KUBAA
KHULNA UNIVERSITY BGE
ALUMNI ASSOCIATION

R for Biologist-Beginner Level

Lead Mentor

Biplab Paul, PhD

Senior Scientist, Merck & Co., Inc.
Former Postdoctoral Fellow,
MGH/Harvard Medical School

Co-Lead Mentor

Mohammad Jamil Shuvo

Doctoral Researcher
Wildlife Ecology & Management
University of Freiburg, Germany

Moderator

Md Tangimul Islam

MSc (Ongoing)
Infectious Diseases and One Health
EMJMD (France, Spain, Germany)

LECTURE SCHEDULE

Lecture 1: R and R Studio Installation
Date: March 16, 2024; Time: 2:30 PM - 5:30 PM

Lecture 2: Data Types, Variables in R
Date: March 17, 2024; Time: 9:00 PM - 11:00 PM

Lecture 3: Conditionals and Loops
Date: March 23, 2024; Time: 9:00 PM - 11:00 PM

Lecture 4: Data Manipulation by Tidyverse
Date: March 24, 2024; Time: 2:30 PM - 5:30 PM

Lecture 5: Data Visualization with ggplot2
Date: March 30, 2024; Time: 2:30 PM - 5:30 PM

Lecture 6: Introductory Statistics - ANOVA
Date: March 31, 2024; Time: 2:30 PM - 5:30 PM

Lecture 7: Capstone Project Discussion
Date: April 06, 2024; Time: 2:30 PM - 5:30 PM

Lecture 8: Recap, ChatGPT, and FAQs
Date: April 07, 2024; Time: 2:30 PM - 5:30 PM

Introduction to the R Software

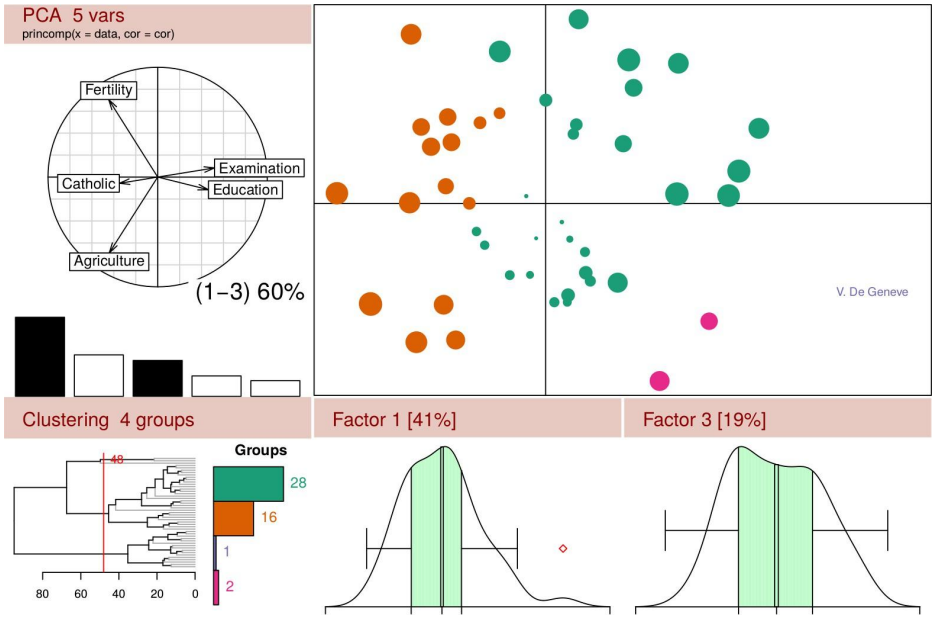
- The R system for statistical computing is an environment for data analysis and graphics.

- The root of R is the S language, developed by John Chambers and colleagues at Bell Laboratories starting in the 1960s.

- The S language was designed and developed as a programming language for data analysis tasks but in fact it is a full-featured programming language in its current implementations.

- The development of the R system for statistical computing is heavily influenced by the open source idea: The base distribution of R and a large number of user contributed extensions are available under the terms of the Free Software Foundation's GNU General Public License in source code form.

Introduction to the R Software



http://www.r-project.org

Introduction to the R Software

- Most widely used language in statistics
- Standard for biostatistical analysis
- Free, open source, available for many OS
- Efficient functions and data structures, even parallelization possible
- Extensible: base system plus user contributed packages
- From simple calculations/graphics to large custom analysis systems

Introduction to the R Software

- It is not safe: Some procedures in Excel produce inaccurate or wrong results
- Many of Excel's charts violate standards of good graphics
- Excel help can be wrong or misleading
- Excel is limited in the number of rows/columns and the size of formulas

reddit https://www.reddit.com/r/rstats/comments/8xg8f0/r_vs_excel/

Introduction to the R Software

- McCullough and Heiser (2008), pages 4570-4578:

  "... it is not safe to assume that Microsoft Excel's statistical procedures give the correct answer. Persons who wish to conduct statistical analyses should use some other package."

- Yalta (2008), pages 4579-4586:

  "researchers should continue to avoid using the statistical functions in Excel 2007 for any scientific purpose."

- October of 2020: In England nearly 16,000 coronavirus cases went unreported: In an automated process the old XLS format was used pull data together.

  But XLS is limited to 65 536 (2^{16}) rows.

  So all cases after the 65 536th were just dropped.

- Limit in R: $2^{32} - 1 = 2\ 147\ 483\ 647$

  → You will run out of memory long before hitting the limit

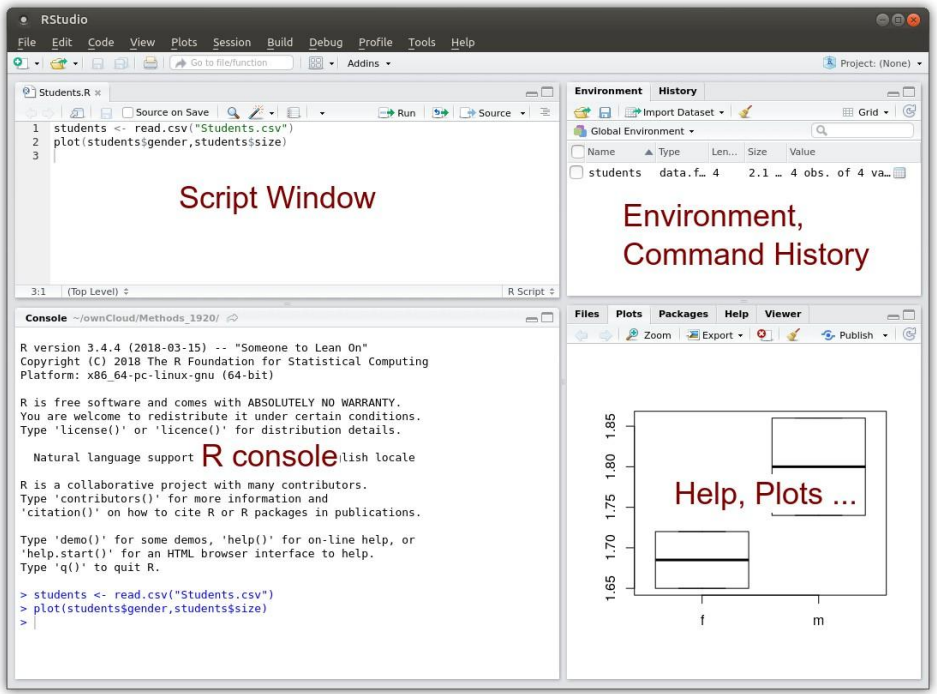
Introduction to the R Software

http://blog.ssgi.com/wp-content/uploads/2012/10/righttool2.jpg

Friends Don’t Let Friends Use Excel for Statistics!

Get the Right Tool for the Job!

Introduction to the R Software



The RStudio interface consists of **four windows:**

- **Script editing** window: The user writes here the commands to perform the analyses, saves the file with an .R extension and can come back to it later
- **Console** window: R commands are entered here to be executed (interaction with the software). Commands are compiled, from script to console, to get the results of the commands.
- **Environment** window: contains the objects in memory (including databases), which can be viewed by clicking on their names.
- **Graphics/help** window

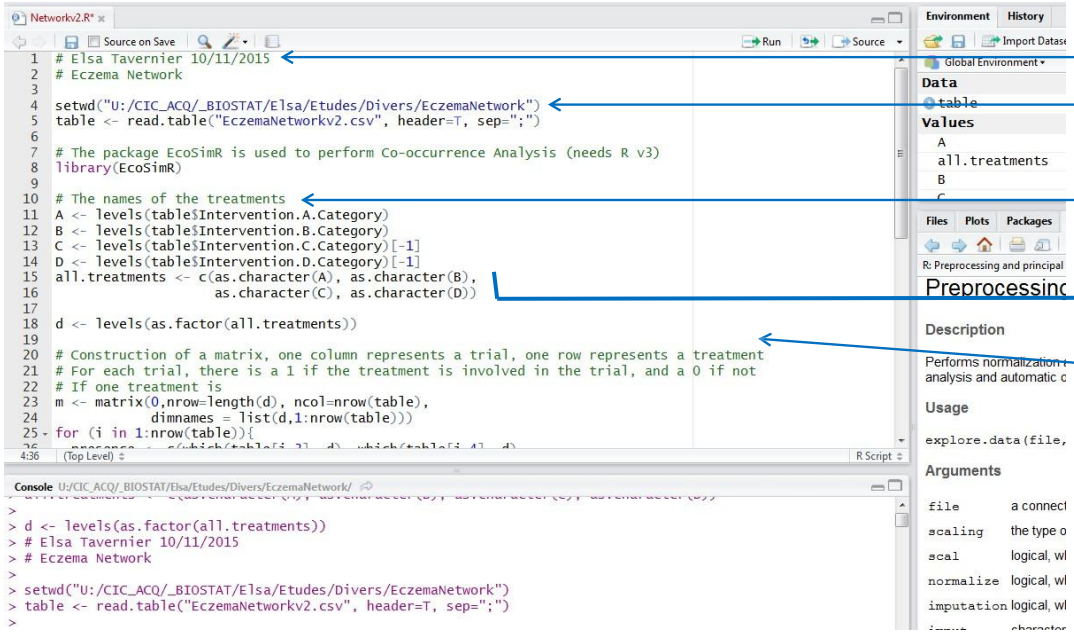
Introduction to the R Software

- Mainly www:
  - An Introduction to R:
    cran.r-project.org/doc/manuals/R-intro.html
  - Cheat sheet for basic R:
    https://cran.r-project.org/doc/contrib/Short-refcard.pdf
  - Cheat sheets for some R packages (ggplot, dplyr ...):
    rstudio.com/resources/cheatsheets/
- In R:

```
>help.start()
>help()
>help(anova)
>?anova
```

Getting Help...!!

