

# Introduction to Data Analysis With R

Gladstone Institutes

Krishna Choudhary  
Bioinformatics Core, GIDB  
September 16, 2019

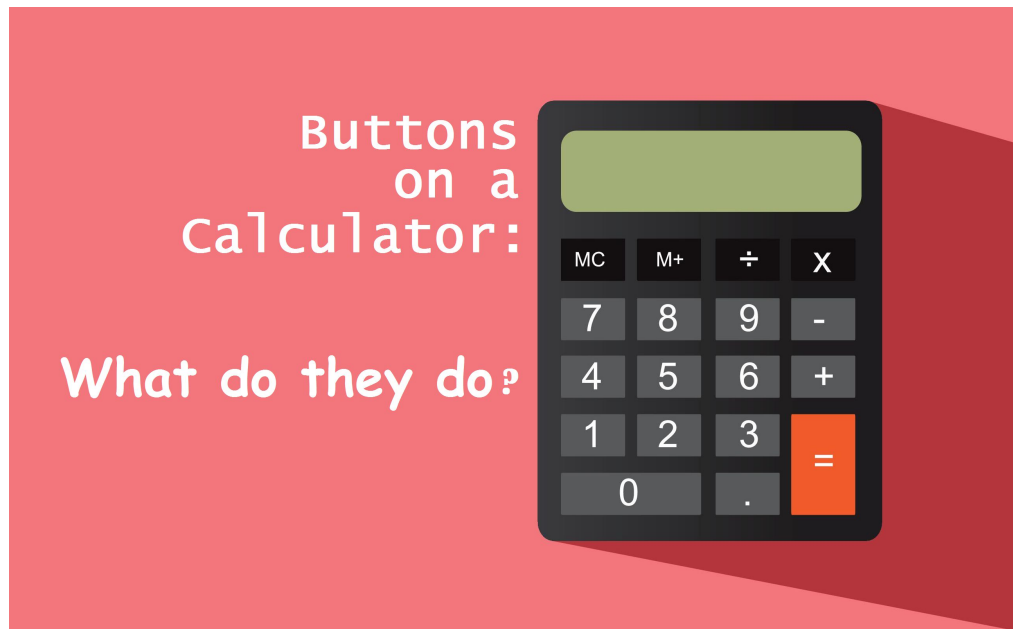
# Contents

- ✦ Motivation
- ✦ Hands-on work in RStudio
- ✦ Exercises
- ✦ Conclusions

# R topics covered

- ◆ RStudio interface.
- ◆ Addition, subtraction, basic math operations.
- ◆ Assigning values to variables.
- ◆ Commenting in a script.
- ◆ Logical operators.
- ◆ Intro to functions and libraries.
- ◆ Reading data.
- ◆ Troubleshooting error messages.
- ◆ Exploring data. (Basic summaries such as mean, median, etc.)
- ◆ Selecting subsets of data.
- ◆ Plotting data.
- ◆ Data structures available in R.

R:= A calculator with more “buttons” than you will ever use.



- ◆ R can do all that a calculator can.
  - ◆ Open R.
  - ◆ Try  $2+3$ .
  - ◆ Try  $2*3$ .
  - ◆ Try  $(2+3)/5$ .
- ◆ Conclusion: Off with the calculators!



# What is R?

- ◆ Collection of “buttons” that will perform mathematical calculations when pressed
  - ◆ The way to press “buttons” in R is by writing a command. Examples:
    - ◆ `sum(2, 3)`
    - ◆ `prod(2,3)`
    - ◆ `sqrt(4)`
- ◆ Links for useful R commands:
  - ◆ <https://www.calvin.edu/~scofield/courses/m143/materials/RcmdsFromClass.pdf>
  - ◆ <https://www.personality-project.org/r/r.commands.html>
  - ◆ <https://www.rstudio.com/wp-content/uploads/2016/10/r-cheat-sheet-3.pdf>

Excel: A boat to explore the surface of “Data-lantic ocean”.  
R: A submarine for deep exploration.

Excel



R



# R offers a lot more than Excel.

## Excel

- ◆ Familiarity.
- ◆ Simple visualizations only.
- ◆ Simple statistics only. Limited support for advanced needs.
- ◆ You know it's falling short. That's why you're here.
- ◆ Pay for an inferior option. Why?

