     98    2
four  Harry      M  45     67    107    1
five  Alana      F  21     65    111    1
>
```

MERGING TWO DATAFRAMES BY=

```
##Merging two data frames by ID
## In this case, both dataframes have ID that are the same
df1 <- Bigger_df
df1$ID <- c(11,22,33,44,55) #add column
df1
(df2 <- data.frame(ID=c(11,22,33,44,55), grade=c(90,78,99,71,94)))
(df12 <- merge(df1,df2,by="ID"))
```

MERGE BY="ID" EXAMPLE

```
> ##Merging two data frames by ID
> ## In this case, both dataframes have ID that are the same
> df1 <- Bigger_df
> df1$ID <- c(11,22,33,44,55) #add column
> df1
       name gender age height weight rank ID
one    John      M  23     67    126    1 11
two   Sally      F  45     69    123    2 22
three  Fred      M  67     60     98    2 33
four  Harry      M  45     67    107    1 44
five  Alana      F  21     65    111    1 55
> (df2 <- data.frame(ID=c(11,22,33,44,55), grade=c(90,78,99,71,94)))
  ID grade
1 11    90
2 22    78
3 33    99
4 44    71
5 55    94
> (df12 <- merge(df1,df2,by="ID"))
  ID  name gender age height weight rank grade
1 11  John      M  23     67    126    1    90
2 22 Sally      F  45     69    123    2    78
3 33  Fred      M  67     60     98    2    99
4 44 Harry      M  45     67    107    1    71
5 55 Alana      F  21     65    111    1    94
>
```

REMOVE A COLUMN AND ROWS BASED ON CRITERIA

```
##Remove the column called weight
(smallerdf <- subset(df12, select= -weight))

##Remove rows with age < 25
(smallerdf <- smallerdf[smallerdf$age>25,])
## Notice the commas after the 25. You need this because you are
## removing rows. Also recall that the parenthesis around the entire
##statement causes the result to print.
```

REMOVE ROWS AND COLUMNS: SUBSETTING

```
> df12
  ID  name gender age height weight rank grade
1 11  John      M  23     67    126    1    90
2 22 Sally      F  45     69    123    2    78
3 33  Fred      M  67     60     98    2    99
4 44 Harry      M  45     67    107    1    71
5 55 Alana      F  21     65    111    1    94
> ##Remove the column called weight
> (smallerdf <- subset(df12, select= -weight))
  ID  name gender age height rank grade
1 11  John      M  23     67    1    90
2 22 Sally      F  45     69    2    78
3 33  Fred      M  67     60    2    99
4 44 Harry      M  45     67    1    71
5 55 Alana      F  21     65    1    94
> ##Remove rows with age < 25
> (smallerdf <- smallerdf[smallerdf$age>25,])
  ID  name gender age height rank grade
2 22 Sally      F  45     69    2    78
3 33  Fred      M  67     60    2    99
4 44 Harry      M  45     67    1    71
>
```

BINNING: CREATING A NEW FEATURE BASED ON A COLUMN VALUE

```
472 ##Create a categorical feature
473 ##Create a new column called over40
474 ##Based on age
475
476 (MyDataFrame <- data.frame(name=c("john", "sally", "bill", "Ally"),
477                             age=c(21, 56, 23, 45), weight=c(112,104,125,116)))
478
479 MyDataFrame$over40 <- ifelse(MyDataFrame$age > 40,1,0)
480 MyDataFrame
481
482
483
```

482:1 Reading data R Scr

Console ~/RStudioFolder/

```
   name age weight
1  john  21    112
2 sally  56    104
3  bill  23    125
4  Ally  45    116
>
> MyDataFrame$over40 <- ifelse(MyDataFrame$age > 40,1,0)
> MyDataFrame
   name age weight over40
1  john  21    112      0
2 sally  56    104      1
3  bill  23    125      0
4  Ally  45    116      1
```

SORTING BY A COLUMN