Introduction to the R Software



Name, date, title Work file, base

import

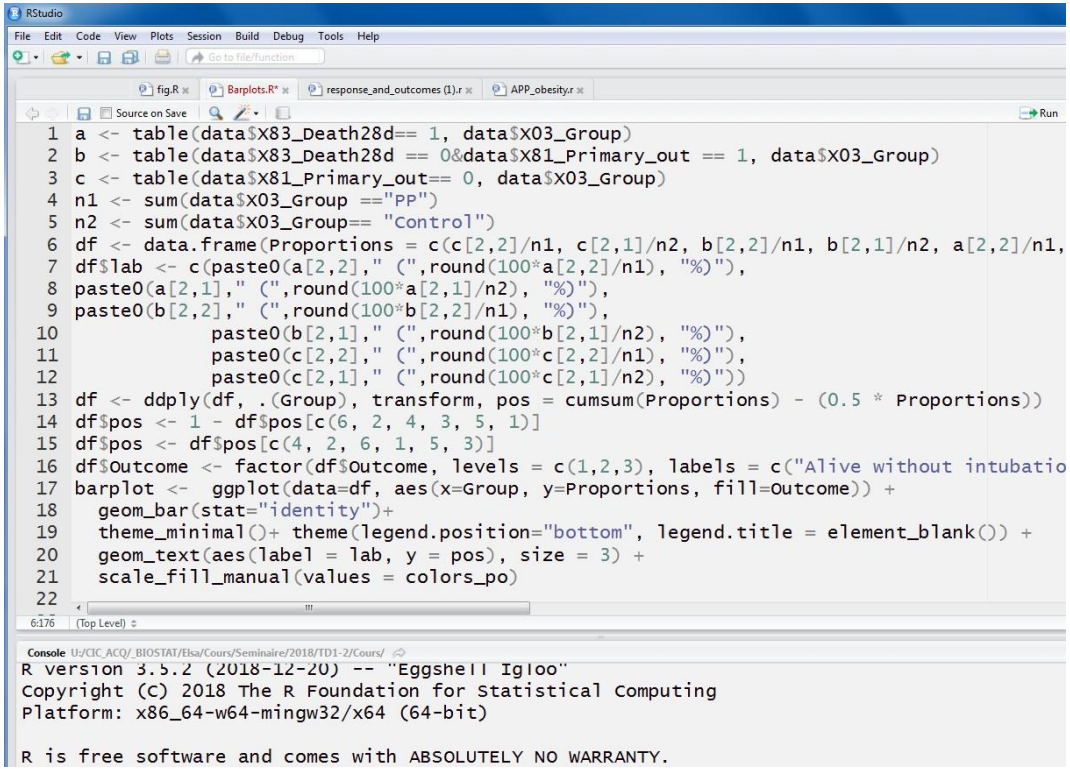
Explanatory
comments with #

Order on 2 lines, return after a comma

Line breaks between the different stages of the analysis for better readability

SCRIPT: GOOD EXAMPLE ☺

Introduction to the R Software



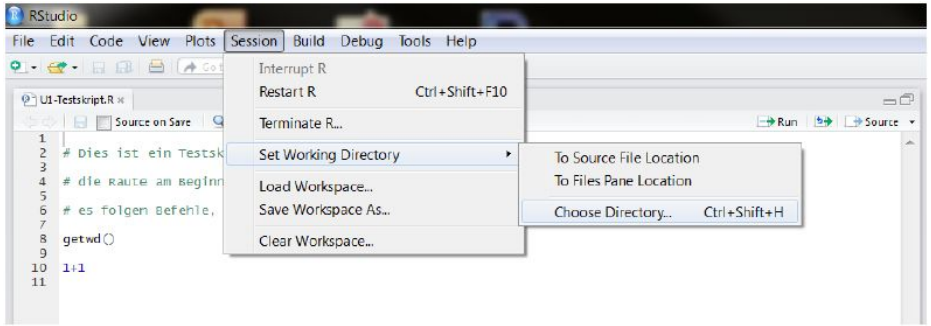
SCRIPT:
BAD EXAMPLE ☹

Introduction to the R Software

- R is case sensitive
- Commands separated by ; or newline
- Comments anywhere, started by #
- If a command is not complete, the prompt will be +
- In this way, commands can be entered across multiple lines
- Previously entered commands can be retrieved using the up-arrow key

https://www.r-bloggers.com/2021/04/the-top-10-r-errors-the-7th-one-will-surprise-you/

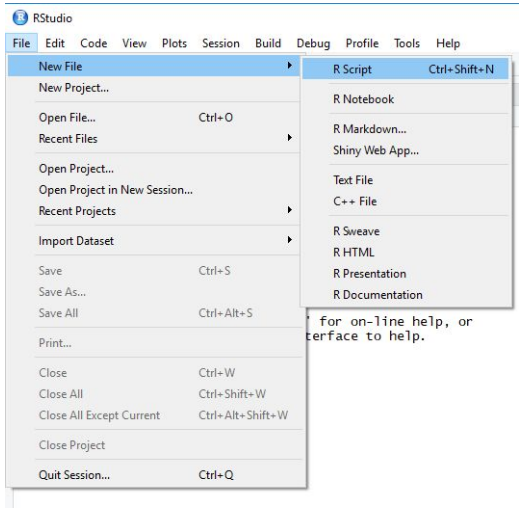
Getting started with R



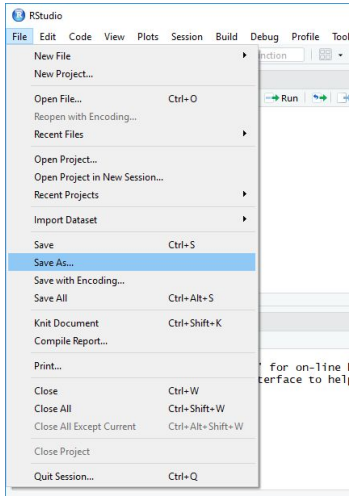
Setting the working directory

> setwd("C:/Users/Paul/Desktop")

> getwd()
[1] "C:/Users/Paul/Desktop"



Creating & Saving an R Script



Introduction to the R Software

```
> 4+5
[1] 9

> 4 * 5
[1] 20

> 4 / 5
[1] 0.8

> 4 ** 5 # or 4^5
[1] 1024

> (1 +4 ^5 )/ 25
[1] 41
```

R as calculator

+ plus (addition)
- minus (subtraction)
* times (multiplication)
/ divided by (division)
^ to the power of (exponentiation)

KUBAA
KHULNA UNIVERSITY BGE
ALUMNI ASSOCIATION

"A person who never made a mistake never tried anything new."

– Albert Einstein

Variables and Function in R

- Variable is container of data.
- The assignment operator is <−

```
# assign value 5 to variable a
> gene_length <- 1500
> gene_name <- 'ACT1'
```

- To output/print the value of a variable, just enter its name

  Print is a function that take name of variable as input and print it.
- The [1] is the row number and indicates that the following number is the first output

```
> intron_length
Error: object 'intron_length' not found
```

```
> intron = 500
> exon1 = 300
>exon2 = 700
> gene_length = exon1 + intron + exon2
```

```
> message <- "Hello from KUBAA"
> message
[1] "Hello from KUBAA"
```

Naming variable and function

variable names should

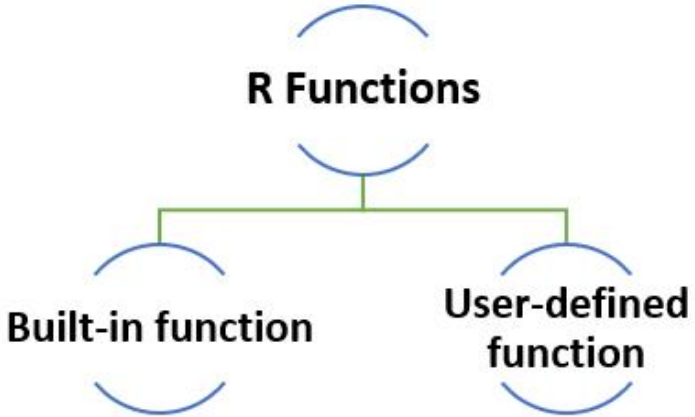
(1) consist of only letters and numbers and underscores,

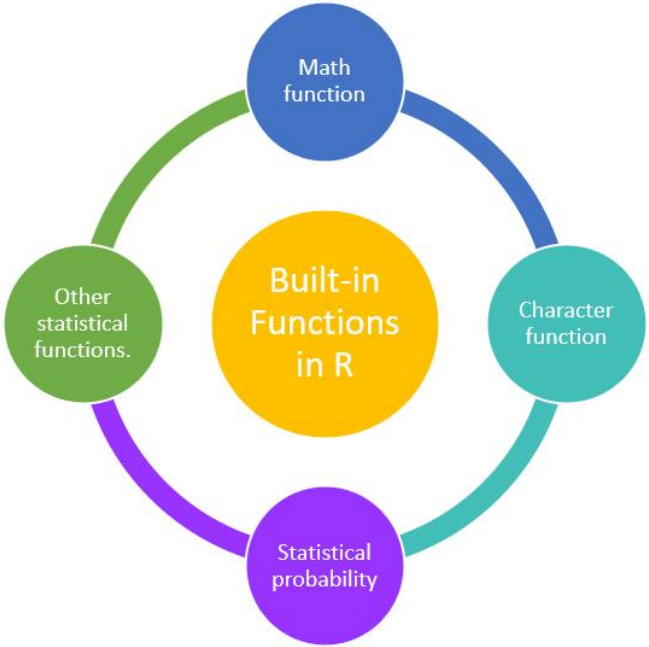
(2) start with a lowercase letter,

(3) use underscores to separate words, and

(4) be meaningful and descriptive to make code more readable.

Functions In R Programming





https://learnetutorials.com/r-programming/functions-with-examples

Data types (Class)

4 main basic data types in R
numeric, integer, character, logical

1. Numeric (include fractions value)

```
> gc_content = 50
```

2. Integer (does not include fractional value)

```
> seq_len = 300
```

3. Character (String of character specified by single or double quotes)

```
dna_seq = 'ATCGAACTTA'
```

You can convert data type

```
> seq_len_int = as.integer(seq_len)
>seq_len2 = as.numberic(seq_len)
```

When converting to an integer type, decimal parts are removed, and thus the values are rounded toward `0` (`5.7` becomes 5, and `-5.7` would become `-5`.)

Converting string to numeric: If string can not be converted to number it introduce 'NA'

4. Logical (also called boolean in other language)
Special value TRUE or T
FALSE or F

```
> sky_is_blue = TRUE
> results = 4 < 5
```

Here are the comparison produce Logical value: `<`, `>`, `<=`, `>=`, `==`, and `!=`

Data types

- In R, there is a datatype “factors”
- In statistics, an independent variable controlled by the experimenter (not necessarily categorical)
- In experiments, factors are the conditions which are being controlled. As not all values of a factor can be tested, usually specific steps are tested, thus resulting in categorical variables
- Many analyses in R require the experiment factors to be input as. . . datatype factor.