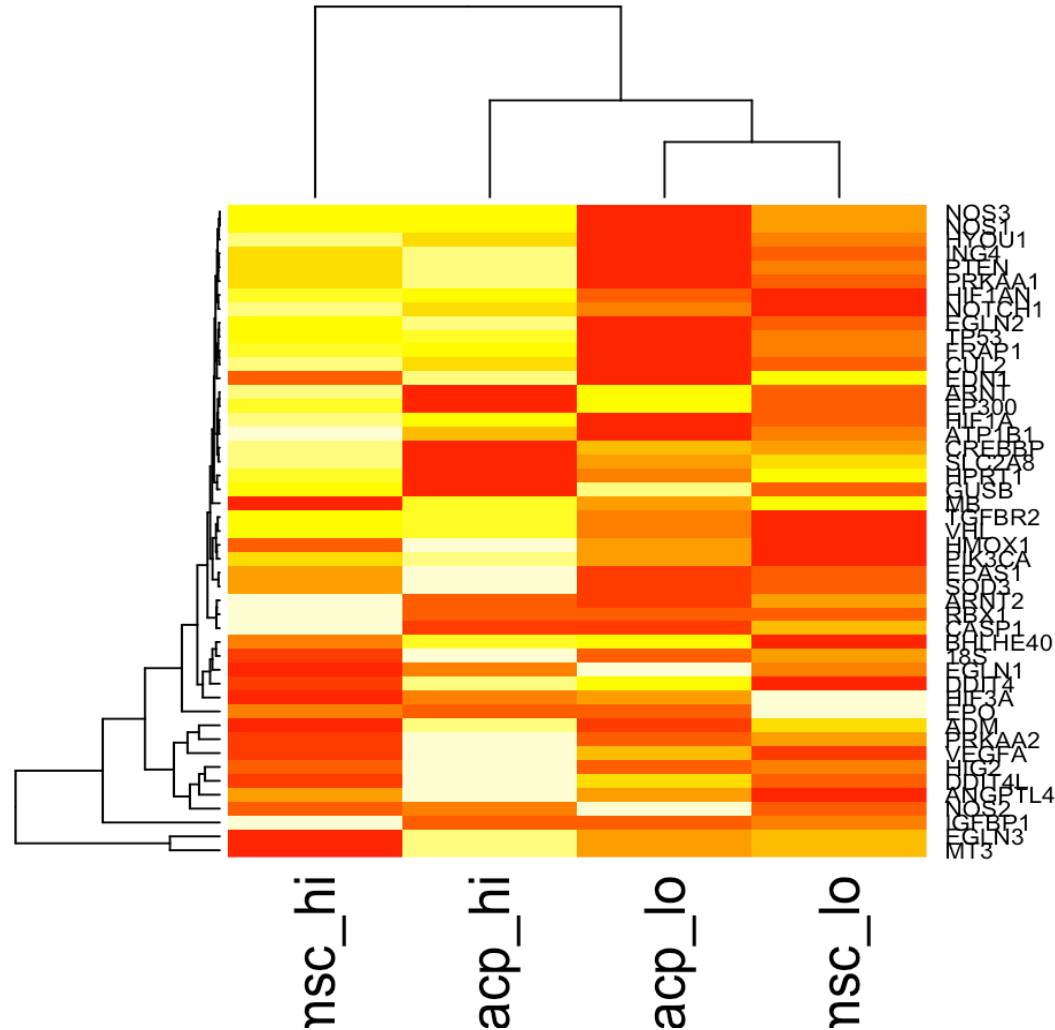
## R

- ◆ Unknown territory.
- ◆ Visualization heaven.
- ◆ If there is a statistical method out there, you'll likely find it in R. All you need is to "hit a button".
- ◆ You will need it ever more.
- ◆ Free of cost.

R: designed specifically to support variety of biological data analysis.

- ◆ Bioconductor: Software for all kinds of biological data.
  - ◆ Understands file formats commonly used in biology.
  - ◆ Vetted by large community of users.
  - ◆ Frequently updated.
  - ◆ <https://www.bioconductor.org/>

R: easy visualization of biological data. Supports interactive visualization.



## R: Increasingly becoming more central to biology.

- ✦ Biological software tools are being integrated with R.
- ✦ Upcoming training workshops may require R.

# RStudio and basic arithmetic.

# R is a console application.

- ◆ Text only interface. Inputs and outputs can be images.
- ◆ RStudio: Brings more Graphical User Interface (GUI) features to R.
  - ◆ RStudio is to R as Microsoft Word is to Notepad in Windows or TextEdit in Mac. (~kinda)
- ◆ Launch RStudio.
- ◆ Try from these:
  - ◆ `sum(2,3)`
  - ◆ `prod(2,3)`
  - ◆ `sqrt(5.5)`



# Commenting inside a source script

- ◆ Looking at an old script, or someone else's script can be scary.
- ◆ Speak to machine in code.
- ◆ Leave comments in natural language (e.g., English).
  - ◆ Anything written after `#` is interpreted by R as a comment for humans.

# Assignment *operators* in R

- ◆ Results of one calculation may be input to another.
- ◆ Save intermediate results by assigning them to variables.
- ◆ `a <- sum(2,3)`
- ◆ `a = sum(2,3)`
  - ◆ Variables store values.
  - ◆ Called so because they can be assigned any value.
- ◆ Both work but prefer to use '`<-`' for now.
- ◆ Change '`=`' to '`<-`' in the script and rerun.

# Rules for naming variables

- ◆ Names can be a combination of letters, digits, period (.) and underscore (\_).
- ◆ It must start with a letter or a period. If it starts with a period, it cannot be followed by a digit.
- ◆ Reserved words in R must not be used as variable names.

# Empty workspace/environment



# Data analysis creates objects that store data.

(top-right pane in RStudio)



## Scientific calculations may require checking criteria.

- ♦ Is p-value less than 0.05?
- ♦ Is organism named *Homo sapiens*?
- ♦ What percent of patients have heart rate greater than 130 bpm?

