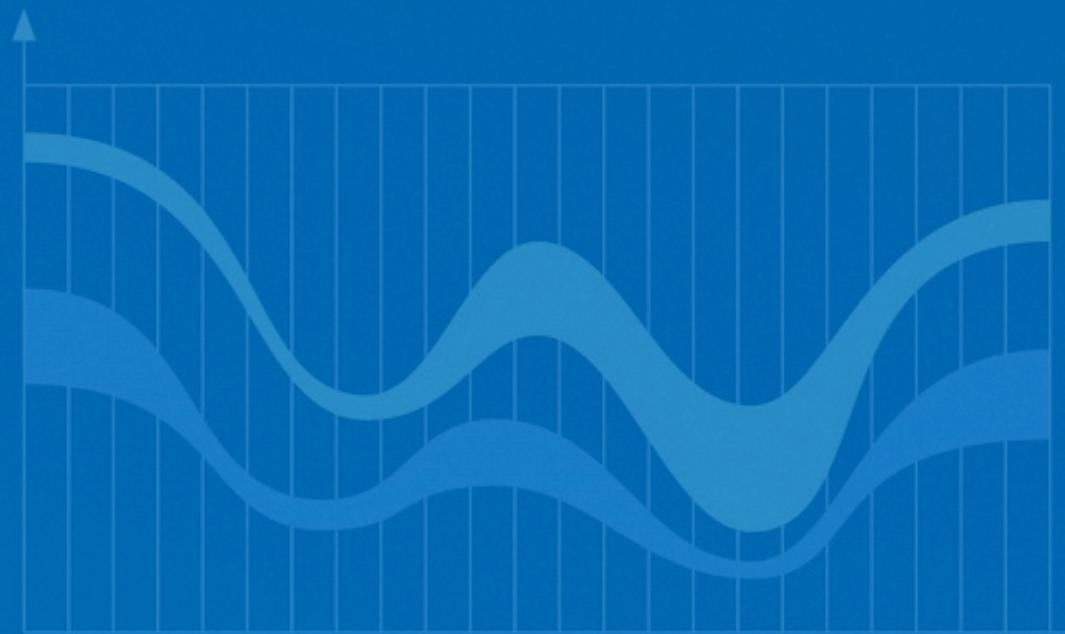
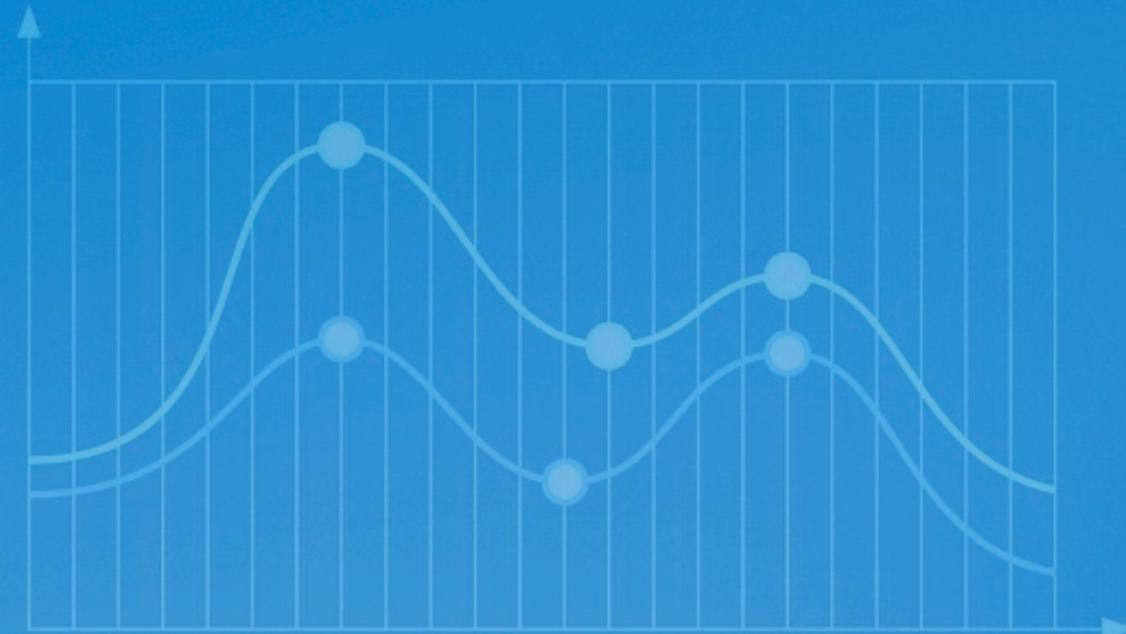




# R for Data Science

Center for Health Data Science



# Who Are We?



## Center for Health Data Science (HeaDS)

- DS Research Groups
- HDS Sandbox
- SUND DataLab

<https://heads.ku.dk/>

## SUND DataLab

- Courses & Workshops
- Consultations
- Commission Research & Supervision
- Events, Seminars



Thilde Terkelsen



Henrike Zschach



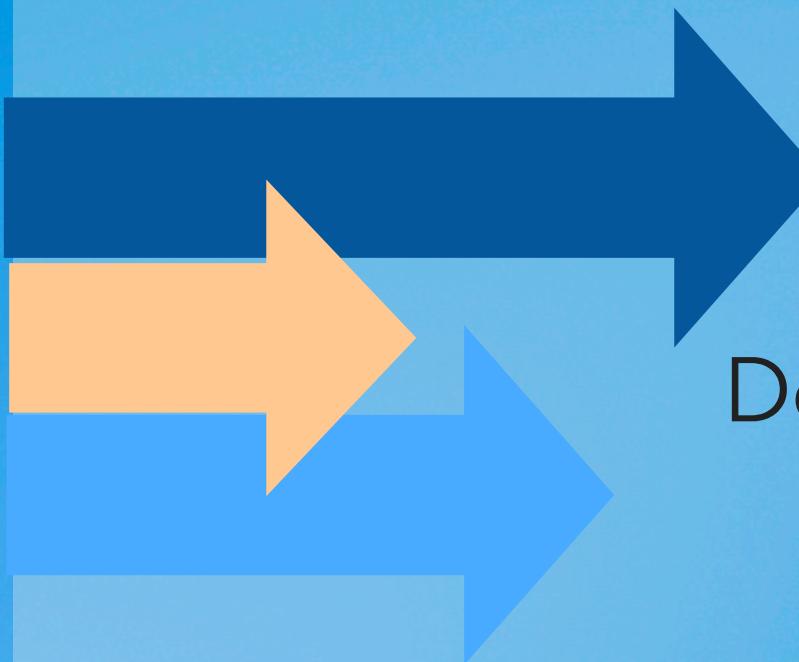
Diana Andrejeva



Helene Wegener

# Course Practicals

- Three days: 9.00-16.00. There shall be breaks with plenty of coffee ☕
  - Location all three days: **Henrik Dam, Panum Building, Blegdamsvej 3B**
- The course is build on hands-on presentations & exercises
- **Books:**
  - “R for Data Science”. Free Online: <https://r4ds.had.co.nz/>
  - “R for Statistical Learning”. Free Online: <https://daviddalpiaz.github.io/r4sl/>