Factors

Data structures (Vectors)

Vectors are ordered collections of single types, usually numerics, integers, characters, or logicals.

```
marks = c(95,96,97,98,99,00)
marks
marks = c(90, sample, 85)
```

We can extract individual elements from a vector using [ ] syntax;
> marks1 = marks[2]

The `length()` function returns the number of elements of a vector
> length(marks)

Let's generate some vectors:
seq() function
> range = seq(1, 20, 0.5)
> range2 = 1:20

Subset a vector:
> sub_range = range[c(1, 2, 3)]
Place elements in vector:
> range[c(1, 2)] = c(10, 20)
Named vector:
marks = c(90, 95, 99)
names(marks) = c('studentA', 'studentB', 'studentC')

Vectorize operations

all of the functions and operators works on all elements of vections

```
 > means(marks)
```

Recycling vectors:

```
 > value = c(2, 4, 3, 5)
 > multi = c(2, 3)
 > results = value * multi
```

Replace value with logical vector:

```
 > value = c(10, 24, 37, 29, 34,  40, 50)
 > select_vec = value > 35
 > value[select_vec] = 0
 > print(value)
```

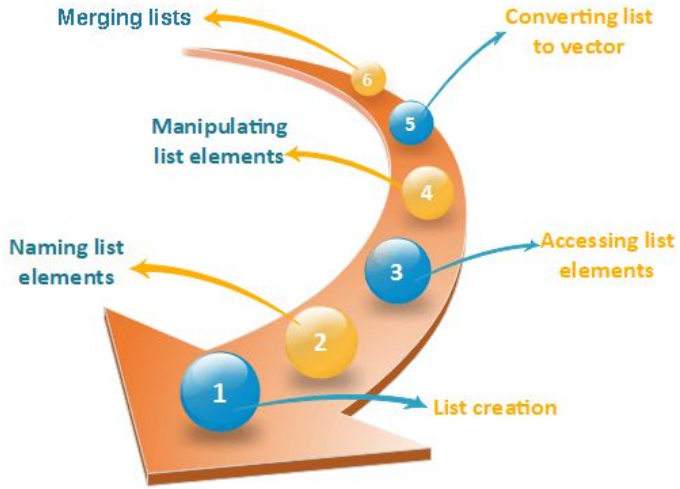
Function on vector:

```
 > order(value)
 > unique(value)
 > rev(value)
 > sample(value)
```

Lists In R Programming

- Lists are quite similar to vectors—they are ordered collections of data, indexable by index number, logical vector, and name (if the list is named).
- Lists, however, can hold multiple different types of data (including other lists).
- Lists are an excellent container for general collections of heterogeneous data in a single organized "object."
- ```
  # Create a list containing strings,
  # numbers, vectors and a logical
  values.
  list_data <- list(c("Red", "Green"),
                    c(21,32,11),
                    C(TRUE, FALSE))

  print(list_data)
  ```



https://www.javatpoint.com/r-lists
  ```

Data structures (Matrix)

In vectors, all data are in one row. In matrices, data are arranged in rows and columns.

```
> vec1 = 1:20
> mtx = matrix(vec1, n=4, byrow=TRUE)
```

Vectors as well as matrices can contain EITHER numbers (= numeric) OR letters (= character). If you mix, R converts the numbers into character elements:

```
abc.123 <- matrix(c("a", "b", "c",
                    1, 2, 2),
                    nrow = 2,
                    byrow = T)
abc.123
```

Operation on matrix:
Number of now and columns:

```
> nrow(mtx)
> ncol(mtx)
```

DMAS (Division, Multiplication, Addition, Subtraction)

Data structures (Data frames)

The function data.frame() combines vectors to a data frame. The vectors can have different types but must have the same length.

```
fieldtrial <- data.frame(
   plot = c( 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12) ,
   variety = c( 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2) ,
   fertilizer = c("N","N","N","NK","NK","NK","N", "N","N","NK", "NK","NK"),
   rep = c( 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3) ,
   yield = c(80, 90, 95, 95, 100, 93, 88, 102, 92, 100, 110, 103) )

fieldtrial
```

Still looking for fun! 😎

It is possible to access the variables in data frames in different ways.

```
str(fieldtrial)
fieldtrial$yield
fieldtrial["yield"]
fieldtrial[,5]
fieldtrial[3,]
```

Example from genomics

```
chr <- c("chr1", "chr1", "chr2", "chr2")
strand <- c("-","-","+","+")
start<- c(200,4000,100,400)
end<-c(250,410,200,450)
mydata <- data.frame(chr,start,end,strand)
#change column names
names(mydata) <- c("chr","start","end","strand")
mydata # OR this will work too
```

Data structures (Objects)

- Variables or data structures are objects that consist only of data
- Functions are objects that consist only of program code
- S3 objects and S4 objects are often the result of carrying out analyses. You carry out an analysis with a data set and get back an object with the results. The object contains the results of the analysis and in addition program code that can plot or print the results of the analysis

Saving and loading data files

If you do not specify a path, the data are written into the working directory.

```
write.table(fieldtrial, "fieldtrial.txt")  # variable name, "filename"

### Alternatively
write.csv(fieldtrial, file = "fieldtrial2.csv")
```

The files can be read in with the function read.table() and the results should be assigned to a variable.

```
x <- read.table("fieldtrial.txt")
x

x2 <-read.table("fieldtrial2.csv",  #name of input file
                header = T,       #data have headers
                sep = ",")        #separator between values
x2
```

?read.table

Cleaning up

```
rm(list = ls())
```

Generate a Random DNA sequence

```
DNA <- c("A","G","T","C")
chain <- sample(DNA, 1000, replace = T)
paste(chain, collapse="")

GCcontent <- sum(chain %in% c('G','C'))/length(chain)
```

KUBAA
KHULNA UNIVERSITY BGE
ALUMNI ASSOCIATION

"Hope is a good thing,

Maybe the best of things,

And no good thing ever dies."

– The Shawshank Redemption (1994)

Procedural programming

- When we're programming in R (or any other language, for that matter), we often want to control when and how particular parts of our code are executed.
- We can do that using control structures like if-else statements, for loops, and while loops.
- Control structures are blocks of code that determine how other sections of code are executed based on specified parameters.
- Control structures set a condition and tell R what to do when that condition is met or not met.

Function

To turn this into a function you need three things:

1. A **name**. Here we'll use `rescale01` because this function rescales a vector to lie between 0 and 1.

2. The **arguments**. The arguments are things that vary across calls and our analysis above tells us that we have just one. We'll call it `x` because this is the conventional name for a numeric vector.

3. The **body**. The body is the code that's repeated across all the calls.

Then you create a function by following the template:

```
name <- function(arguments) {
  body
}
```

User defined function

```
sqSum<-function(x,y){
result=x^2+y^2
return(result)
}
# now try the function out
sqSum(2,3)
```

Write a function for GC content:

Conditionals:

- Conditionals are tests that return TRUE or FALSE
- equality:

```
> 4 + 4 == 8
[1] TRUE
```

- also for characters:

```
> month.abb[1] == "Jan"
[1] TRUE
```

- Equality of characters:

```
> y <- c("h", "e", "l", "l", "o")
> y == "l"
[1] FALSE FALSE TRUE TRUE FALSE
```

- others:

```
<    >       <=      >=      !=
```

Conditionals

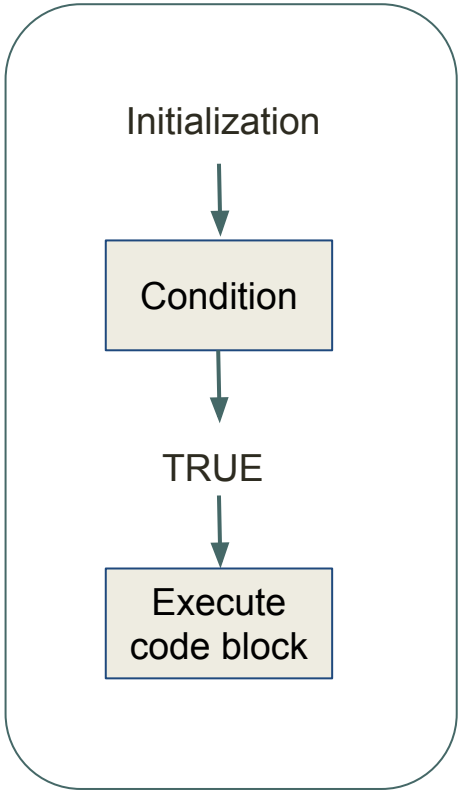
```
if (condition) {
  # code executed when condition is TRUE
} else {
  # code executed when condition is FALSE
}

if (this) {
  # do that
} else if (that) {
  # do something else
} else {
  #
}
```

Understanding If-Else in R

```
If (condition) {
    Code block
}
```

```
marks = 45
if (marks > 40) {
    print('pass')}
```

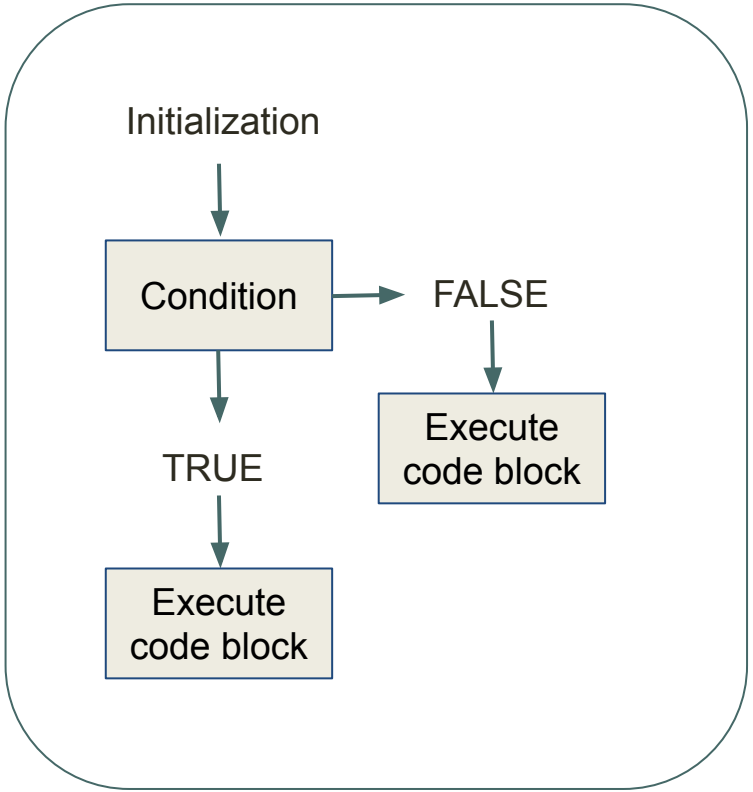


Adding the else Statement

```
If (condition) {
    code block
} else {
    code block
}
```

```
marks = 45
if (marks > 40) {
    print('pass')} else {
    print('failed')}

x <- 0
if (x < 0) {
  print("Negative number")
} else if (x > 0) {
  print("Positive number")
} else
  print("Zero")
```



Using the for loop in R

Now that we've used if-else in R to display the pass fail status of many student?

Let's say we have a vector of marks of many students.
Marks = c(40, 20, 14, 50, 45, 90)

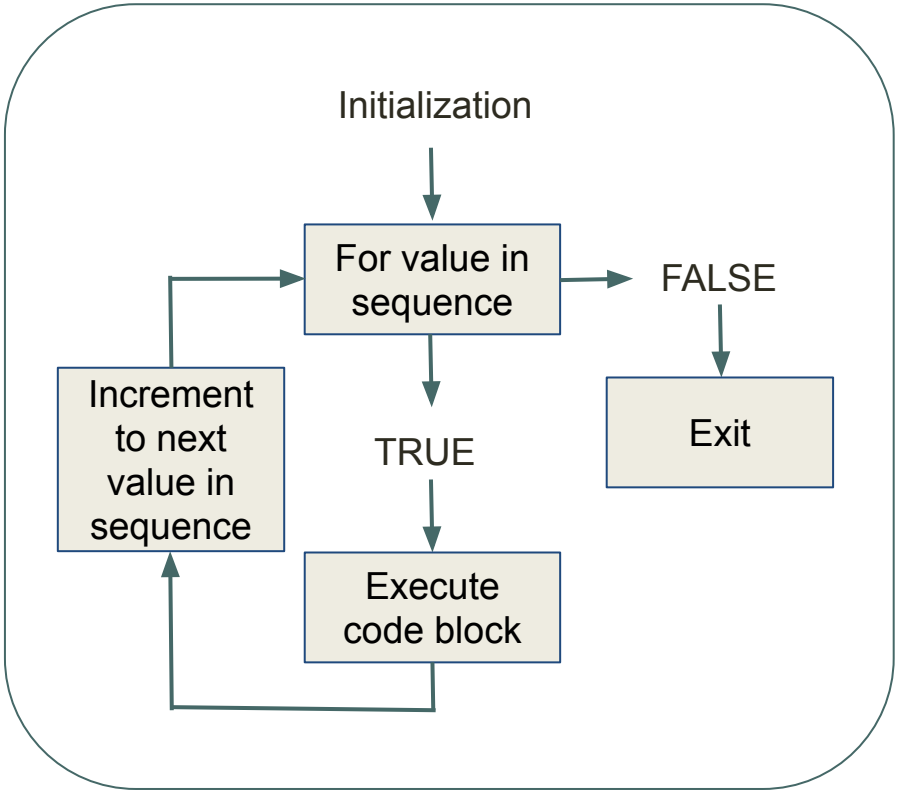
```
If (marks[1] > 40){
print(pass)}
else {‘failed’}
If mark[2] > 40 {
print(‘passed’)}
else {print(‘failed’)}
If (marks[3] > 40){
print(pass)}
else {‘failed’}
If mark[4] > 40 {
print(‘passed’)}
else {print(‘failed’)}
If (marks[5] > 40){
print(pass)}
else {‘failed’}
If mark[6] > 40 {
print(‘passed’)}
else {print(‘failed’)}
```

Using the for loop in R

What if we had a list of 100 or 1000 accounts to evaluate?