# Scientific calculations may require checking criteria.

- ♦ Is p-value less than or equal to 0.05?
  - ♦  $p \leq 0.05$
- ♦ Is organism named Homo sapiens?
  - ♦ `name == "Homo sapiens"`
- ♦ What percent of patients have heart rate greater than 130 bpm?
  - ♦ `bpm > 130`

# Might need to check multiple criteria simultaneously.

- ✦  $p \leq 0.05 \ \& \ \text{bpm} > 130$
- ✦  $\text{bpm} == 120 \ \& \ \text{name} == \text{"Homo sapiens"}$



## Useful *operators* in R

- ◆ ! means 'NOT'
- ◆ != means 'NOT EQUAL'
- ◆ == means 'EQUAL'
- ◆ & means 'AND'
- ◆ | means 'OR'

## Thus far...

- ◆ R console
- ◆ Source scripts
- ◆ Commenting
- ◆ Environment
- ◆ Simple calculations
- ◆ Variables

## Coming up...

- ◆ Intro to functions and libraries.
- ◆ Reading data.
- ◆ Troubleshooting error messages.
- ◆ Exploring data. (Basic summaries such as mean, median, etc.)
- ◆ Selecting subsets of data.
- ◆ Plotting data.
- ◆ Data structures available in R.

# Functions

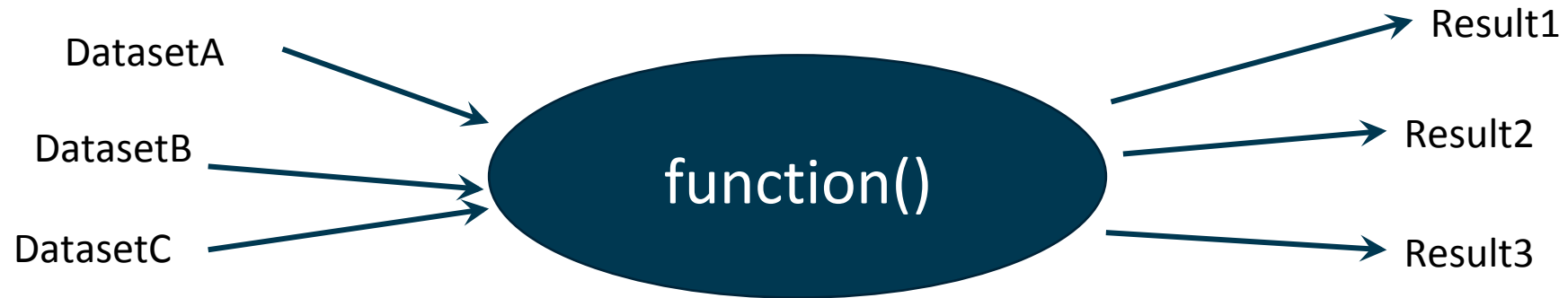
# Working with buttons for modern scientific calculations?

- ◆ The best of calculators => 10s of buttons for specific tasks.
- ◆ Modern science needs many times more than 10s.



# Functions are to R what buttons are to calculators.

- ♦ R can perform 100s of 1000s of tasks.
- ♦ Tasks are performed by using functions.
  - ♦ Examples: `sum()`, `prod()`, `mean()`, `t.test()`.



# There are functions for all sorts of things.

- ◆ Example to read a table saved in a file:
  - ◆ `read.table()`
  - ◆ Usage: `read.table("filename")`



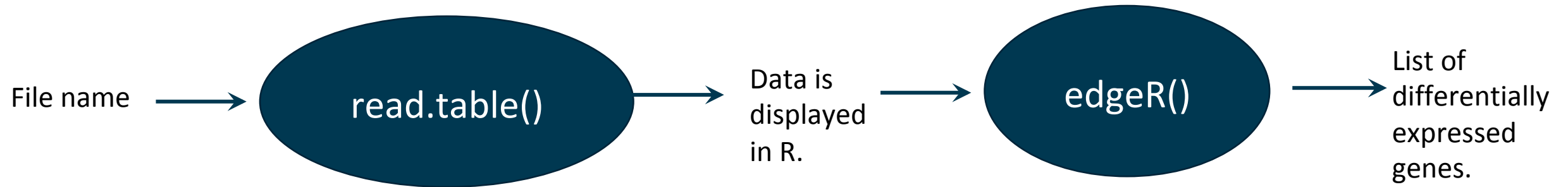
# There are functions for all sorts of things.

- ◆ Example to find complementary DNA sequence.
  - ◆ `complement()`
  - ◆ Usage: `complement(sequence)`





## Directing output of one function as input to another.



# *Library* is a collection of functions.

- ◆ R works with > 100000 functions.
- ◆ More being added everyday.
- ◆ If the software were to load (i.e., make available) all of them at startup, it will take a while to launch.
- ◆ Solution: Bundle related functions together in a *library*.
  - ◆ Example: edgeR.
- ◆ Load functions only when needed. (:= Install apps on smartphone as needed)

# Functions may need lots of information

- ◆ Calculators may take only numbers.
- ◆ R functions take variety of things. Numbers, characters, etc.
- ◆ Example:
  - ◆ Reading a file. Inputs required: File name and address on computer, formatting of file, type of file, etc.
  - ◆ Writing a file. Inputs required:
    - ◆ what to write?
    - ◆ where to write?
    - ◆ how to format the file?
    - ◆ Include column names, row names?
    - ◆ separate with comma?