Download & install the **newest version of R**: <https://cran.r-project.org/>

Download & install newest version of **R-studio**: <https://posit.co/download/rstudio-desktop/>

**Go to course website:** <https://center-for-health-data-science.github.io/R4DataScience/>

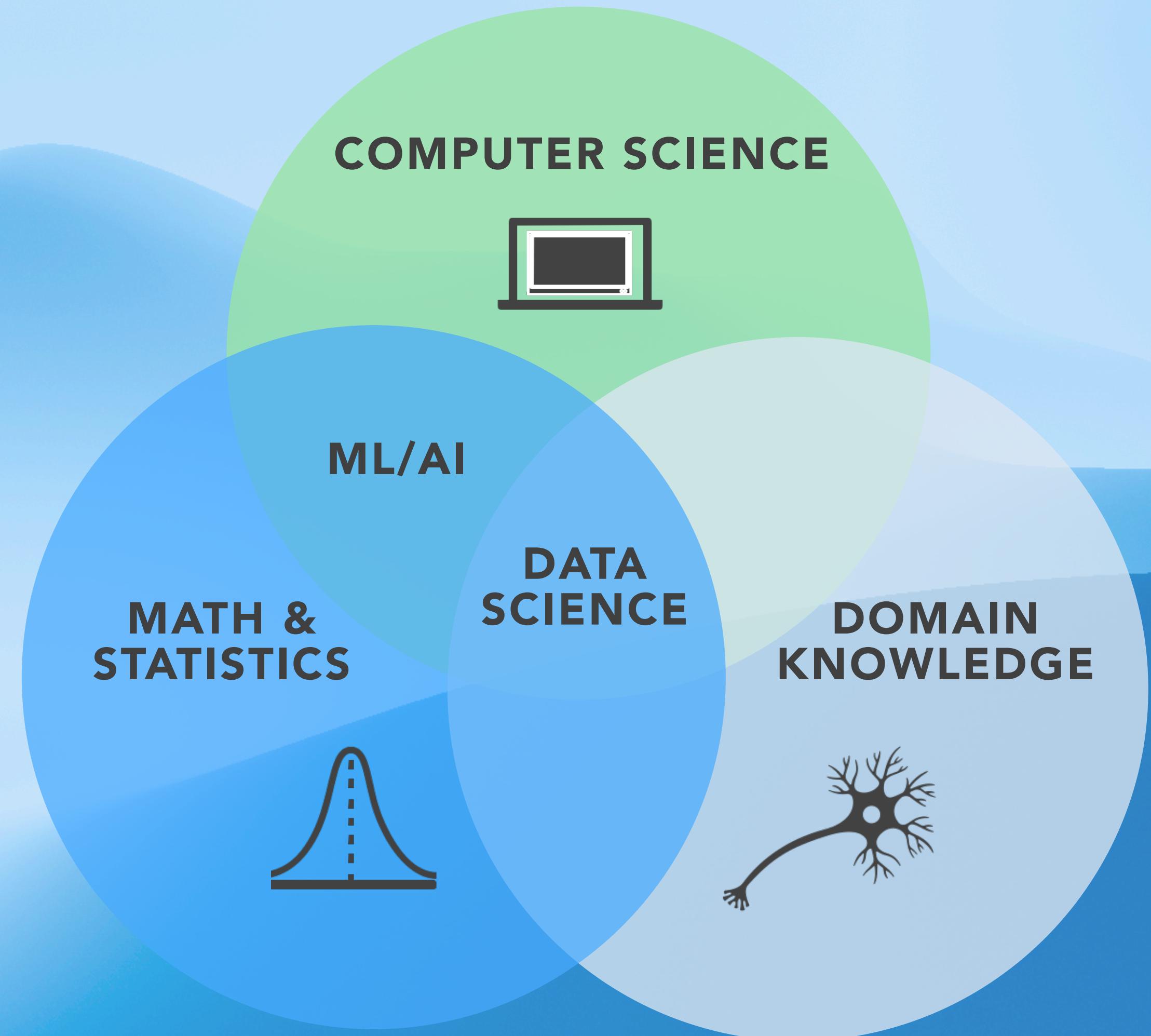
# Program

Day 1		Day 2		Day 3	
08:30-09:00	<b>Installation Issues + Coffee</b>	08:45-09:00	<b>Coffee</b>	08:45-09:00	<b>Coffee</b>
09:00-09:30	<b>Introduction to Course</b>	09:00-09:45	<b>P3: Exploratory Data Analysis</b>	09:00-09:45	<b>P5A: Models in R</b>
09:30-10:00	<b>P0: Intro to R and Quarto</b>	09:45-11:00	<b>E3: Exploratory Data Analysis</b>	09:45-11:00	<b>E5A: Regression Models</b>
10:00-11:00	<b>E0: Intro to R and Quarto</b>	11:00-11:15	<b>Break</b>	11:00-11:15	<b>Break</b>
11:00-11:15	<b>Break</b>	11:15-12:00	<b>P4A: Conditionals &amp; Loops</b>	11:15-12:00	<b>P5B: Model Evaluation</b>
11:15-12:00	<b>P1: Data Cleaning</b>	12:00-13:00	<b>Lunch</b>	12:00-13:00	<b>Lunch</b>
12:00-13:00	<b>Lunch</b>	13:00-14:00	<b>E4A: Conditionals &amp; Loops</b>	13:00-14:15	<b>E5B: Elastic Net &amp; RF</b>
13:00-14:00	<b>E1: Data Cleaning</b>	14:00-14:45	<b>P4B: Functions</b>	14:15-14:30	<b>Break</b>
14:00-14:45	<b>P2: Summary Stats &amp; Wrangling</b>	14:45-15:00	<b>Break</b>	14:30-14:45	<b>Course evaluation</b>
14:45-15:00	<b>Break</b>	15:00-16:00	<b>E4B: Functions</b>	14:45-16:00	<b>Bring your own data</b>
15:00-16:00	<b>E2: Summary Stats &amp; Wrangling</b>				
16:00	<b>See you tomorrow!</b>	16:00	<b>See you tomorrow!</b>	16:00	<b>Course Finished!</b>