```
for (value in sequence) {
      code block
}
```

```
for ( value in marks){
      If (value > 40) {
print('pass') else {
print('failed')
}
```



If-Else Statements and Loops in R

Adding the Results of a Loop to an Object

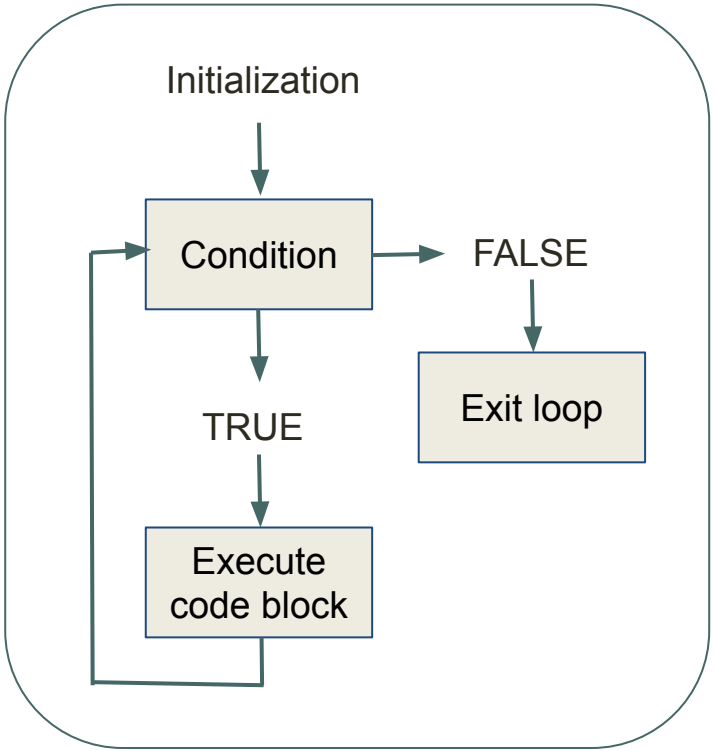
```
marks = c(23, 46, 34, 98)
results = c()
for (value in marks){
  if (value > 40){
    results = c(results, 'passed')
  } else {
    results = c(results, 'failed')
  }
  }
```

```
ifelse(marks > 40, 'passed', 'failed')
```

Using a while loop in R

- A while loop in R is a close cousin of the for loop in R.
- However, a while loop will check a logical condition, and keep running the loop as long as the condition is true.

```
while (condition) {
        expression
}
```



Using a while loop in R

Let's take a batch that's starting the TikTok with zero follower. They'll need to get 10 followers to qualify for the promotion.

```
followers <- 0
while (followers < 10){
  print ("Does not qualify")
  followers <- followers + 1
}
```

```
followers  <- 0
while (followers  <= 10){
  if (followers  < 10){
    print("Does not qualify")
  } else {
    print ("Qualified")
  }
  followers  <- followers  + 1
}
```

Using an if-else Statement within a while loop

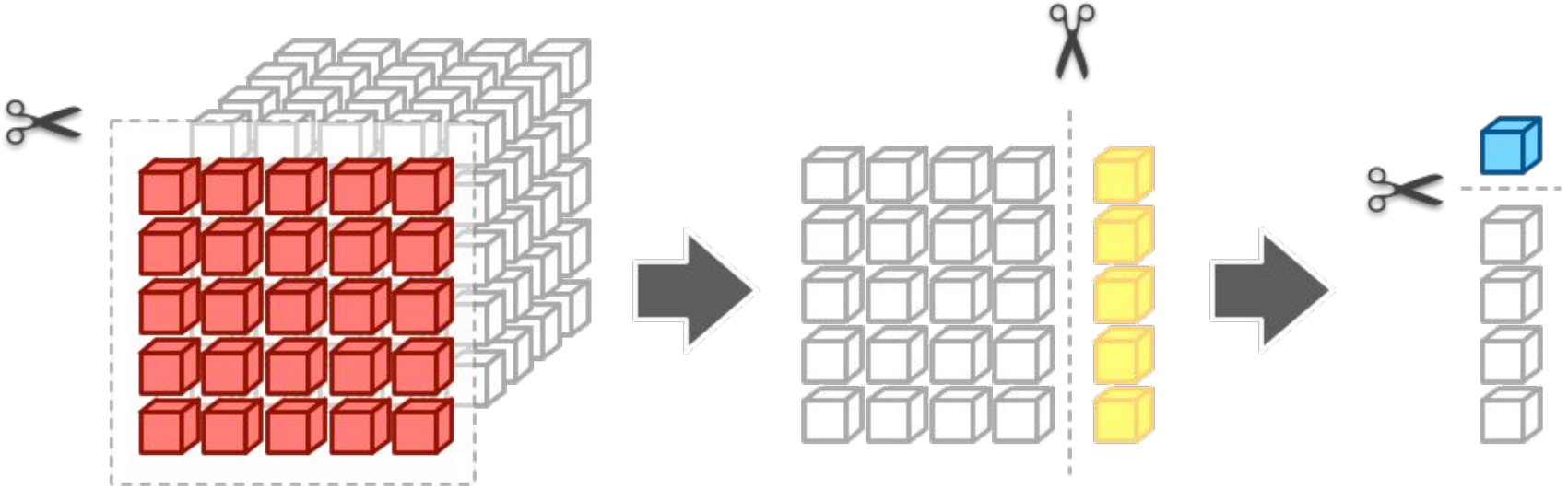
Breaking the for loop

```
followers  <- 0
while (followers  <= 10){
  if (followers  < 10){
    print("Does not qualify")
  } else {
    print ("Qualified")
    break
  }
  followers  <- followers  + 1
}
```

"The best way to predict your future is to create it."

– Abraham Lincoln

Data manipulations in R



Dplyr/tidyverse package

There are 8 fundamental data manipulation verbs that you will use to do most of your data manipulations. These functions are included in the dplyr/tidyverse package:

- filter(): Pick rows (observations/samples) based on their values.
- distinct(): Remove duplicate rows.
- arrange(): Reorder the rows.
- select(): Select columns (variables) by their names.
- rename(): Rename columns.
- mutate() and transmutate(): Add/create new variables.
- summarise(): Compute statistical summaries (e.g., computing the mean or the sum)

Data manipulations in R

R package and dataset input

```
install.packages("tidyverse")   #install package
library("tidyverse")            #load library

data("iris")                    #dataset input

head(iris)                      #Inspect dataset
str(iris)
```

This famous (Fisher's or Anderson's) **iris data** set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

How this iris dataset look like

	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

Select data by its column name

selected <- select(iris, Sepal.Length, Petal.Length) head(selected)	#To select the following columns
selected1 <- select(iris, Sepal.Length:Petal.Length)	#To select all columns from Sepal.Length to Petal.Length
head(selected1, 4)	#To print first four rows
selected1 <- select(iris,c(3:5)) head(selected1)	#To select columns with numeric indexes
selected <- select(iris, -Sepal.Length, -Sepal.Width) head(selected)	#We use(-)to hide a particular column

Finding rows with matching criteria

filtered <- filter(iris, Species == "setosa") head(filtered,3)	#To select the first 3 rows with Species as setosa
filtered1 <- filter(iris, Species == "versicolor", Sepal.Width > 3) tail(filtered1)	#To select the last 5 rows with Species as versicolor and Sepal width more than 3

Finding rows with matching criteria

Output:

Sl. No.	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4	6.3	3.3	4.7	1.6	Versicolor
5	6.7	3.1	4.4	1.4	Versicolor
6	5.9	3.2	4.8	1.8	Versicolor
7	6.0	3.4	4.5	1.6	Versicolor
8	6.7	3.1	4.7	1.5	Versicolor

Creates new columns

#To create a column “Greater.Half” which stores TRUE if given condition is TRUE

```
col1 <- mutate(iris, Greater.Half = Sepal.Width > 0.5 * Sepal.Length)
tail(col1)
```

#To check how many flowers satisfy this condition

```
table(col1$Greater.Half)
```

Sort rows by variables

```
#To arrange Sepal Width in ascending order
arranged <- arrange(col1, Sepal.Width)
head(arranged)

#To arrange Sepal Width in descending order
arranged <- arrange(col1, desc(Sepal.Width))
head(arranged)
```

How this iris dataset look like

	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

Group observations

Sl. No.	Species <fct>	Mean.Sepal <dbl>
1	setosa	3.43
2	versicolor	2.77
3	virginica	2.97

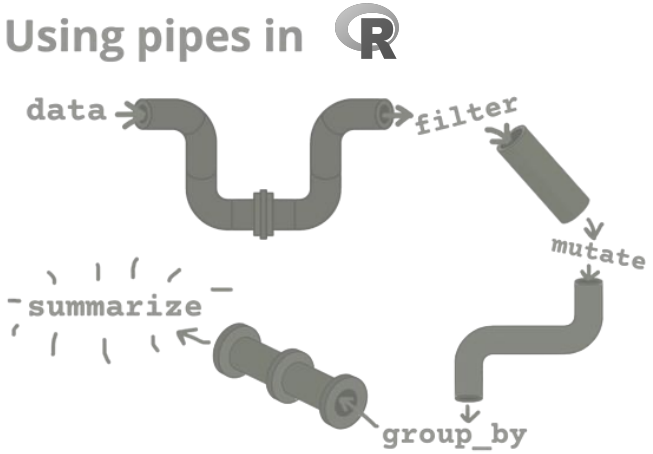
```
#To find mean sepal width by Species, we use grouping as follows
gp <- group_by(iris,Species)
mn <- summarise(gp,Mean.Sepal = mean(Sepal.Width))
head(mn)
```

Wrap multiple functions together

```
#To get rows with the following conditions
iris %>% filter(Species == "setosa",Sepal.Width > 3.8)