Reading a data file.

# There are functions for all sorts of things.

- ◆ Functions to read files:
  - ◆ `read.csv()`
  - ◆ `read.table()`
  - ◆ `read.xlsx()`
- ◆ Try reading a file called “iris.csv”
  - ◆ `read.table(file = “iris.csv”)`

Error!!!



# How to find help?

- ◆ Execute a command name with the question mark before it. Example:
  - ◆ `?read.table`
- ◆ Read the error message!
  - ◆ Often, no need to understand every word.
  - ◆ Look for key words in error message.
- ◆ Copy-paste the error message verbatim in Google!

# We need to set directory to where the file is.

- ◆ The function for this is:
  - ◆ `setwd()`
- ◆ Can also do it interactively using the menu bar.
  - ◆ In RStudio: Session -> Set Working Directory -> Choose Working Directory.
- ◆ Verify using
  - ◆ `getwd()`



# Working with a dataset

(Read, Subset, Visualize)

# Reading a dataset

- ◆ Function we will use: `read.table()`.
- ◆ Dataset used: Iris dataset.
  - ◆ Iris is a plant genus.
  - ◆ Species of this genus may be identified based on the dimensions of petals and sepals of its flowers.
- ◆ Try `read.table("iris.csv")`.
  - ◆ `dat <- read.table("iris.csv")`

# Explore the current data

- ◆ `View(dat)` #Shows the data in a tabular format.
- ◆ `dim(dat)` #Shows the dimensions of the table, i.e. rows and columns.
- ◆ `colnames(dat)` #Shows the names of columns, if any.
- ◆ `summary(dat)` #Outputs summary statistics of data. Ex- minimum, max, median, etc.
- ◆ `head(dat)` #Displays first 6 rows of data.
- ◆ `tail(dat)` #Displays last 6 rows of data.

# Extracting parts of data in another variable.

- ♦ `spl_len <- dat$Sepal.Length`
  - ♦ `$` is used to refer to a feature in the data.
  - ♦ Check what `spl_len` is.
  - ♦ `class(spl_len)`
- ♦ It contains numbers, makes sense to call it “numeric”.
- ♦ `spl_len` is an ordered series of numeric values.
  - ♦ Ordered series of values are called vectors in R. (More on that later)

## Another data type: *Factor*

- ◆ `spcs <- dat$Species`
  - ◆ Check what `spcs` is.
  - ◆ Factor variable.
  - ◆ `spcs <- as.character(spcs)`
  - ◆ `spcs <- unique(spcs)`
  - ◆ How to check if `spcs` includes `sapiens`?
    - ◆ `"sapiens" %in% spcs`
- ◆ Factors are used to represent categorical variables.
- ◆ `spcs` is now a character vector.

## Goal: To extract subset of data for *Iris setosa*.

- ◆ `which_setosa <- dat$Species == "setosa"`
  - ◆ `dat$Species == "setosa"` compares each value in `dat$Species` with character "setosa".
  - ◆ If a value equals "setosa", output is TRUE.
  - ◆ Check what `which_setosa` is. (a logical vector)
- ◆ To extract the subset for *Iris setosa*:
  - ◆ `dat_setosa <- dat[which_setosa, ]`

## Goal: To extract subset of data for *Iris setosa*.

- ◆ Alternatively
  - ◆ `dat_setosa <- subset(dat, Species == "setosa")`
  - ◆ If only I had googled!!
- ◆ Wishing you could just hit enter?
- ◆ Feeling like a lot of people would want it?
- ◆ Google!
- ◆ TRUE for most anything at a beginner level.
- ◆ If you get an error, someone else would have got it too. Google!