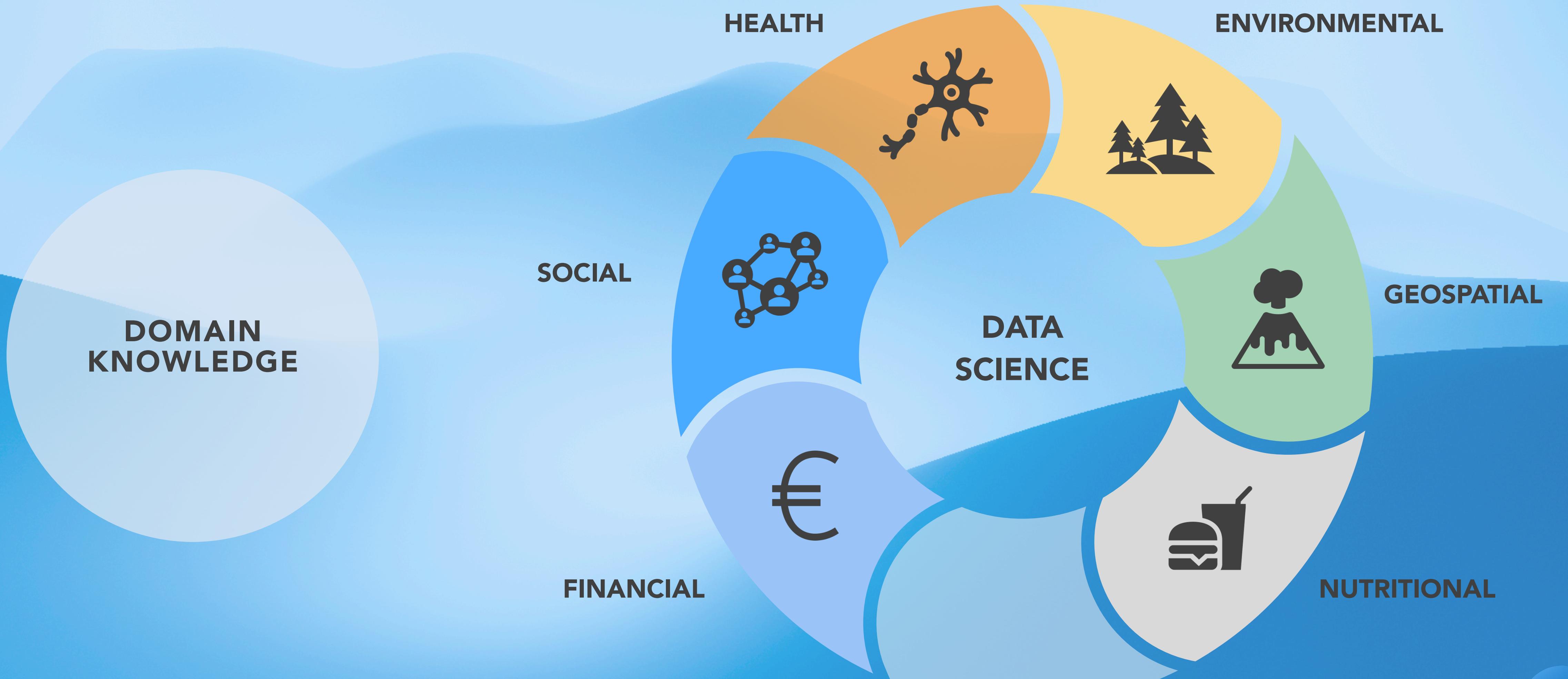


# What is Data Science?

- A **cross-disciplinary** undertaking that draws on many disciplines and is in turn becoming part of many disciplines.
- While ML is an important part of DS it is just one part of it.
- Data matters, one size analysis does not fit all.



# Flavors of Data Science?



# Analysis of Health Data

Research /Clinical Studies

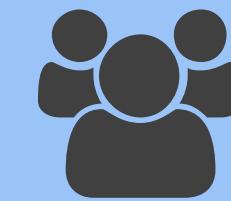
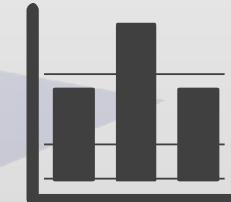
Registries & Questionnaires  
Pharmacological Databases