#To find mean Sepal Length by Species
#we use pipe operator as follows

iris  %>% group_by(Species) %>%
  summarise(Mean.Length = mean(Sepal.Length))
```



KUBAA
KHULNA UNIVERSITY BGE
ALUMNI ASSOCIATION

"Today we don't have any motivational quotes, if you want to give up, give up!"

Data visualization with R

R package and dataset input

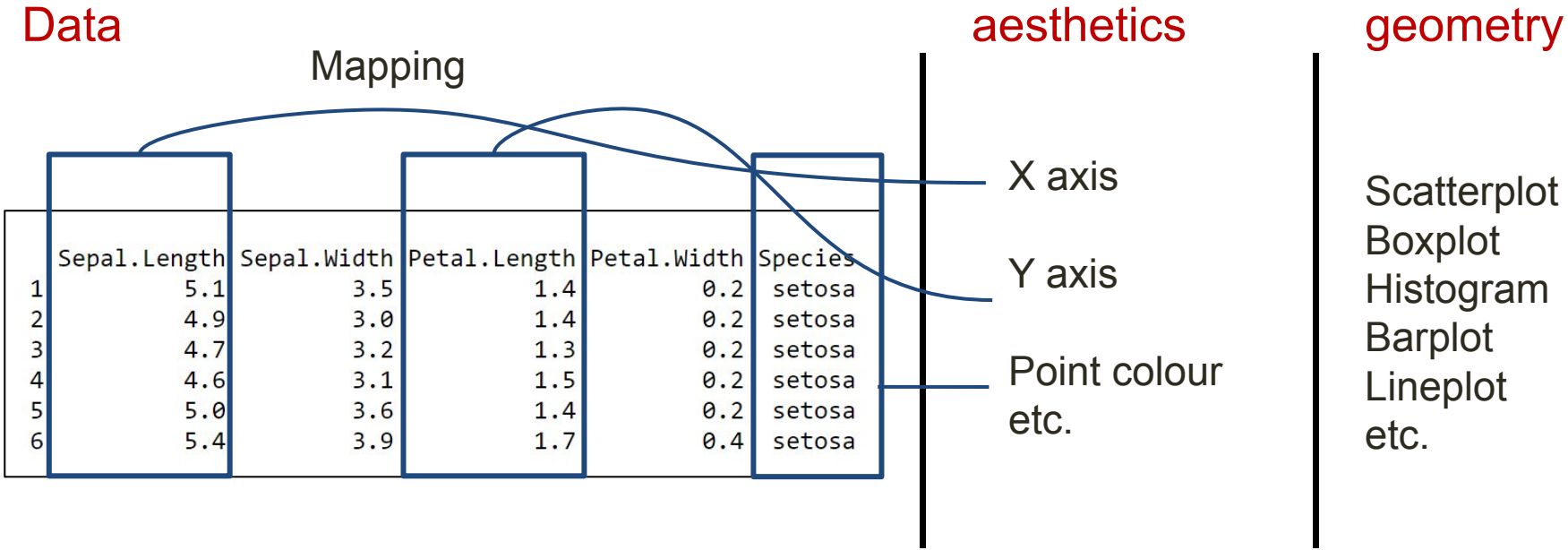
```
install.packages("ggplot2")      #install package
library("ggplot2")               #load library

data("iris")                     #dataset input

head(iris)                       #Inspect dataset
str(iris)
```

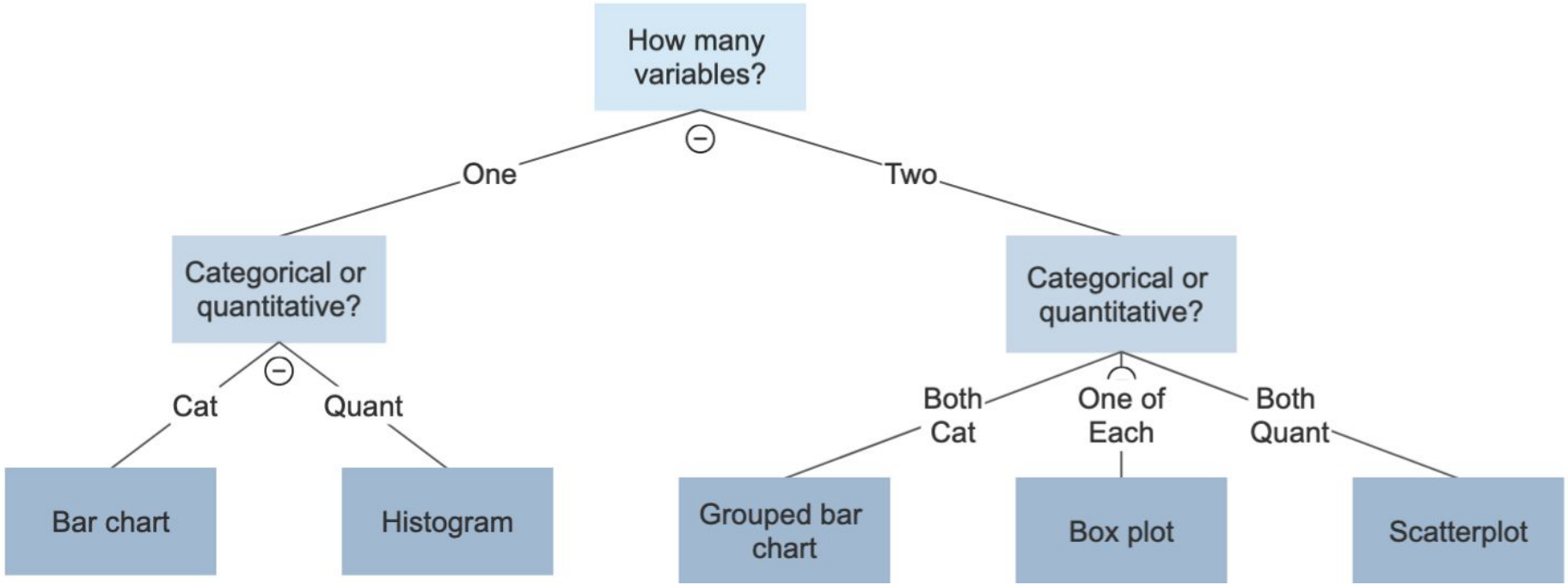
This famous (Fisher's or Anderson's) **iris data** set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.

Grammar of Graphics (ggplot2)



	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

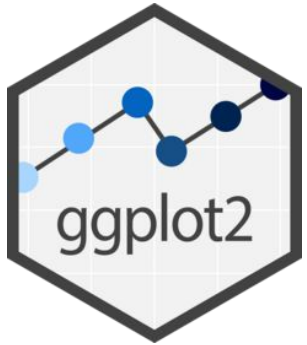
Planning a data visualization



Anatomy of a ggplot

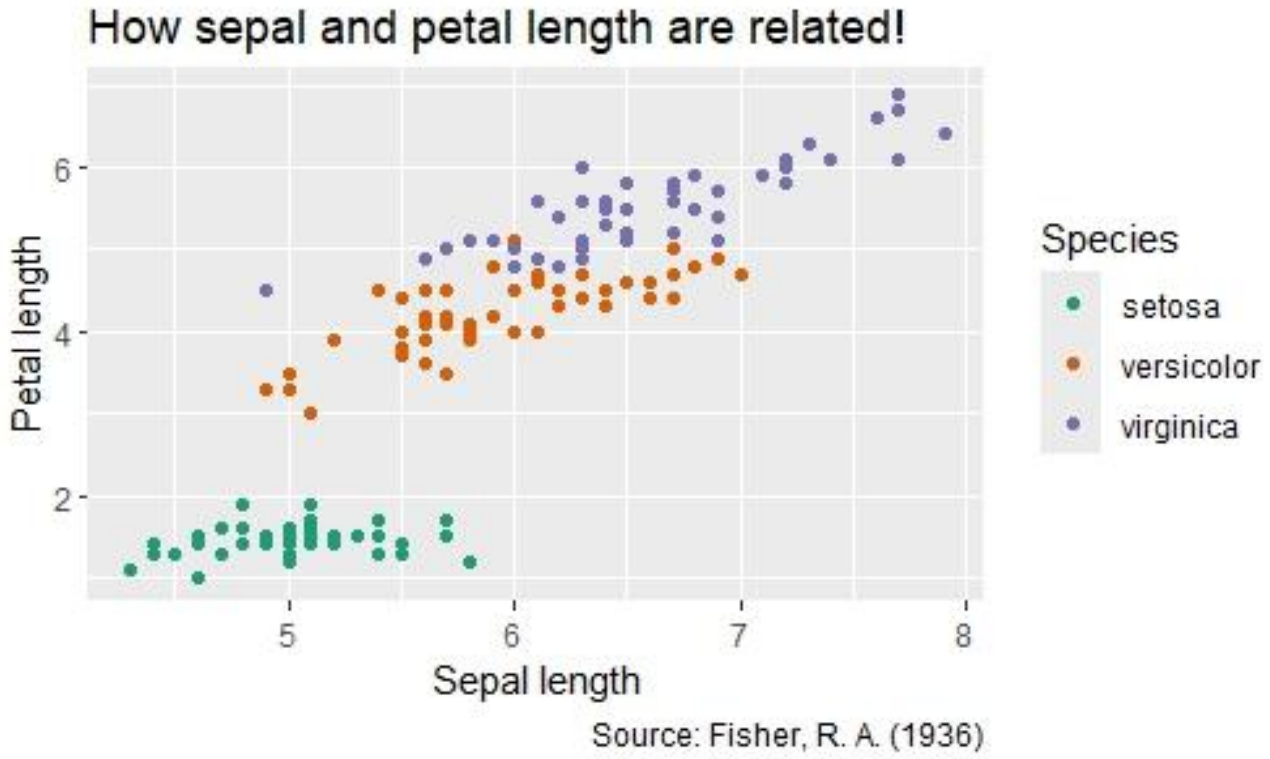
data
aesthetics
geometry
coordinates
theme

```
ggplot (iris, aes (x=Sepal.Length, y=Petal.Length))+
    geom_point(aes(color=Species))+
    coord_***()+
    theme_***()
```



https://rstudio.github.io/cheatsheets/html/data-visualization.html

Get started with ggplot



Basic Scatterplot

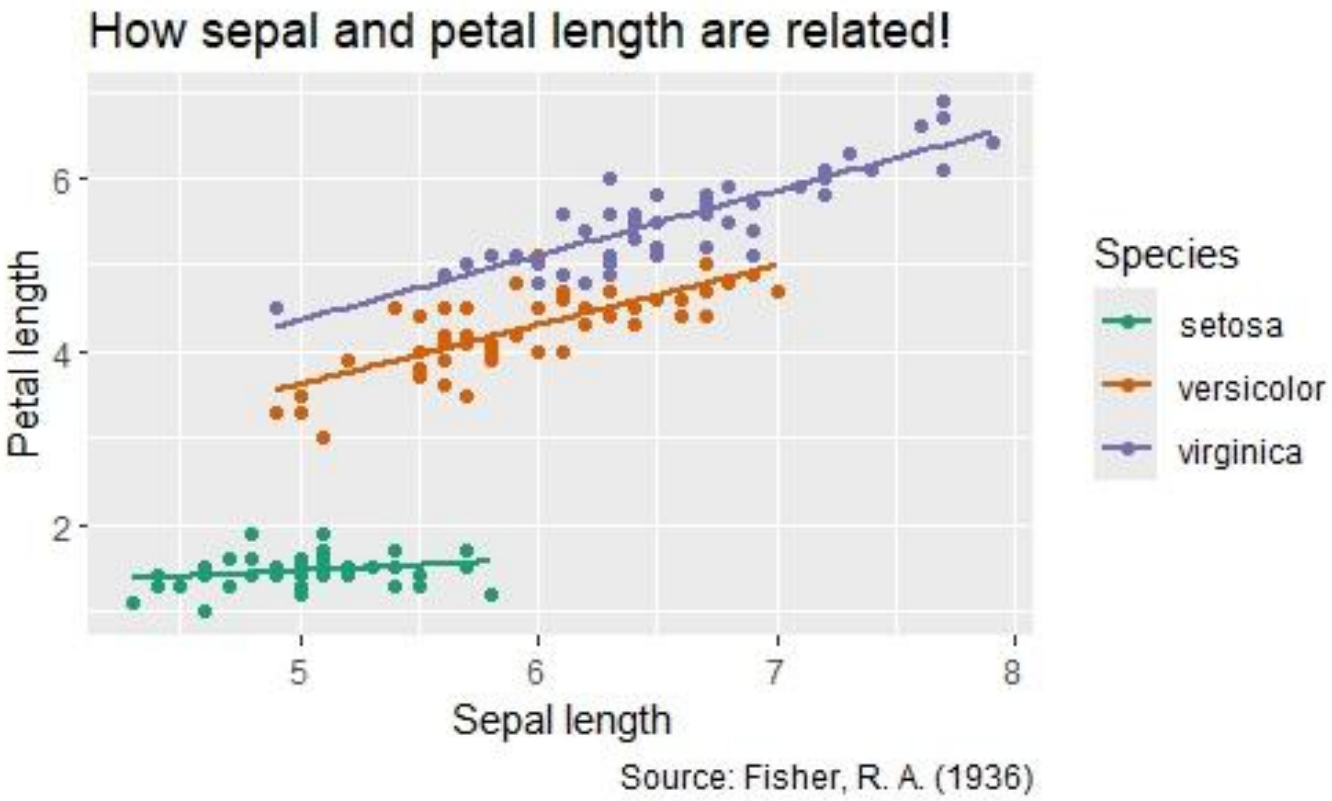
```
#basic ggplot
ggplot (iris, aes (x=Sepal.Length, y=Petal.Length, color=Species))+      #input data
  geom_point()+                                                          #define geometry
  labs(x = "Sepal length",
       y = "Petal length",                                               #add labels
       title = "How sepal and petal length are related!",                #add titles
       caption = "Source: Fisher, R. A. (1936)")+                        #citations
  scale_color_brewer(palette = "Dark2")                                  #change color
```

Modifying basic properties

?geom_point