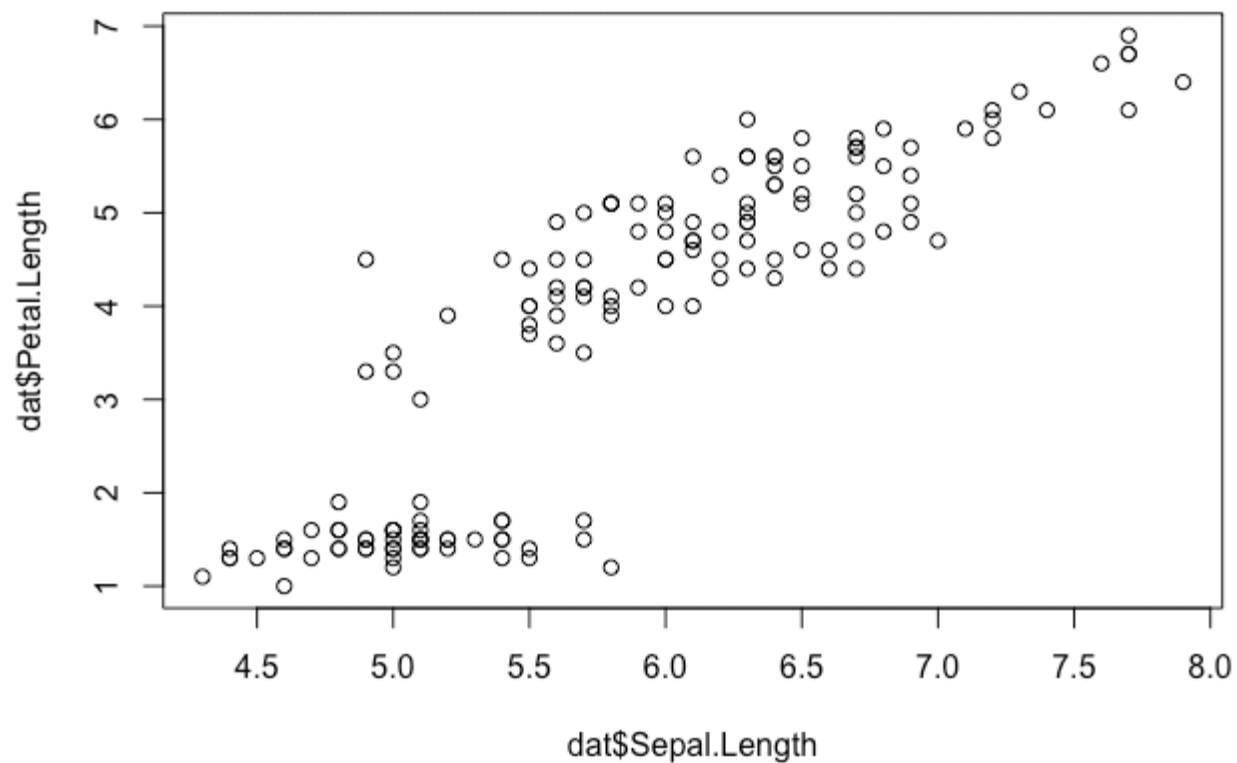
# Basic statistics.

- ◆ Mean
  - ◆ `mean(dat$Sepal.Length)`
- ◆ Median
  - ◆ `median(dat$Petal.Length)`
- ◆ Standard deviation
  - ◆ `sd(dat$Petal.Width)`
- ◆ Histogram
  - ◆ `hist(dat$Sepal.Width)`
- ◆ Boxplot
  - ◆ `boxplot(dat$Sepal.Length)`



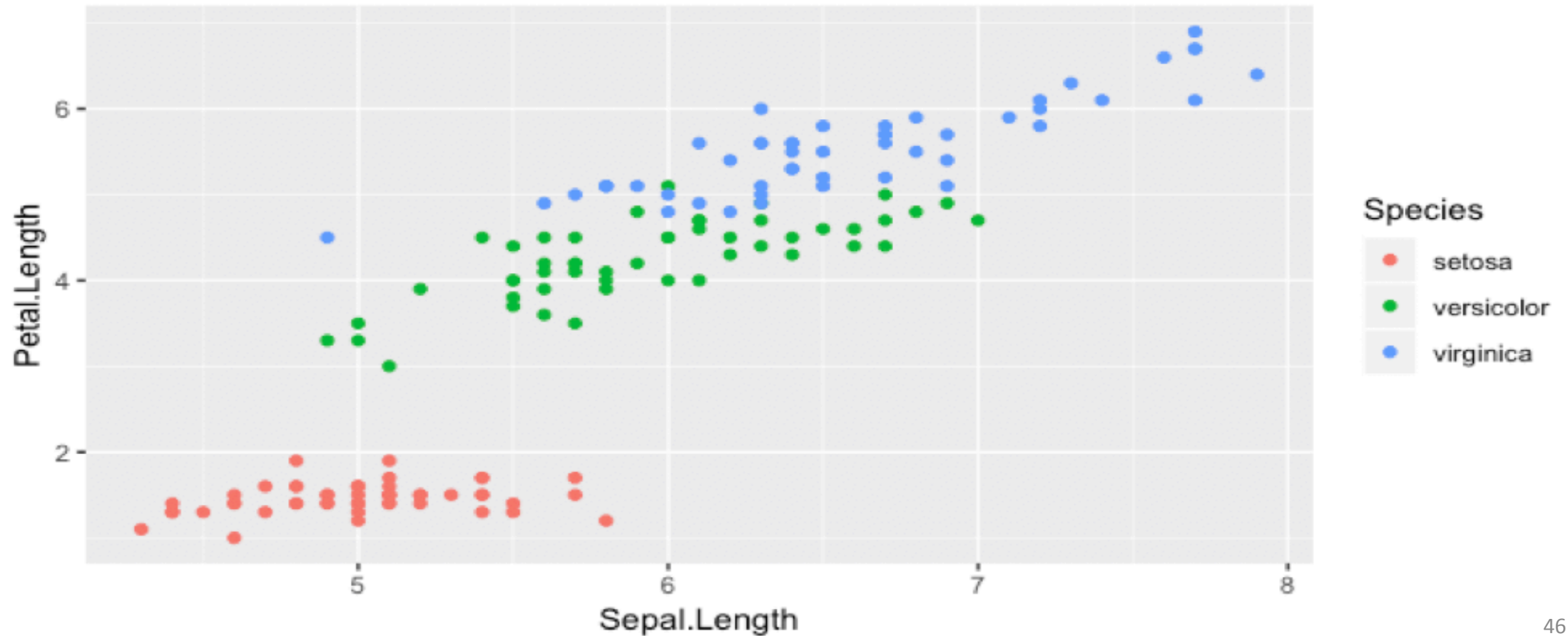
# Visualizing data.