Biological (omics) Databases  
Medical Image Data

Wearable Devices  
Search Engine Data

'Future' Data

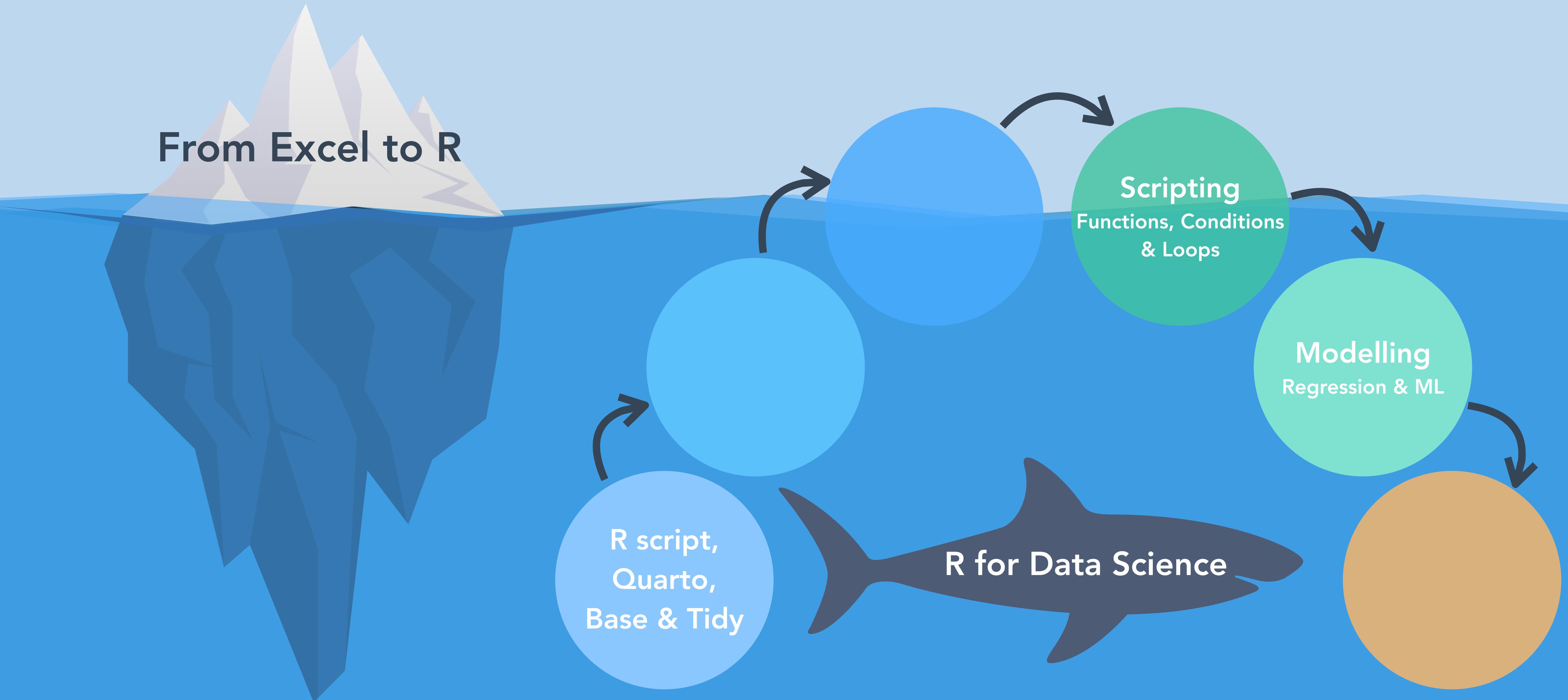
**DATA SCIENCE  
SKILLSET**



?



# What Will We Learn



# **Part 0**

# **Intro to R and Quarto**



# R script

- Flat script
  - Fast, less complex parts than Quarto
  - Make a Software/R-package
  - Submit to high performance compute (HPC) environment
- Comment script and build structure using #
- Source content of script to session

# Quarto

- Markdown-based
- Render to get a nice report in html or website
- Text and code easily distinguishable
- Analysis well-documented

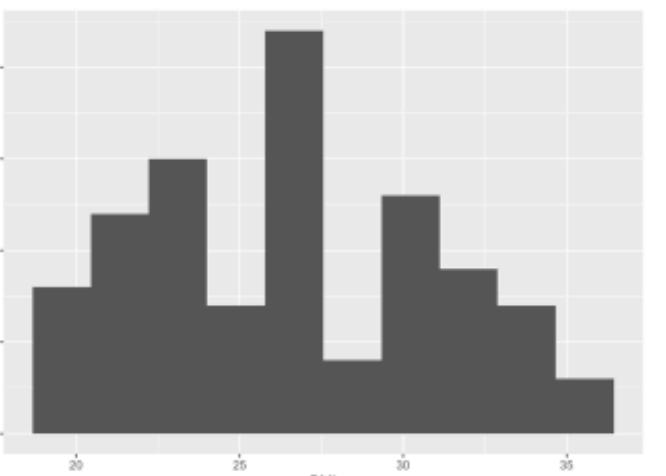
# R script

```
+ R_script_example.R x
+ Source on Save |   
1 #####
2 # R for Data Science - How to R script #
3 # Author: DataLab HeaDS #
4 # Date: 8 November 2024 #
5 #####
6
7 #####
8 ##### Load Packages #####
9 #####
10 library(tidyverse)
11 library(readxl)
12
13 #####
14 ##### Load Data #####
15 #####
16 diabetes <- read_excel('~/Desktop/DataLab/R4DataScience/data/diabetes_toy.xlsx')
17 #####
18 ##### Inspect Data #####
19 #####
20 #####
21
22 # Check dimensions of data
23 dim(diabetes)
24
25 # Check structure of data
26 str(diabetes)
27
28 # Check for NA's in each column
29 colSums(is.na(diabetes))
30
31 #####
32 ##### Exploratory Data Analysis #####
33 #####
34
35 # Plot distribution of BMI
36 diabetes %>%
37   ggplot(aes(x = BMI)) +
38   geom_histogram(bins = 10)
39
40
```

# Quarto

```
+ Quarto_example.qmd x
+ Source | Visual | B | I | </> | Normal |     Format | Insert | Table
1 ---
2   title: "R for Data Science - How to R script"
3   format: html
4   author: DataLab HeaDS
5   editor: visual
6
7
8 #####
9 ##### Load Packages #####
10 {r}
11 library(tidyverse)
12 library(readxl)
13
14 #####
15 ##### Load Data #####
16 {r}
17 diabetes <- read_excel('~/Desktop/DataLab/R4DataScience/data/diabetes_toy.xlsx')
18
19 #####
20 ##### Inspect Data #####
21
22 Check dimensions of data
23 {r}
24 dim(diabetes)
25
26 Check structure of data
27 {r}
28 str(diabetes)
29
30 Check for NA's in each column
31 {r}
32 colSums(is.na(diabetes))
33
34 #####
35 ##### Exploratory Data Analysis #####
36
37 Plot distribution of BMI
38 {r}
39 diabetes %>%
40   ggplot(aes(x = BMI)) +
41   geom_histogram(bins = 10)
```