```
467  ##Sorting by a column
468  Bigger_df
469  Bigger_df <- Bigger_df[order(Bigger_df$age),]
470  Bigger_df
471
```

465:1 Reading data

Console ~/RStudioFolder/

```
> Bigger_df
      name gender age height weight rank
one    John      M  23     67    126    1
two   Sally      F  45     69    123    2
three  Fred      M  67     60     98    2
four  Harry      M  45     67    107    1
five  Alana      F  21     65    111    1
> Bigger_df <- Bigger_df[order(Bigger_df$age),]
> Bigger_df
      name gender age height weight rank
five  Alana      F  21     65    111    1
one    John      M  23     67    126    1
two   Sally      F  45     69    123    2
four  Harry      M  45     67    107    1
three  Fred      M  67     60     98    2
>
```

NEXT TOPIC

1. Working with data and dataframes in R:
- Reshaping data in R: row-wise and column-wise joining, row manipulations, column manipulations, sorting, merging, subsetting, binning.

2. The apply() Family

3. Cleaning and preparing data in R: regular expression, updating data, removing data, dealing with missing or NA data
4. Packages in R - these will be part of various examples and will not be exhaustive.
5. Data partitioning in R: kmeans, PCA
6. Visualization in R: plot, ggplot, qplot, leaflet

USING APPLY AND LAPPLY

```
##Examples
data <- list(x = 1:5, y = 6:10, z = 11:15)
data
lapply(data, FUN = median)

Mydf=data.frame(x=c(1,2,3,4), y=c(20,30,40,50), z=c(100,200,300,400))
Mydf
apply(Mydf[,c('x','z')], 1, function(x) sum(x) )
apply(Mydf[,c('x','y')], 2, function(x) sqrt(x))
apply(Mydf[,c('x','y')], 2, FUN=mean)
apply(Mydf[,c('x','z')], 2, FUN=sqrt)
```

Further References:

https://nsaunders.wordpress.com/2010/08/20/a-brief-introduction-to-apply-in-r/

https://www.r-bloggers.com/the-r-apply-function-a-tutorial-with-examples/

RESULTS FROM CODE ON PREVIOUS SLIDE

```
> Mydf=data.frame(x=c(1,2,3,4), y=c(20,30,40,50), z=c(100,200,300,400))
> Mydf
  x  y   z
1 1 20 100
2 2 30 200
3 3 40 300
4 4 50 400
> apply(Mydf[,c('x','z')], 1, function(x) sum(x) )
[1] 101 202 303 404
> apply(Mydf[,c('x','y')], 2, function(x) sqrt(x))
             x        y
[1,] 1.000000 4.472136
[2,] 1.414214 5.477226
[3,] 1.732051 6.324555
[4,] 2.000000 7.071068
> apply(Mydf[,c('x','y')], 2, FUN=mean)
   x    y
 2.5 35.0
> apply(Mydf[,c('x','z')], 2, FUN=sqrt)
             x        z
[1,] 1.000000 10.00000
[2,] 1.414214 14.14214
[3,] 1.732051 17.32051
[4,] 2.000000 20.00000
>
```

MORE ON APPLY

```
##More on apply
dat <- data.frame(x=c(1,2), y=c(3,4), z=c(5,6))
dat
##apply to dat all rows and columns x and z rowwise the sum
apply(dat[,c('x','z')], 1, function(x) sum(x) )
rowSums(dat[,c('x','z')]) #does the same thing
someFunc <- function(a, b) a * b
newFunc <- function(c,d) c^d
apply(dat[,c('x','z')], 1, function(x) someFunc(x[1],x[2]))  #The 1 is rowwise
apply(dat[,c('x','y')], 2, function(x) newFunc(x[1],x[2]))   #The 2 is columnwise
```

https://datascienceplus.com/using-the-apply-family-of-functions-in-r/

CODE RESULTS