◆ `plot(x= dat$Sepal.Length, y = dat$Petal.Length)`



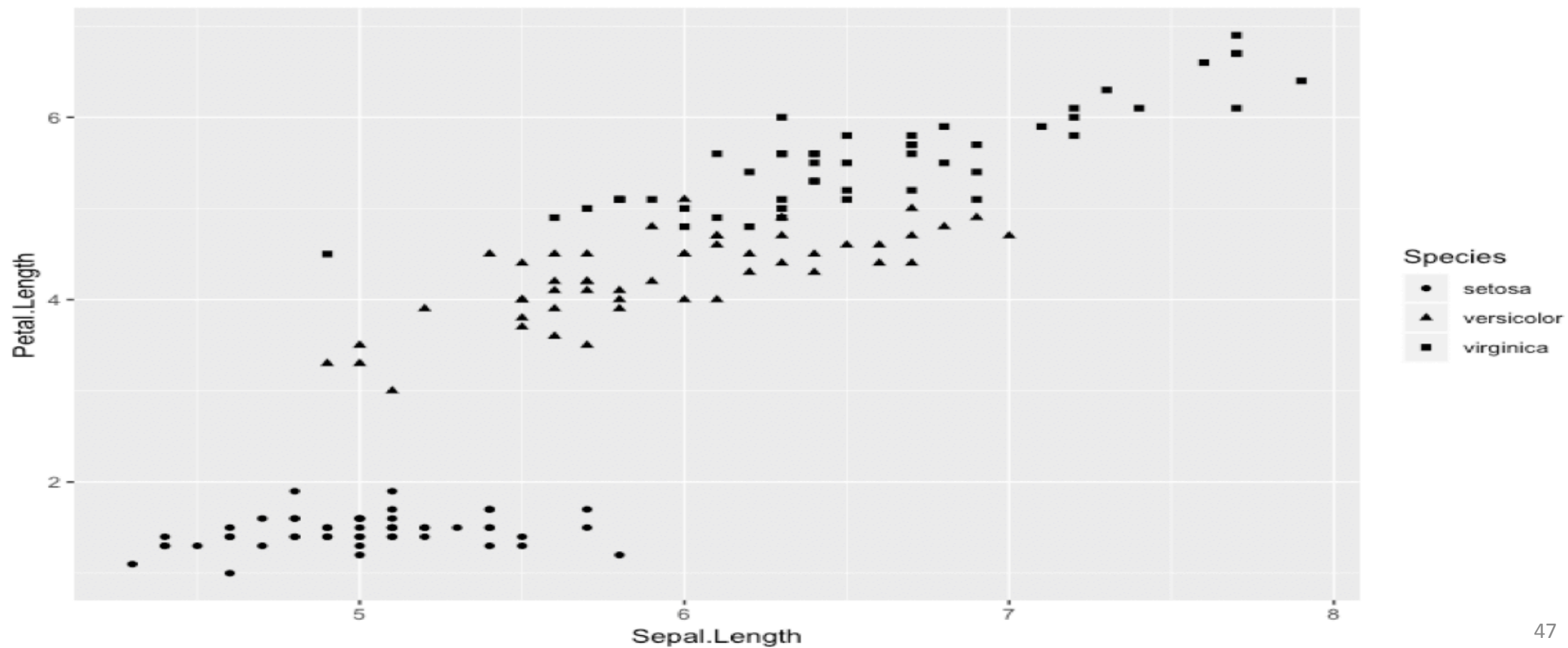
# Visualizing data.