**Render .html .pdf**



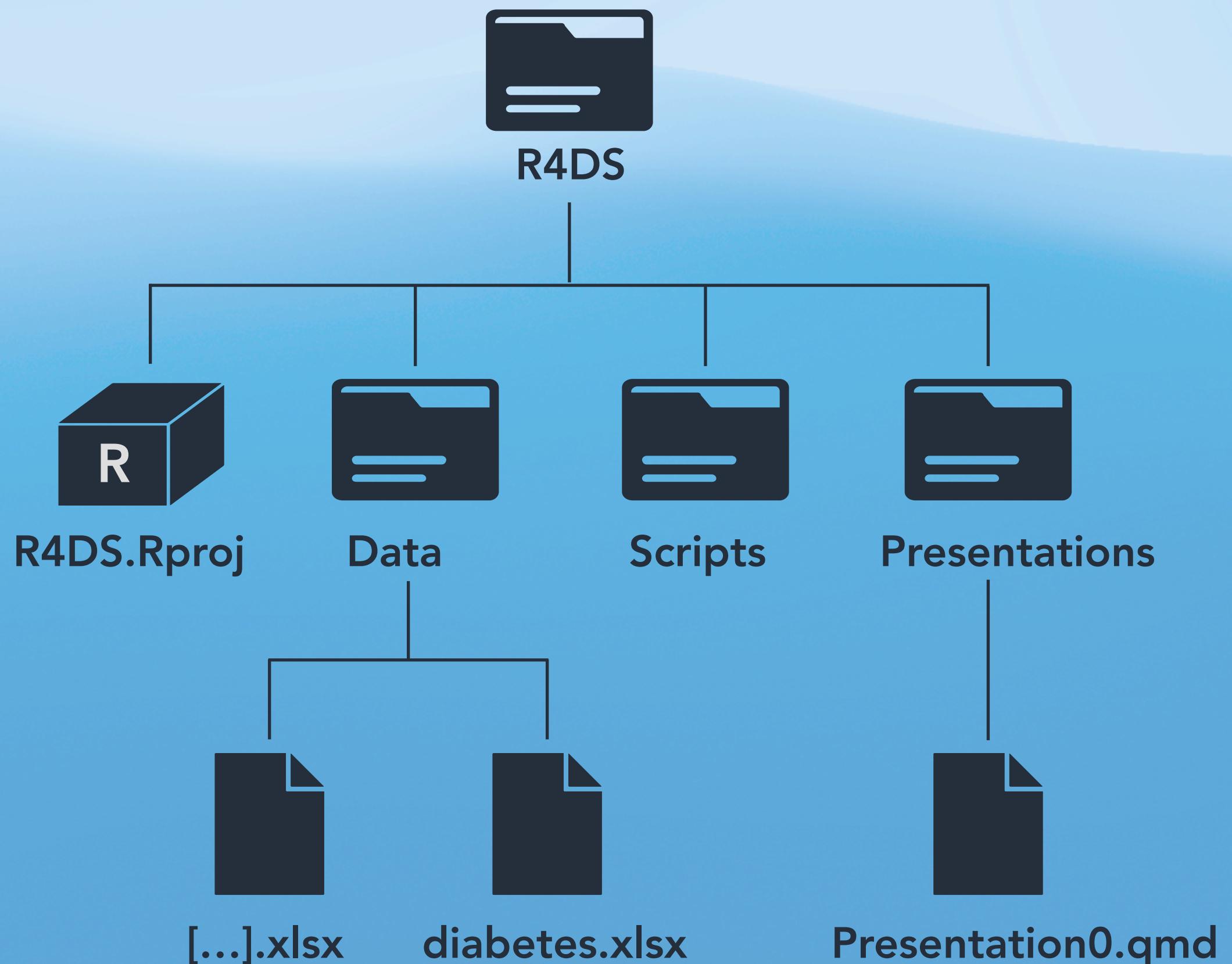
# R project

## R4DS.Rproj

- Location of files is the root for all files within the directory.

## .Rproj.user

- Hidden directory to store of temporary files.



# **Exercise 0**

## **Intro to R and Quarto**

# **Part 1**

# **Data Cleaning**



# Data Cleaning

1. Drop Variables with Too Many Missing Values
  - Can introduce noise and bias
2. Fix Variable Types
  - Change one type of variable to another type
3. Drop Non-Informative Variables
4. Recode and Reorder Factors
  - Ensure categorical variables are spelled the same
5. Create New Variables from existing ones

Merge several datasets into one.

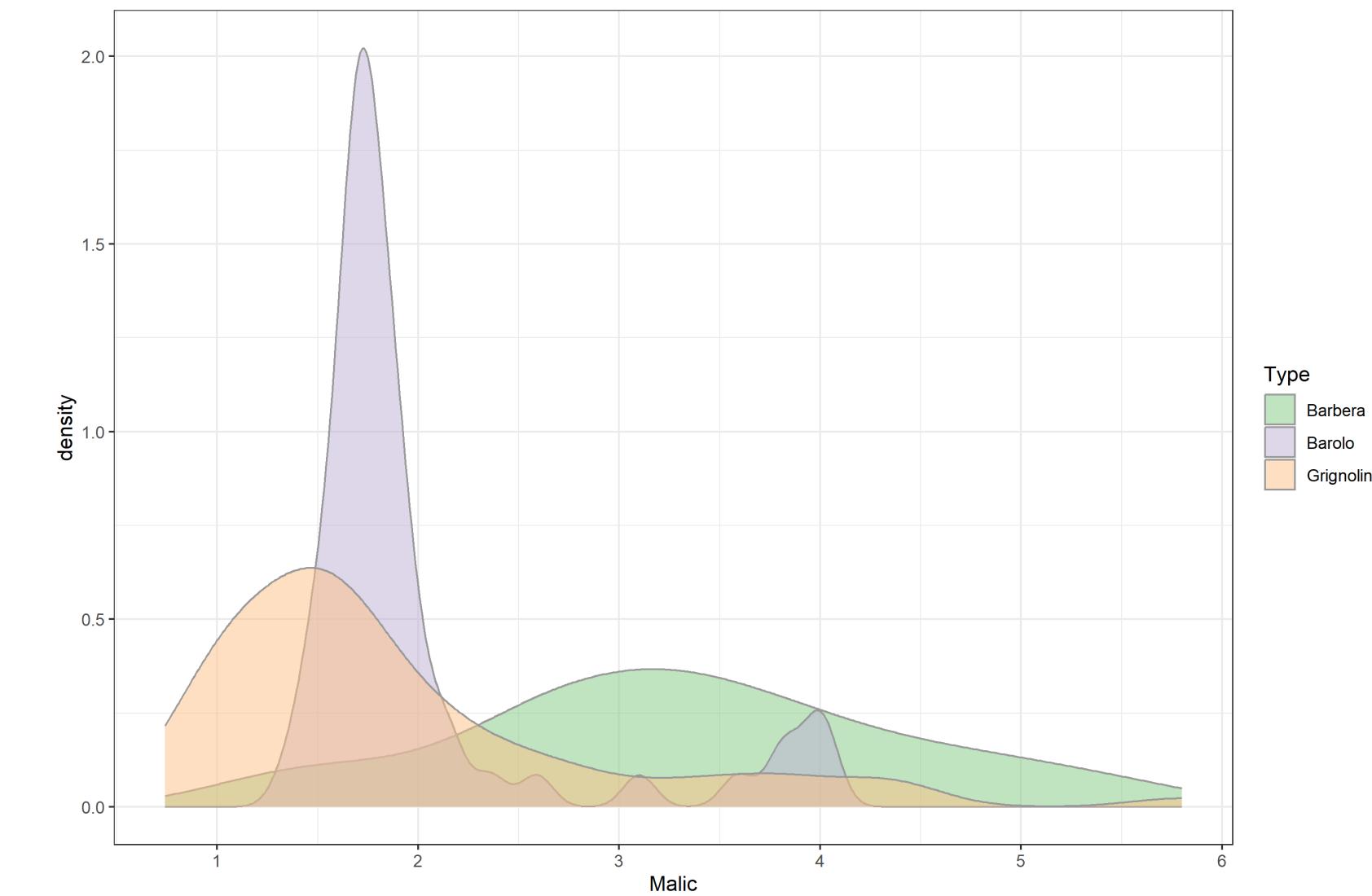
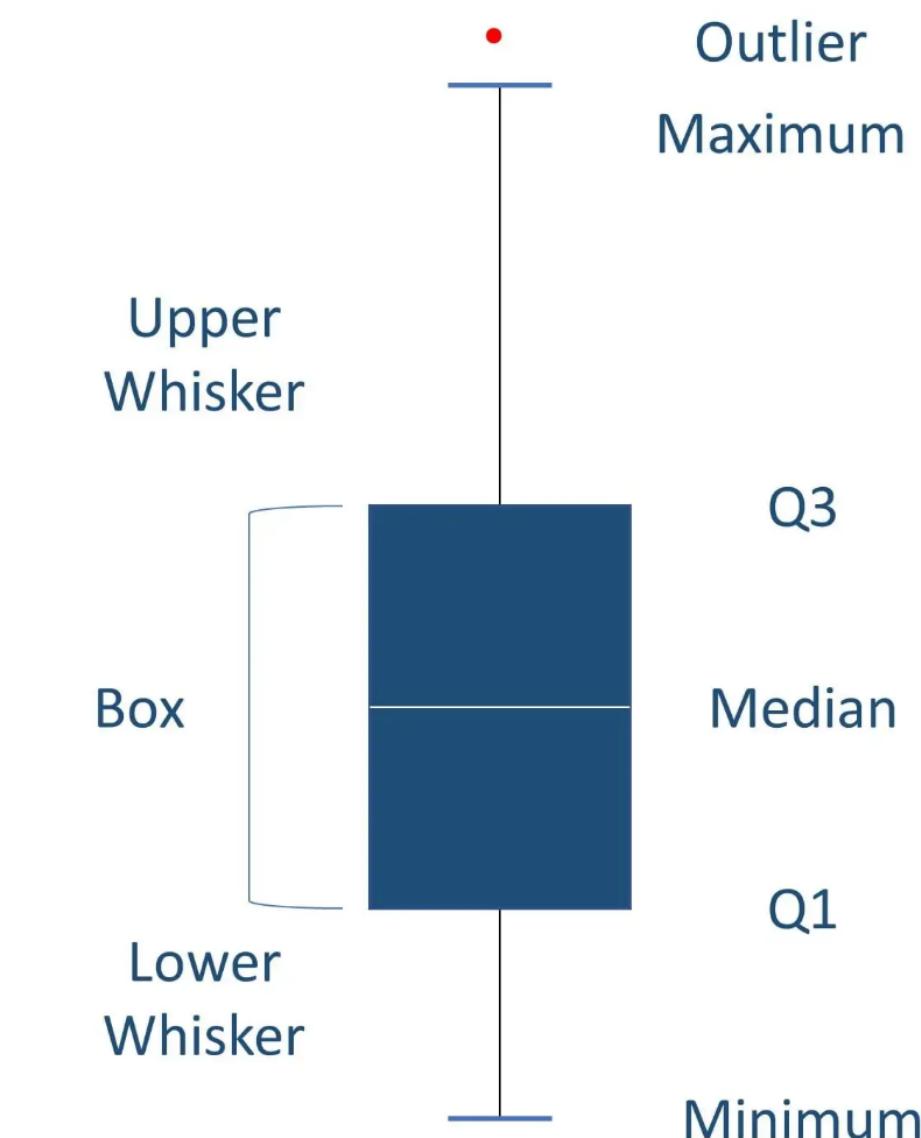
Spot using summary statistics.