```
> dat <- data.frame(x=c(1,2), y=c(3,4), z=c(5,6))
> dat
  x y z
1 1 3 5
2 2 4 6
> apply(dat[,c('x','z')], 1, function(x) sum(x) )
[1] 6 8
> rowSums(dat[,c('x','z')])
[1] 6 8
> someFunc <- function(a, b) a * b
> apply(dat[,c('x','z')], 1, function(x) someFunc(x[1],x[2]))
[1]  5 12
> someFunc <- function(a, b) a * b
> newFunc <- function(c,d) c^d
> apply(dat[,c('x','z')], 1, function(x) someFunc(x[1],x[2]))
[1]  5 12
> apply(dat[,c('x','y')], 2, function(x) newFunc(x[1],x[2]))
 x  y
 1 81
>
```

APPLY USING THE IRIS DATASET IN R

#Example of apply using the iris dataset

#Create a function that takes a single argument x.

#If x is numeric, it returns mean(x)

otherwise, it returns NA.

sapply() traverses the list (each column) in turn,

attach(iris)

sapply(iris, function(x) ifelse(is.numeric(x), mean(x), NA))

```
> attach(iris)
> sapply(iris, function(x) ifelse(is.numeric(x), mean(x), NA))
Sepal.Length  Sepal.Width Petal.Length  Petal.Width      Species
    5.843333     3.057333     3.758000     1.199333           NA
```

NEXT TOPIC

1. Working with data and dataframes in R:
- Reshaping data in R: row-wise and column-wise joining, row manipulations, column manipulations, sorting, merging, subsetting, binning.

2. The apply() Family
3. Cleaning and preparing data in R: regular expression, updating data, removing data, dealing with missing or NA data
4. Packages in R - these will be part of various examples and will not be exhaustive.
5. Data partitioning in R: kmeans, PCA
6. Visualization in R: plot, ggplot, qplot, leaflet

CLEANING DATA IN R

1) It is common to hear that data cleaning and preparation is 80% of data analysis.

2) Datasets often require considerable pre-processing before analysis can begin.

3) Such pre-processing/cleaning can include removing NA or incorrect values, dealing with outliers, correcting errors, removing unwanted additions, etc. etc.

4) This Guide is not on data cleaning per se. However, you must learn methods in R that will enable you to prepare data.

CODE TO REMOVE/UPDATE DATAFRAMES

```
##Read in a small dataset

ddata <-
read.csv("ANLY500DirtyData.csv", sep=",",
header = TRUE)

head(ddata)

#Remove rows with incorrect id

ddata <-ddata[ddata$id >= 1000 &
ddata$id <= 9999, ]

## This removes any row with NA

## another option: Remove rows with NA

ddata <- ddata[ ! ddata$gender %in%
c("NA", "<NA>", NA, "na"), ]
```

```
## Remove incorrect ages, weights, heights
ddata <-ddata[ddata$age >= 18 & ddata$age <= 105, ]
ddata <-ddata[ddata$weight >= 50 & ddata$weight <=
500, ]
ddata <-ddata[ddata$height >= 50 & ddata$height <= 80, ]
##Convert the income to numbers and remove commas
ddata$income <- as.numeric(gsub(",","",ddata$income))
#remove income less than 10000
ddata <-ddata[ddata$income > 10000, ]
#option for removing rows w NA
ddata <- ddata[complete.cases(ddata), ] ddata
```

RESULTS FROM PREVIOUS SLIDE

Here, this is the original dataset. Once all the rows are removed, only one row remains.

This is an extreme example that shows how to adjust rows and columns.

The previous slide has all the code updates.

```
> ddata <- read.csv("ANLY500DirtyData.csv", sep=",", header = TRUE)
> head(ddata)
    id age weight height gender income
1 3461 129    120     70      M      0
2 3977  45     99     65      F     -1
3  111  21    900     68      0  50000
4 6999  34    109     69      F 60,000
5 4578  65    112     NA      M 100000
6 7649  33    135     68   <NA>  apple
```