◆ `qplot(x = Sepal.Length, y = Petal.Length, data = dat, color = Species)`



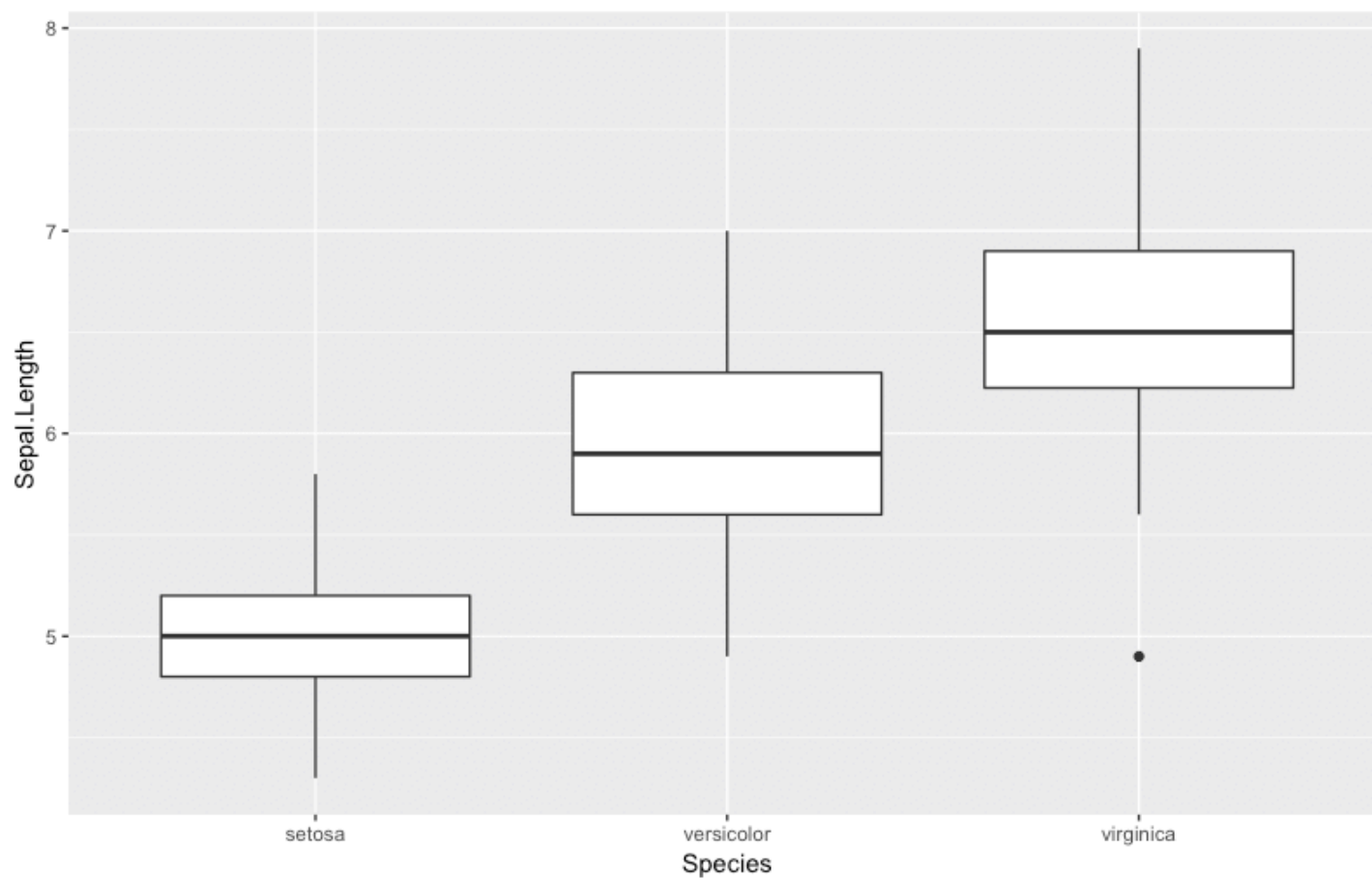
# Visualizing data.

◆ `qplot(x = Sepal.Length, y = Petal.Length, data = dat, shape = Species)`



# Visualizing data.

- ◆ `qplot(x = Species, y = Sepal.Length, data = dat, geom = "boxplot")`



# Loops and conditional statements.

# Loop over rows in a tabular data

```
◆ table <- read.table("iris.csv")
◆ for (row in table) {
    #Do some calculation for each row
    ...
    ...
}
```

# Conditional statements

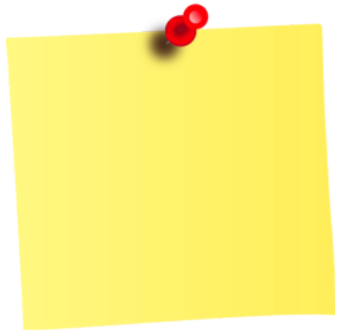
```
♦ if (p < 0.05) {  
    print(gene_info)  
    ...  
} else {  
    print(gene_name)  
}
```

# Data structures.

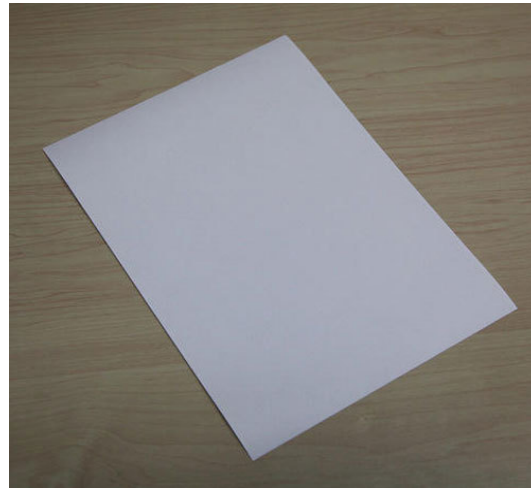


# Data structures are data organization/storage formats.

◆ Storing data physically? Options:



Sticky note



Printer paper



Task pad



Long scroll

# Data structures are data organization/storage formats.

- ♦ Working with data in R? Options:
  - ♦ Numeric vector
    - ♦ `c(2, 5, 10, 11)`
  - ♦ Character vector
    - ♦ `c("apples", "oranges", "butter", "dry yeast")`
  - ♦ Data frames
    - ♦ `dat` from Iris example.
  - ♦ Matrix
    - ♦ Example counts for genes in several samples.
  - ♦ Lists
    - ♦ Flexible data structures.
    - ♦ `mylist <- list(a = c(2, 5, 10, 11),  
                  b = c("apples", "oranges", "butter", "dry yeast")  
                  )`

# Data structures for special needs?

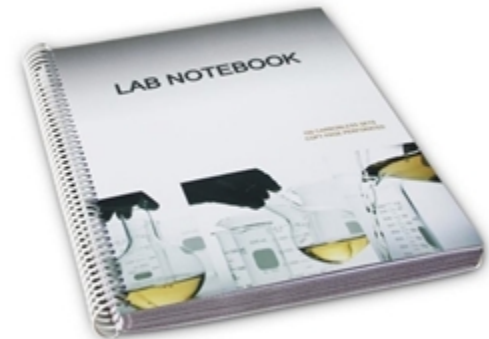
- ◆ Storing data physically? Options:



Sheet music



Ledger



Lab notebook

# Data structures for special needs?

- ◆ Working with data in R?
  - ◆ Packages may have author-defined data structures.
- ◆ Example – DGEList defined in edgeR.
  - ◆ Stores counts, sample grouping information, normalization factors, etc. in one place.

# Exercise

# Reading unclean data.

- ◆ “dirty\_data.txt” contains data that is formatted differently.
- ◆ Figure out how to read it.
- ◆ Tips:
  - ◆ Error message comes in peace. It is your friend!
  - ◆ Google!

# Alternative file types

- ♦ Scientific studies may involve various kinds of files.
- ♦ Example:
  - ♦ FASTA files for sequences.
  - ♦ FASTQ files for sequencing reads.
  - ♦ SAM files for alignment.
  - ♦ GTF files for gene annotations.
  - ♦ Excel sheets with multiple sheets.
- ♦ Google for how to read FASTA file in R. (5 min)

# How to read in FASTA files?

- ◆ Searching for packages returns:
  - ◆ seqinr
  - ◆ Biostrings
  - ◆ ...
- ◆ Example with seqinr
  - ◆ `install.packages("seqinr")`
  - ◆ `library(seqinr)`
  - ◆ `read.fasta("16S.fasta")`
- ◆ Check options available with `read.fasta`.
  - ◆ `?read.fasta`



# Recap of R topics covered

- ◆ RStudio interface.
- ◆ Addition, subtraction, basic math operations.
- ◆ Assigning values to variables.
- ◆ Commenting in a script.
- ◆ Logical operators.
- ◆ Intro to functions and libraries.
- ◆ Reading data.
- ◆ Troubleshooting error messages.
- ◆ Exploring data. (Basic summaries such as mean, median, etc.)
- ◆ Selecting subsets of data.
- ◆ Plotting data.
- ◆ Data structures available in R.

# Your feedback is important to us!

- ✦ <https://bioinformatics-course-feedback.questionpro.com/>
- ✦ ~3 min.

# Possibilities

# Detailed high resolution graphics : Examples

- ◆ <https://www.r-graph-gallery.com/>

# Bioconductor for biological data analysis

## Packages

- ◆ edgeR, DESeq2, limma, voom
- ◆ phyloseq, microbiome
- ◆ rWikipathways, KEGGgraph
- ◆ MSstats
- ◆ bsseq, methylKit
- ◆ chipseq
- ◆ CRISPRseek, gCrisprTools, CrispRVariants

## Uses

- ◆ Differential gene expression analysis
- ◆ Microbiome data analysis
- ◆ Pathway analysis
- ◆ Proteomic data analysis
- ◆ DNA methylation analysis
- ◆ ChIP-Seq data analysis
- ◆ CRISPR related data analysis

# Conclusions

- ◆ R is a “smart” version of calculators.
- ◆ Error message comes in peace.
- ◆ Google!

## Upcoming office hour and workshops

- ◆ Bioinformatics office hour (Sept. 24, 1:30-2:30 PM, MH 1107)
- ◆ Hypothesis testing (Sept. 25 and Oct. 9)
- ◆ Intermediate R: Data visualization (Oct. 1 and 3)
- ◆ Intro to Unix command line (Nov. 5)
- ◆ Intermediate RNA-seq data analysis (Nov. 18 and Dec. 11)







The background features a dark blue field with several large, overlapping, wavy lines in a lighter blue shade. These lines create a sense of depth and movement. Overlaid on these waves is a pattern of small, light blue rectangular dashes arranged in a grid-like fashion, which also follows the contours of the waves.

# GLADSTONE INSTITUTES