# Basic Summary Stats

Summary statistics are used to distill a dataset into some key characteristics.

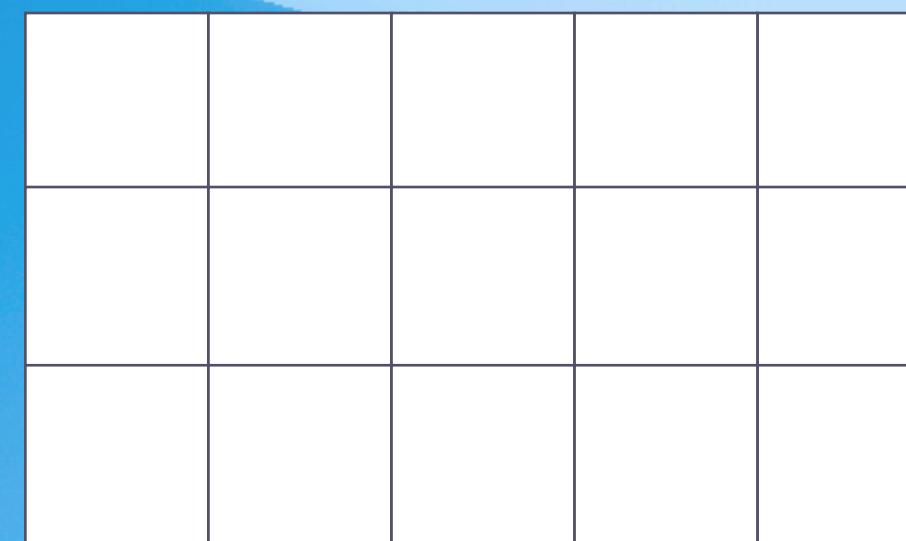
They often include the following measures:

- Central tendency (mean, median, mode)
- Spread (standard deviation)
- Minimum and Maximum
- Quantiles
- Shape of the distribution



# R Data Types & Structures

Observations



Variables

## Data Types

Numeric

-1.5, 2.7, 3.2

Integer

-2, -1, 0, 3, 5

Character

"A", "Bat", "Tom"