```
> ddata
    id age weight height gender income
4 6999  34    109     69      F  60000
```

NEXT TOPIC

1. Working with data and dataframes in R:
- Reshaping data in R: row-wise and column-wise joining, row manipulations, column manipulations, sorting, merging, subsetting, binning.

2. The apply() Family
3. Cleaning and preparing data in R: regular expression, updating data, removing data, dealing with missing or NA data
4. Packages in R - these will be part of various examples and will not be exhaustive.
5. Data partitioning in R: kmeans, PCA
6. Visualization in R: plot, ggplot, qplot, leaflet

PACKAGES/LIBRARIES

1) There are hundreds of packages that can be used with R.

2) The best method to use to discover packages is to search for a topic in R, such as PCA or clustering or text manipulation.

3) On the following slide is a list of packages that I used to create a leaflet interactive online map. This list is not exhaustive. It will give you an idea of some popular packages that you can look up and practice with.

4) There is no magic method to know every package. It is better to learn how to research and learn about new packages.

LIBRARIES THAT I USED IN MY LAST PROGRAM

```
library(leaflet)
library(sp)
library(mapproj)
library(maps)
library(mapdata)
library(maptools)
library(htmlwidgets)
library(magrittr)
library(XML)
library(plyr)
library(rgdal)
library(WDI)
library(raster)
library(noncensus)
library(stringr)
library(tidyr)
library(tigris)
library(rgeos)
library(ggplot2)
library(scales)
```

POPULAR PACKAGES/LIBRARIES REFERENCES

https://support.rstudio.com/hc/en-us/articles/201057987-Quick-list-of-useful-R-packages

http://www.computerworld.com/article/2921176/business-intelligence/great-r-packages-for-data-import-wrangling-visualization.html

https://www.r-bloggers.com/the-50-most-used-r-packages/

A CLOSER LOOK AT A FEW KEY LIBRARIES

1) ggplot
2) stringr
3) plyr
4) dplyr

GGPLOT

For this topic, I have created an entire PowerPoint Guide.

You can locate it HERE:

http://drgates.georgetown.domains/ANLY500/Week2ggplotR.pdf

(Or on the class site on the Outline)

http://drgates.georgetown.domains/ANLY500/Outline.html

STRINGR

There are **four main families of functions in stringr:**

Character manipulation: these functions allow you to manipulate the individual characters inside the strings inside character vectors.

Whitespace tools to add, remove, and manipulation whitespace.

Locale sensitive operation whose operation will vary for locale to locale

Pattern matching functions. These recognise four engines of pattern description. The most common is regular expresssions, but there are a three other tools.

STRINGR EXAMPLES

You can access **individual characters** using **sub_str().**

It takes **three arguments**: a character vector, a starting position and an end position.

It is "inclusive".

Because x has two strings, the str_length works on both.

```
564  #########stringr
565  library(stringr)
566  str_length("abc")
567  x <- c("abcdef", "ghifjk")
568  str_sub(x, 3, 3)
569  str_sub(x, 3, 5)
570
571
572
```

570:1 PACKAGES

Console ~/RStudioFolder/

```
> library(stringr)
> str_length("abc")
[1] 3
> x <- c("abcdef", "ghifjk")
> str_sub(x, 3, 3)
[1] "c" "i"
> x <- c("abcdef", "ghifjk")
> str_sub(x, 3, 3)
[1] "c" "i"
> str_sub(x, 3, 5)
[1] "cde" "ifj"
```

STRINGR STR_SUB

Use to replace.

Here, I am replacing the comma in 100,000 with blank to give 100000.

I am replacing Jo in John with RO.

```
571  ##Use str_sub to modify strings
572  y <- c("John", "100,000")
573  str_sub(y[2], 4,4) <- ""
574  y
575  str_sub(y[1],1,2) <-"RO"
576  y
577
```

565:17 PACKAGES

Console ~/RStudioFolder/