```
ggplot (iris, aes (x=Sepal.Length, y=Petal.Length))+                    #input data
  geom_point(color = "black",
             size = 3,                                                  #modify geometry
             alpha = .5,
             shape = "square")+
  labs(x = "Sepal length",
       y = "Petal length",                                              #add labels
       title = "How sepal and petal length are related!",               #add titles
       caption = "Source: Fisher, R. A. (1936)")+                       #citations
  scale_color_brewer(palette = "Dark2")                                 #change color
```

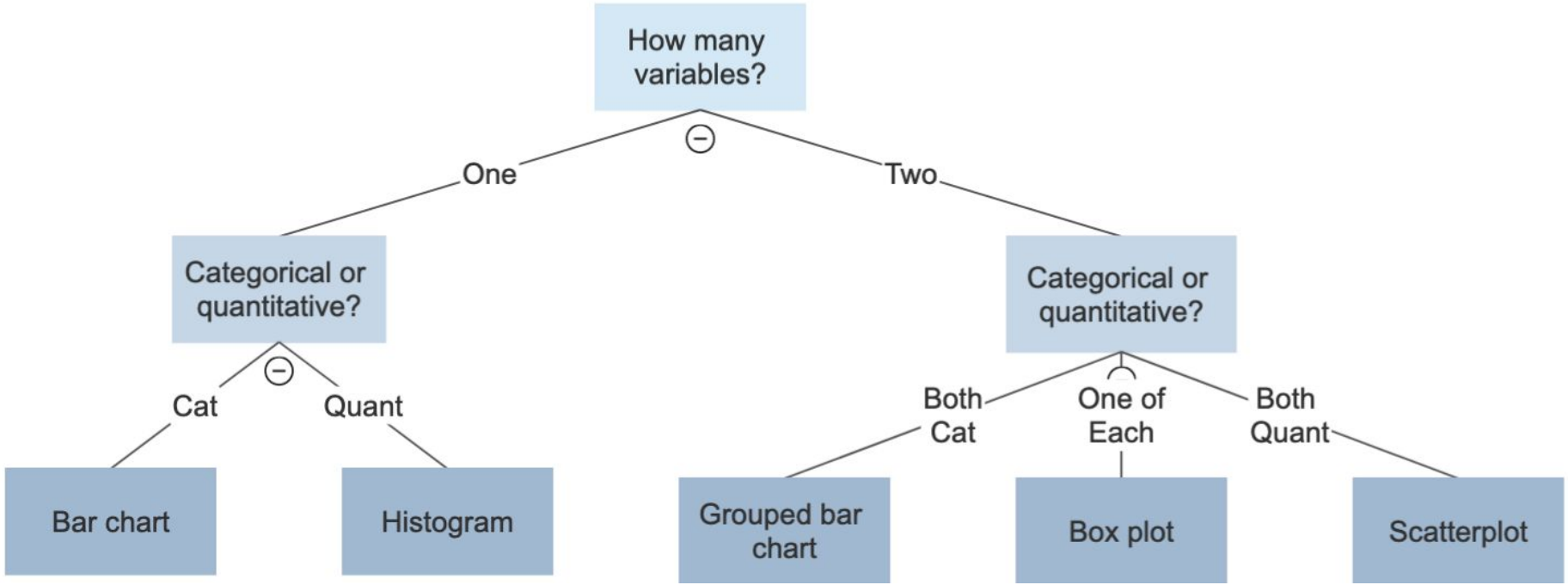
Adding another layer



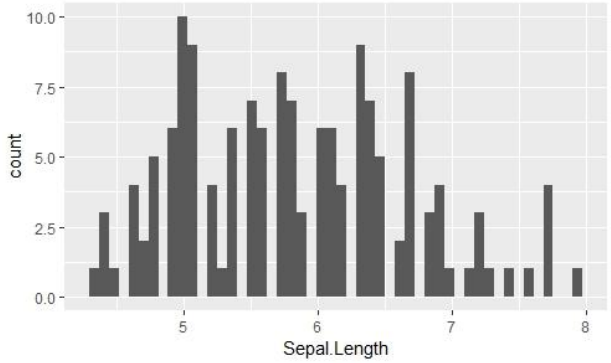
Adding another layer

```
ggplot (iris, aes (x=Sepal.Length, y=Petal.Length, color=Species))+      #input data
  geom_point()+                                                          #define geometry
  geom_smooth(method = "lm",
              se = FALSE) +
  labs(x = "Sepal length",
       y = "Petal length",                                               #add labels
       title = "How sepal and petal length are related!",                #add titles
       caption = "Source: Fisher, R. A. (1936)")+                        #citations
  scale_color_brewer(palette = "Dark2")                                  #change color
```

Planning a data visualization

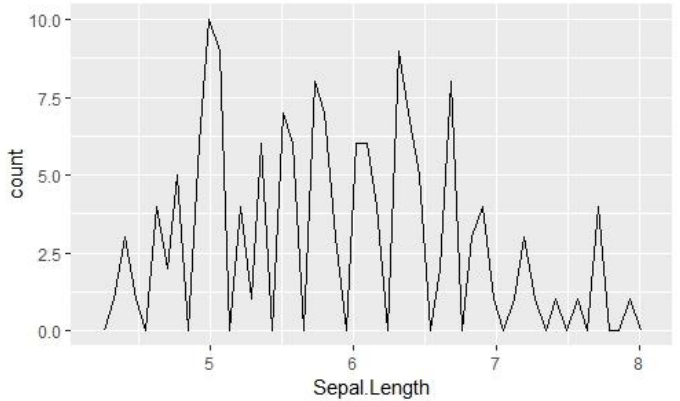


One quantitative variable

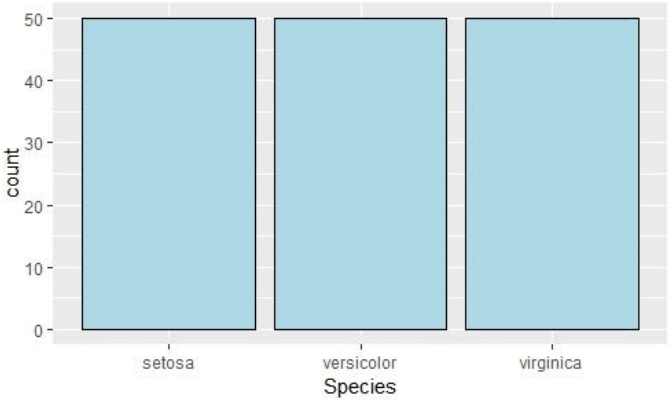



```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_histogram(bins = 50)
```

```
ggplot(iris, aes(x = Sepal.Length)) +
  geom_freqpoly(bins = 50)
```

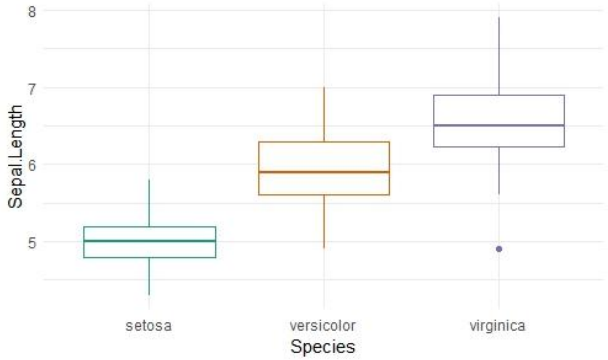


Barplot and Boxplot

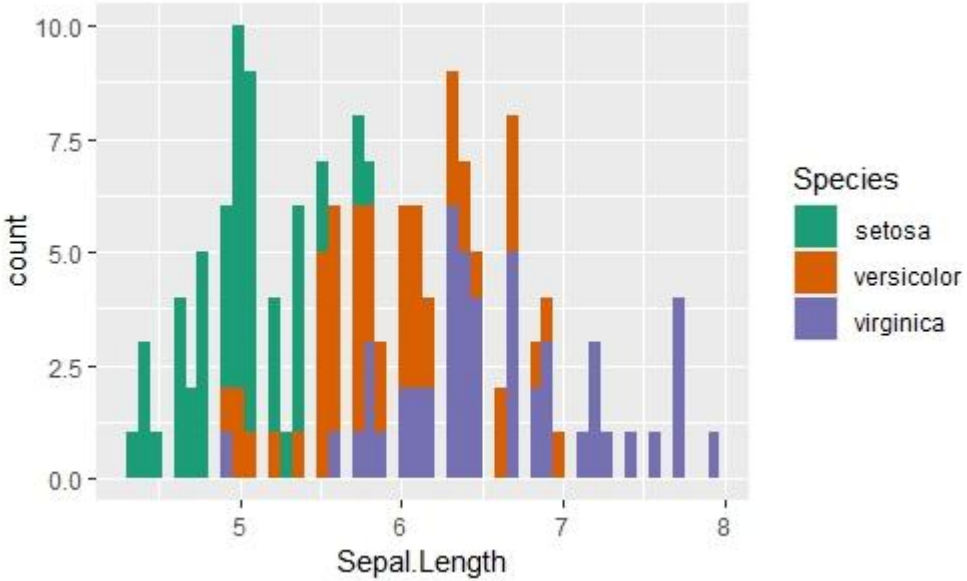



```
ggplot(iris, aes(x = Species)) +
  geom_bar(color = "black",
           fill = "lightblue")
```