Factor

G1, G1, G2, G3

Logical

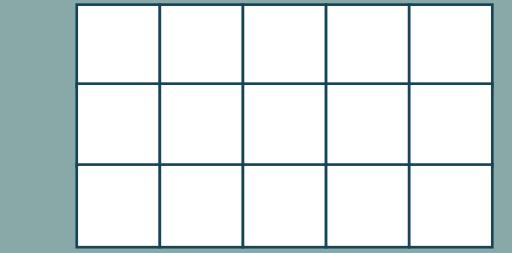
TRUE, FALSE

## Data Structures

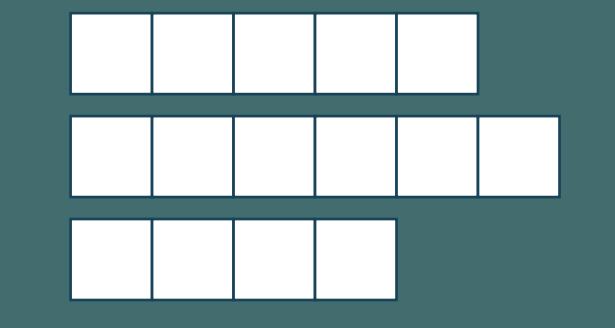
Vector



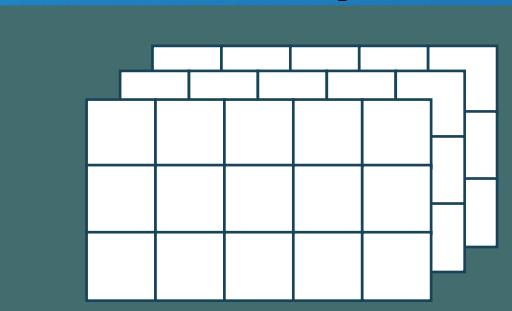
Data Frame



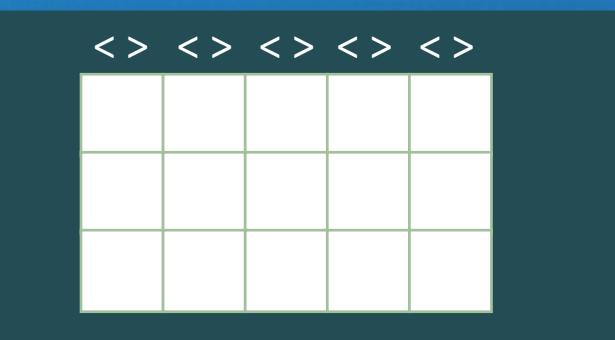
List



Array



Tibble



# Base R



- Basic library that is pre-installed in R
- Supports all (also older) versions of R
- Efficient for many tasks!
- Functions assume **dataframe** or **matrix**
- Quick preliminary plots
- For complex workflows the syntax gets convoluted

# Tidyverse



- A collection of packages:
  - dplyr (data manipulation)
  - ggplot2 (visualization)
  - tidyr (reshaping)
  - stringr (string manipulation)
- Uses pipe symbol `%>%`
- Intuitive syntax
- Functions assume **tibble**
- Can get slow for very big datasets

# Cheat Sheet - String manipulation

	Command
Combines string with a separator	<code>paste("STRING", sep = " ")</code>
Combines string without a separator	<code>paste0("STRING")</code>
Splits a string into parts in a list.	<code>str_split("STRING", pattern = " ")</code>
Splits a string into parts in a list and retrieves the i'th index.	<code>str_split_i("STRING", pattern = " ", i = 1)</code>
Detect substring in main string.	<code>str_detect("STRING", SUBSTRING)</code>
Replaces pattern with replacement in string	<code>str_replace_all("STRING", pattern = "", replacement = "")</code>
Remove whitespace	<code>str_trim("STRING")</code>

More on stringr: <https://stringr.tidyverse.org/>

# **Exercise 1**

## **Data Cleaning**

# **Part 2**

# **Summary Statistics and Data Wrangling**

# ggplot2 Recap

## Syntax

```
df %>%  
  ggplot(aes(x = var1,  
             y = var2,  
             color = var3)) +  
  geom_XXX() +  
  scale_color_manual(values = c("A", "B")) +  
  labs(title = "title",  
       x = "x axis title",  
       y = "y axis title",  
       color = "legend title") +  
  theme_XXX()
```

## Aesthetics (aes)

- x
- y
- color
- fill
- shape
- group
- linetype

## geom

geom\_point()  
geom\_line()  
geom\_boxplot()  
geom\_violin()  
geom\_bar()  
geom\_col()

## Themes

theme\_grey() # default  
theme\_classic()  
theme\_minimal()  
theme\_bw()  
theme\_light()  
theme\_dark()

# Cool New Plot?

The R Graph Gallery

CHART TYPES PKG BEST QUICK TOOLS ALL RELATED

Distribution

Violin Density Histogram Boxplot Ridgeline Beeswarm

Evolution

Line plot Area Stacked area Streamchart Time Series

Correlation

Scatter Heatmap Correlogram Bubble Connected scatter Density 2d

Map

Map Choropleth Hexbin map Cartogram Connection Bubble map

Ranking

Barplot Spider / Radar Wordcloud Parallel Lollipop Circular Barplot Table

Flow

Chord diagram Network Sankey Arc diagram Edge bundling

Part of a whole

Grouped and Stacked barplot Treemap Doughnut Pie chart Dendrogram Circular packing Waffle

General knowledge

Cheatsheets Animation Interactivity 3D Caveats Data art

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

# Outlier Samples



- When is something an outlier?
- If it doesn't make sense, you can:
  - Cap them at a reasonable threshold
  - Transform them (e.g., log transformation).

- An outlier is a sample or a single datapoint
- An outlier may be a natural (extreme) variation
- ... or it may arise from errors in:
  - Data collection
  - Lab experiment
  - Database/Registry Entry
  - A natural (extreme) variation