```
> y <- c("John", "100,000")
> str_sub(y[2], 4,4) <- ""
> y
[1] "John"   "100000"
> y <- c("John", "100,000")
> str_sub(y[2], 4,4) <- ""
> y
[1] "John"   "100000"
> str_sub(y[1],1,2) <-"RO"
> y
[1] "ROhn"   "100000"
>
```

STRINGR: FURTHER METHODS

1) The stringr library has many methods for working with strings:

2) You can find many of them here:

https://cran.r-project.org/web/packages/stringr/vignettes/stringr.html

https://cran.r-project.org/web/packages/stringr/stringr.pdf

PLYR

Tools for splitting, applying, and combining data.

References:

https://cran.r-project.org/web/packages/plyr/plyr.pdf

https://cran.r-project.org/web/packages/plyr/index.html

http://myweb.facstaff.wwu.edu/minerb2/biometrics/plyr.html

https://mqwilber.github.io/2015-04-17-ucsb/lessons/plyr_reshape/datamanipulation.html

PLYR: DDPLY()

The **ddply()** function take a data.frame, summarizes it, and returns a data.frame to the user.

The function ddply() requires **several arguments**.

The first is the data.frame that you want to summarize.

The second is the column(s) that you want to summarize by.

DDPLY EXAMPLE:

NOTE: the ddply() function allows us to summarise specific attributes from the data.

```
> head(drugdata)
  subject sex condition before after change
1       1   F   placebo   10.1   6.9   -3.2
2       2   F   placebo    6.3   4.2   -2.1
3       3   M   aspirin   12.4   6.3   -6.1
4       4   F   placebo    8.1   6.1   -2.0
5       5   M   aspirin   15.2   9.9   -5.3
6       6   F   aspirin   10.9   7.0   -3.9
>
> # Run the functions length, mean, and sd on the value of "change" for each group,
> # broken down by sex + condition
> cdata <- ddply(drugdata, c("sex", "condition"), summarise,
+                N    = length(change),
+                mean = mean(change),
+                sd   = sd(change),
+                se   = sd / sqrt(N)
+ )
> cdata
  sex condition  N      mean        sd        se
1   F   aspirin  5 -3.420000 0.8642916 0.3865230
2   F   placebo 12 -2.058333 0.5247655 0.1514867
3   M   aspirin  9 -5.411111 1.1307569 0.3769190
4   M   placebo  4 -0.975000 0.7804913 0.3902456
>
```

DPLYR

1) **dplyr** is a package for **data manipulation,** written and maintained by Hadley Wickham. It provides some great, easy-to-use functions that are very handy when performing exploratory data analysis and manipulation.

2) The following slide contains a lot of code examples. Type these in to see what they do.

CODE EXAMPLES

```
########## Using dplyr
library(datasets)
library(dplyr)
head(airquality)
##filter
##The filter function will return all the rows that satisfy a
##following condition.
##For example below will return all the rows where
## Temp is larger than 70.
filter(airquality, Temp > 70)
##filter more than one attribute
filter(airquality, Temp > 80 & Month > 5)
##Mutate is used to add new variables to the data.
##For example lets adds a new column that displays the
##temperature in Celsius.
mutate(airquality, TempInC = (Temp - 32) * 5 / 9)
```