```
ggplot(iris, aes(x = Species,
                 y = Sepal.Length,
                 color = Species)) +
  geom_boxplot(show.legend = FALSE) +
  scale_color_brewer(palette = "Dark2") +
  theme_minimal()
```

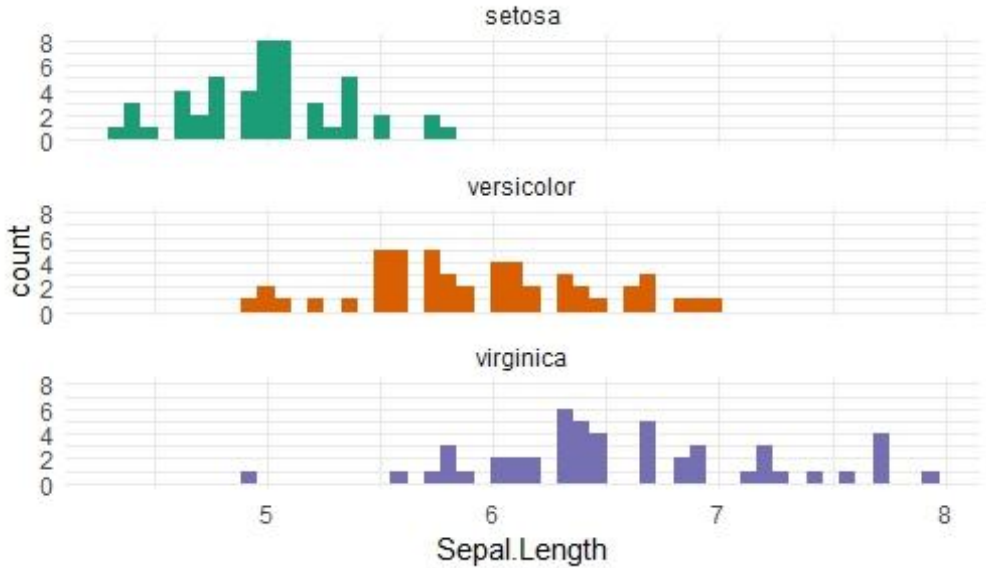


Graphs faceting



Who can interpret these graphs?

Graphs faceting

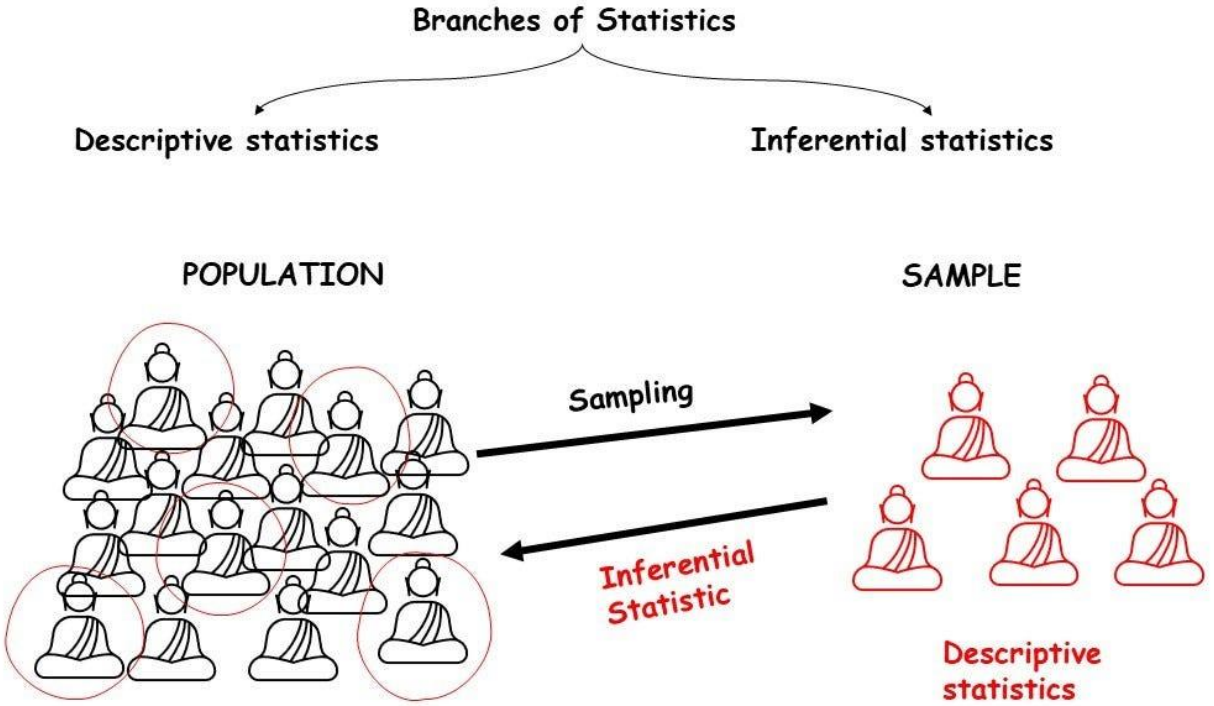



```
ggplot(iris, aes(x = Sepal.Length,
                 fill = Species)) +
  geom_histogram(bins = 50,
                 show.legend = FALSE) +
  facet_wrap(~Species,
             ncol = 1) +
  scale_fill_brewer(palette = "Dark2") +
  theme_minimal()
```

KUBAA
KHULNA UNIVERSITY BGE
ALUMNI ASSOCIATION

"If you truly love nature, you will find beauty everywhere."
–Vincent van Gogh

Introductory statistics



Descriptive statistics

https://youtu.be/5C9LBF3b65s?si=oY5-l9q8uZfXm_lf

Descriptive statistics

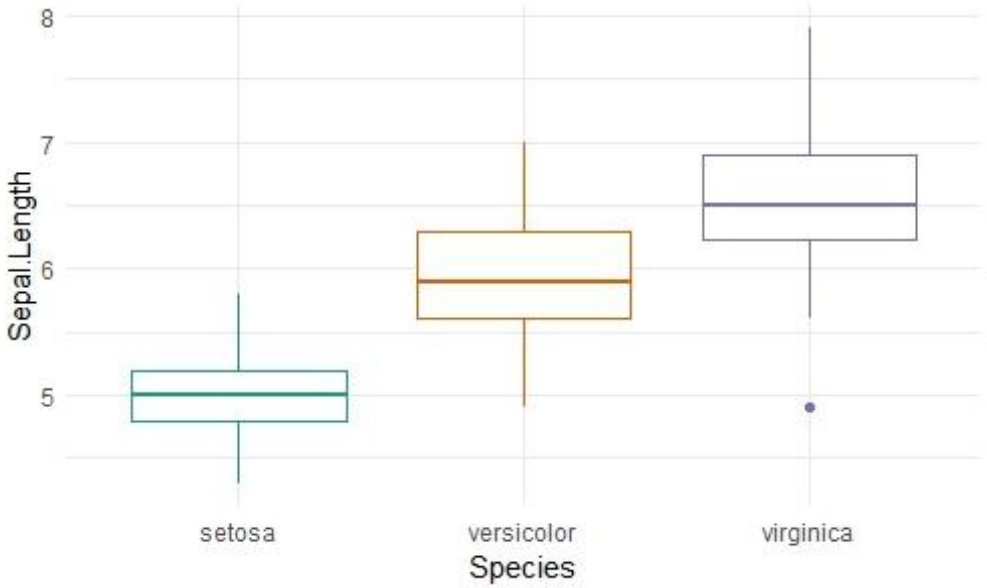
Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100	setosa:50
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.300	versicolor:50
Median :5.800	Median :3.000	Median :4.350	Median :1.300	virginica :50
Mean :5.843	Mean :3.057	Mean :3.758	Mean :1.199	NA
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800	NA
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500	NA

```
data("iris")
sum <- summary(iris)
write.csv(sum, file = "summary.csv")
```

Boxplot interpretation

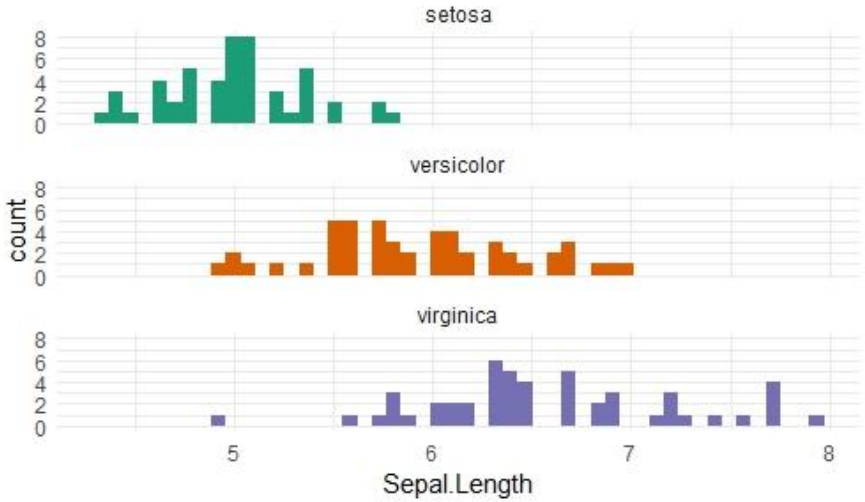
https://youtu.be/tpToLyZibKM?si=5xjg7xxGyZiuAjKy

Boxplot interpretation




```
ggplot(iris, aes(x = Species,
                 y = Sepal.Length,
                 color = Species)) +
  geom_boxplot(show.legend = FALSE) +
  scale_color_brewer(palette = "Dark2") +
  theme_minimal()
```

Histogram interpretation