```
##The summarise function is used to summarise multiple values
##into a single value
summarise(airquality, mean(Temp, na.rm = TRUE))
##The group_by function is used to group data by one or
##more variables
summarise(group_by(airquality, Month), mean(Temp, na.rm =
TRUE))
##The sample function is used to select random rows from a
##table.
sample_n(airquality, size = 10)
##The count function tallies observations based on a group.
count(airquality, Month)
##The arrange function is used to arrange rows by variables
arrange(airquality, desc(Month), Day)
##The pipe operator in R, represented by %>% can be used to
##chain code together
airquality %>%
  filter(Month != 5) %>%
  group_by(Month) %>%
  summarise(mean(Temp, na.rm = TRUE))
```

NEXT TOPIC

1. Working with data and dataframes in R:
- Reshaping data in R: row-wise and column-wise joining, row manipulations, column manipulations, sorting, merging, subsetting, binning.

2. The apply() Family
3. Cleaning and preparing data in R: regular expression, updating data, removing data, dealing with missing or NA data
4. Packages in R - these will be part of various examples and will not be exhaustive.
5. Data partitioning in R: kmeans, PCA
6. Visualization in R: plot, ggplot, qplot, leaflet

KMEANS CLUSTERING IN R

K Means Clustering is an unsupervised learning algorithm that tries to cluster data based on their similarity.

Unsupervised learning means that there is no outcome to be predicted, and the algorithm just tries to find patterns in the data.

In k means clustering, **we have to specify the number of clusters** we want the data to be grouped into.

The algorithm randomly assigns each observation to a cluster, and finds the centroid of each cluster. Then, the algorithm iterates through two steps:

- Reassign data points to the cluster whose centroid is closest.
- Calculate new centroid of each cluster.

These two steps are repeated till the within cluster variation cannot be reduced any further.

KMEANS ON THE IRIS DATASET

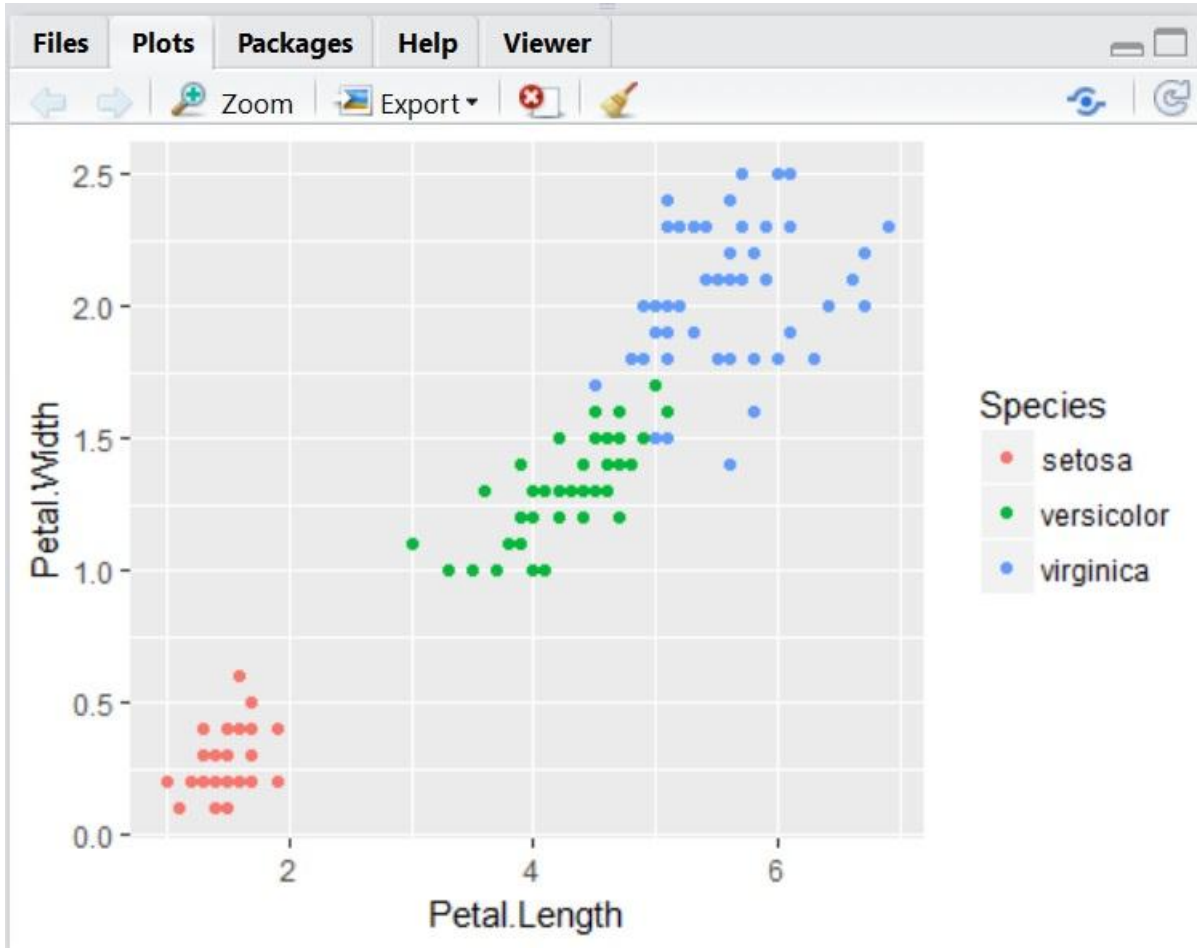
This is what the iris dataset in R looks like:

```
> 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 OF THE IRIS DATASET

```
library(ggplot2)

ggplot(iris, aes(Petal.Length,
Petal.Width, color = Species))
+ geom_point()
```



CLUSTER CODE

```
head(iris)

##plot iris using ggplot

library(ggplot2)

ggplot(iris, aes(Petal.Length, Petal.Width, color = Species)) + geom_point()

set.seed(20)

#Use "3" because there are three groups

irisCluster <- kmeans(iris, 3, nstart = 20)

irisCluster

#Compare the clusters with the species.

table(irisCluster$cluster, iris$Species)
```

CLUSTER RESULTS

NOTE: You can see that all setosa were classified correctly. Next, 48 out of 50 versicolor were classified correctly and 46 virginica were classified correctly.

```
> irisCluster
K-means clustering with 3 clusters of sizes 50, 52, 48

Cluster means:
  Petal.Length Petal.Width
1     1.462000    0.246000
2     4.269231    1.342308
3     5.595833    2.037500

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [27] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2
 [53] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3
 [79] 2 2 2 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3
[105] 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 2 3 3 3 3
[131] 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3

Within cluster sum of squares by cluster:
[1]  2.02200 13.05769 16.29167
 (between_SS / total_SS =  94.3 %)

Available components:

[1] "cluster"      "centers"      "totss"
[4] "withinss"     "tot.withinss" "betweenss"
[7] "size"         "iter"         "ifault"
> #Compare the clusters with the species.
> table(irisCluster$cluster, iris$Species)

    setosa versicolor virginica
  1     50          0         0
  2      0         48         4
  3      0          2        46
>
```

PCA IN R

The following tutorial is very good for an overview of using PCA in R:

http://www4.ncsu.edu/~slrace/LinearAlgebra2016/RChapters/PCA.pdf

Once we get to Python, I will also include a full PCA tutorial and examples for Eigenfaces.

NEXT TOPIC

1. Working with data and dataframes in R:
- Reshaping data in R: row-wise and column-wise joining, row manipulations, column manipulations, sorting, merging, subsetting, binning.

2. The apply() Family
3. Cleaning and preparing data in R: regular expression, updating data, removing data, dealing with missing or NA data
4. Packages in R - these will be part of various examples and will not be exhaustive.
5. Data partitioning in R: kmeans, PCA
6. Visualization in R: plot, ggplot, qplot, leaflet

VIS IN R

Class – on our Outline page on the class website, I will include two other PowerPoint Guides that I have created.

1) The first is on EDA (Exploratory Data Analysis) using R. This is a great review of R, basic plotting in R, and EDA.

2) The second is an extensive overview of leaflet, as well as a number of libraries and methods in action.

NOTES

At this point, you have been exposed to a lot of R.

However, R is extensive and has many libraries, etc.

Always use the Internet as a reference.

Many students (and I) like to have several R books as references as well. There is no "perfect" R book. It is a good idea to shop around or review R books at a local bookstore to see if there is one that you like.

The Internet also have many free R books as pdf and other R resources.

REFERENCES

RE: https://www.rstudio.com/wp-content/uploads/2016/09/RegExCheatsheet.pdf

http://gastonsanchez.com/Handling_and_Processing_Strings_in_R.pdf

ftp://cran.r-project.org/pub/R/doc/contrib/de_Jonge+van_der_Loo-Introduction_to_data_cleaning_with_R.pdf

http://www.cookbook-r.com/Manipulating_data/Summarizing_data/

Probability in R

http://www.stat.umn.edu/geyer/old/5101/rlook.html