```
ggplot(iris, aes(x = Sepal.Length,
                 fill = Species)) +
  geom_histogram(bins = 50,
                 show.legend = FALSE) +
  facet_wrap(~Species,
             ncol = 1) +
  scale_fill_brewer(palette = "Dark2") +
  theme_minimal()
```

Inferential Statistics

Parametric test

1. **Correlation**

   Correlation tests check whether two variables are related without assuming cause-and-effect relationships.

2. **Regression**

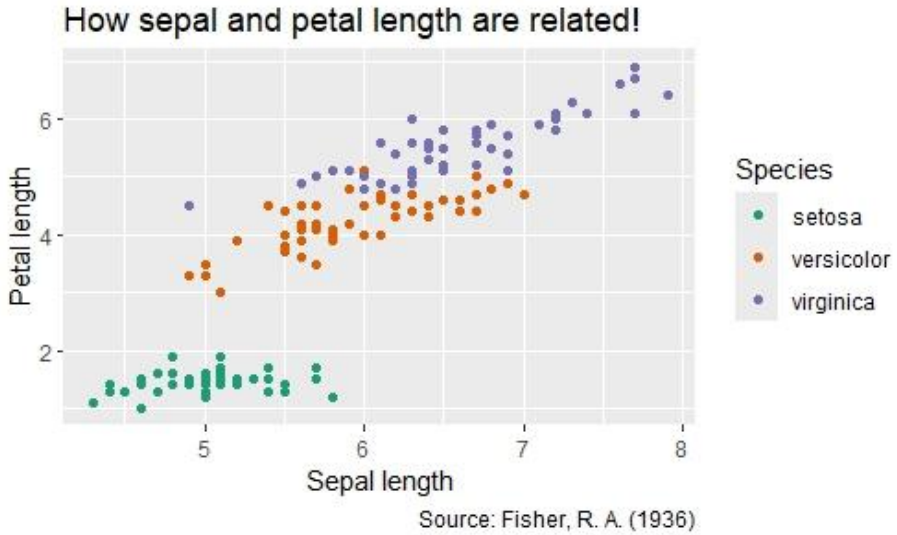
   Used to test cause-and-effect relationships

3. **Comparison**

   Comparison tests look for differences among group means.

Pearson Correlation

Pearson correlation between 2 variables
cor(iris$Sepal.Length, iris$Petal.Length, method = "pearson")



Linear Regression Model

To model the data onto a straight line using two variables, lm=linear model

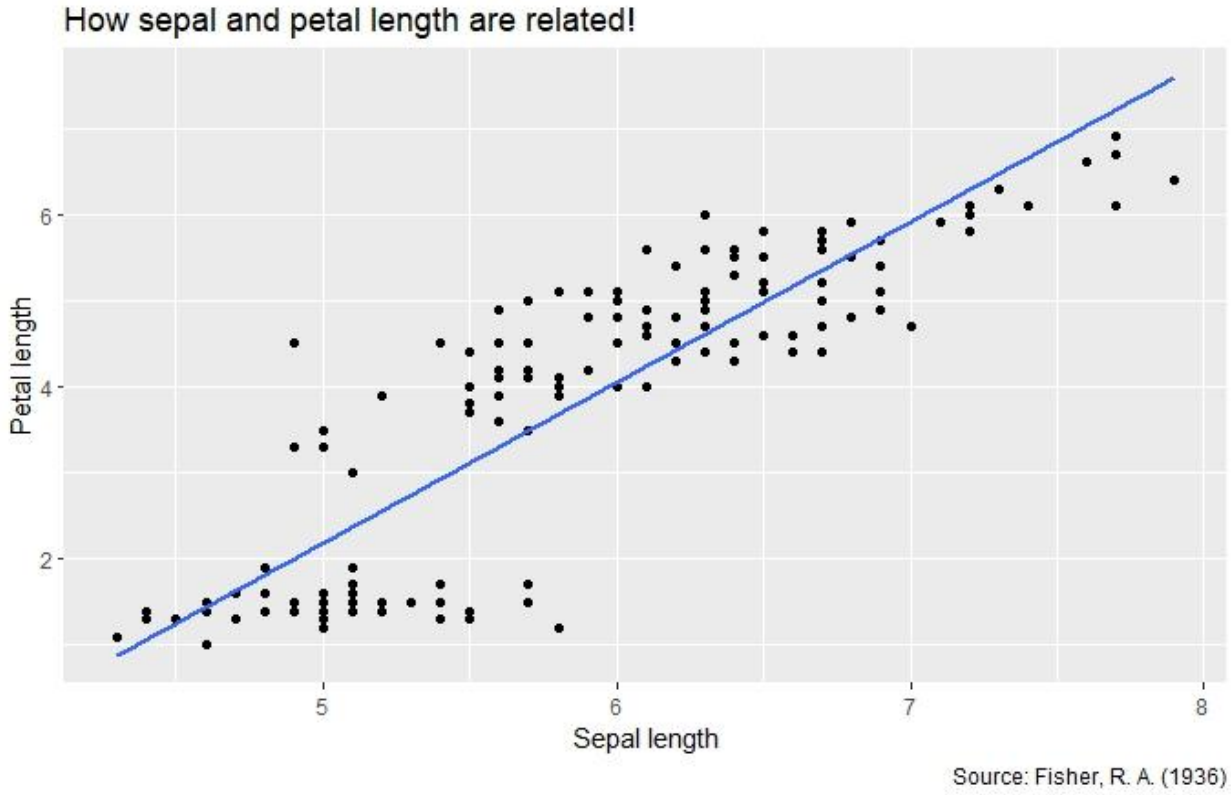
y = mx + C, (or in excel term, plot scatter graph, get trendline and show R-square value)

lm(x ~ y, data=dataset)

model <- lm(iris$Petal.Length~ iris$Sepal.Length, data=iris)

summary(model)

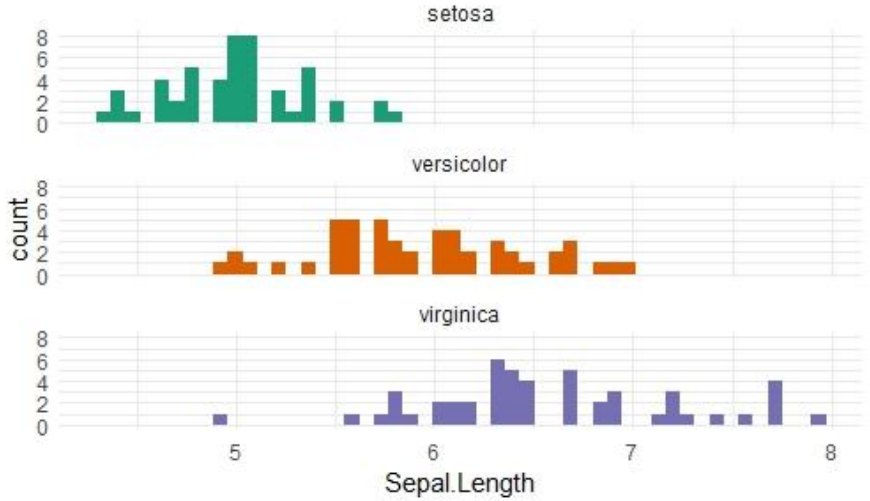
Linear Regression Model



Comparison tests

https://youtu.be/idZHQwVohOo?si=MUYUu8KXizrjkUai

Histogram interpretation



Just looking at the data for two species of irises, it looks like the Sepal Length are different, but are they *significantly* different?

Data modifications

```
setosa <- iris[iris$Species == "setosa", ]
versicolor <- iris[iris$Species == "versicolor", ]
```

We'll start by comparing the data of *Iris setosa* and *Iris versicolor*, so we need to create two new data objects, one corresponding to the *I. setosa* data and one for the *I. versicolor* data.

p-value interpretation

```
# Compare Sepal Length of these two species
  t.test(x = setosa$Sepal.Length, y = versicolor$Sepal.Length)
```

```
        Welch Two Sample t-test

data:  setosa$Sepal.Length and versicolor$Sepal.Length
t = -10.521, df = 86.538, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -1.1057074 -0.7542926
sample estimates:
mean of x mean of y
    5.006     5.936
```

Now to compare the two species, we call the `t.test` function in R, passing each set of data to `x` and `y`.

The lower the p-value, the greater the statistical significance of the observed difference. A p-value of 0.05 or lower is generally considered statistically significant.

So we reject the hypothesis that these species have the same Sepal Length.

Analysis of Variance (ANOVA)

response variables predictor variables

```
anova <- aov(Sepal.Length ~ Species, data = iris)
summary(anova)
```

```
> summary(anova)
             Df Sum Sq Mean Sq F value Pr(>F)
Species       2  63.21  31.606   119.3 <2e-16 ***
Residuals   147  38.96   0.265
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

ANOVA (one-way) allows us to simultaneously compare multiple groups, to test whether group membership has a significant effect on a variable of interest.

Post Hoc Tests – Tukey HSD

```
t1 <- TukeyHSD(anova, conf.level=0.95)
t1
```

```
> t1
  Tukey multiple comparisons of means
    95% family-wise confidence level

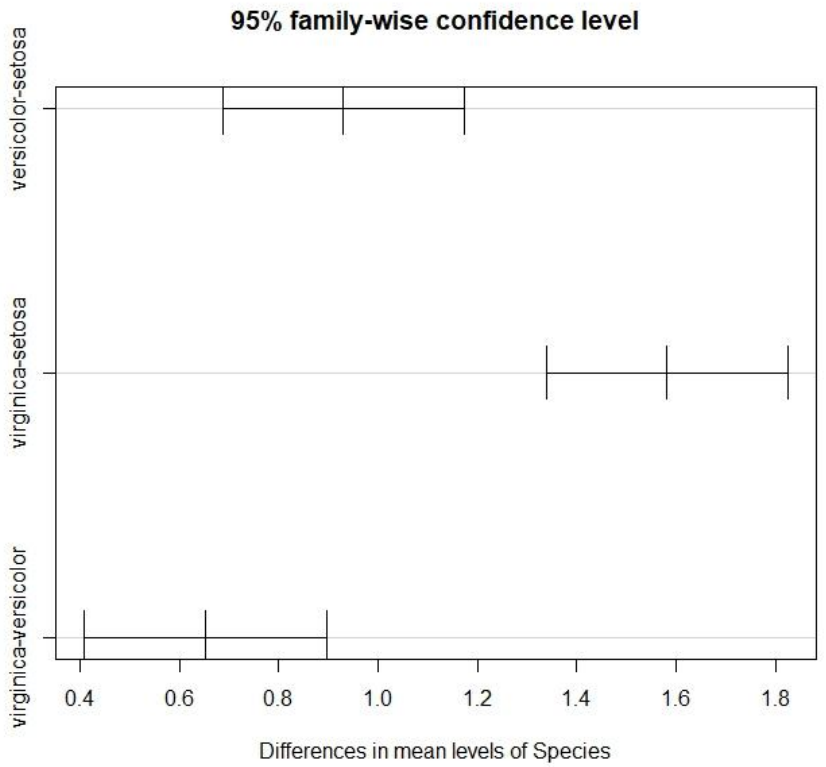
Fit: aov(formula = Sepal.Length ~ Species, data = iris)

$Species
                       diff       lwr       upr p adj
versicolor-setosa    0.930 0.6862273 1.1737727     0
virginica-setosa     1.582 1.3382273 1.8257727     0
virginica-versicolor 0.652 0.4082273 0.8957727     0
```

Tukey's Honest Significant Difference (HSD) test is a post hoc test commonly used to assess the significance of differences between pairs of group means. **Tukey HSD** is often a follow up to **one-way ANOVA**, when the F-test has revealed the existence of a significant difference between some of the tested groups.

Tukey HSD

plot(t1)



KUBAA
KHULNA UNIVERSITY BGE
ALUMNI ASSOCIATION

"Every morning we are born again. What we do today is what matters most."
–Gautam Buddha

Assignment: Exploring the Iris Dataset

The task will encourage students to explore different aspects of the Iris dataset, including data loading, visualization, manipulation, and interpretation. By answering these questions, students will gain a deeper understanding of the dataset and practice their skills in R programming using Tidyverse and ggplot2.

Task 1: Data Loading and Inspection (Data (iris), head ())

- Load the Iris dataset into R studio, how you can display the first few rows of the dataset? Write #command what did you observe after running the code?
- Calculate descriptive statistics (mean, median, min, max) for Sepal Length? What is the minimum and maximum value?

Task 2: Data Manipulation (Dplyr/Tidyverse)

- Filter the iris dataset to include only rows where Sepal. Length is greater than 5?

Task 3: Data Visualization (ggplot2)

- Create a scatter plot using ggplot2 to visualize Sepal Length vs. Sepal Width?
- Color the points based on the species of iris flowers?

Task 4: Introductory Statistics (t-test)

Assess whether there exists a notable difference in the average Sepal Length between "setosa" and "versicolor" iris flowers. Develop null and alternative hypothesis, execute a t-test, and interpret/comment the findings.

Submission Guidelines:

- Participants may submit individually or in groups.
- Each submission should include R code and any generated visualizations or tables in attachment
- Submissions must be emailed to rifat.tangimul@gmail.com by the deadline: April 15th, 2024.

Resources:

- Participants will utilize the Iris built in dataset and R programming language.
- All recorded class were provided for reference.

Outcome:

By submitting their work, participants will receive recognition in the form of a course completion certificate, validating their newly acquired skills.

Notes:

- Make sure to include all necessary libraries at the beginning of your script (e.g., library(ggplot2)).
- For additional help or clarification, refer to any online tutorials/forums or rifat.tangimul@gmail.com