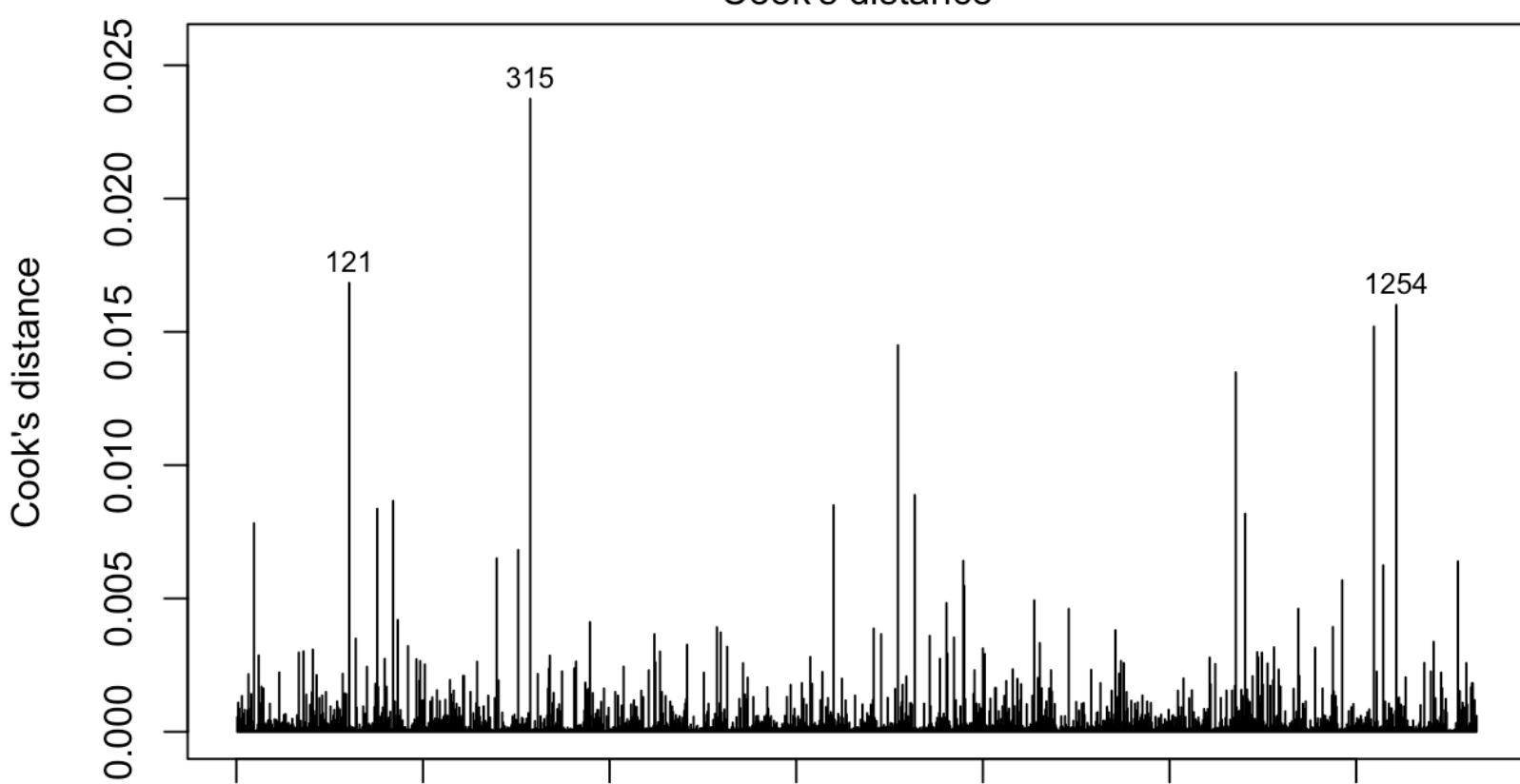
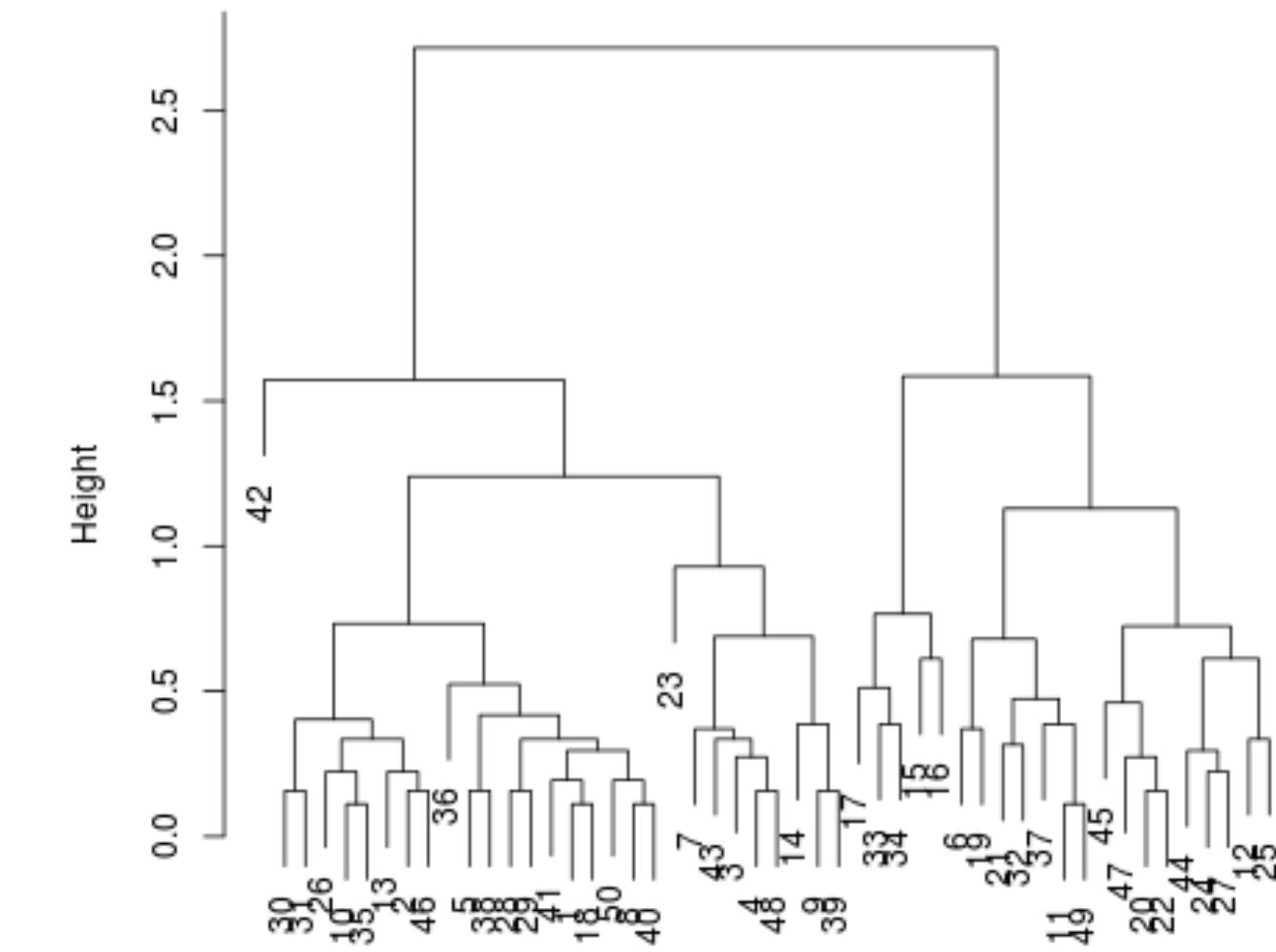


# Outliers - Visual Inspection & Stats Test

- **Visualization:**
  - Distributional plots
  - Clustering Dendogram
  - Principal Component Analysis (PCA)
- **Scores and tests:**
  - Cooks Distance
  - Z-score / Standard score
  - Interquartile range (IQR)
  - Robust Mahalanobis Distance
  - Local Outlier Factor (LOF)

## IMPORTANTLY:

- Check the assumptions and aim of a method!
- Use common sense and your critical thinking skills



# Missing Values

## Missing values:

- Filter out some/all missing values
- Numerical columns: Replace with mean, median, or mode.
- Categorical columns: Use the most frequent category or introduce a new “Unknown” category.
- Impute missing values (Regression, Maximum Likelihood, ML)



# Types of Missing Values

## MAR:

- Missing data point depends on observed data, but not on the missing value itself
- I.e. health records in lower median income countries

**Missing at Random  
(MAR)**

**Missing Completely  
at Random  
(MCAR)**

## MCAR:

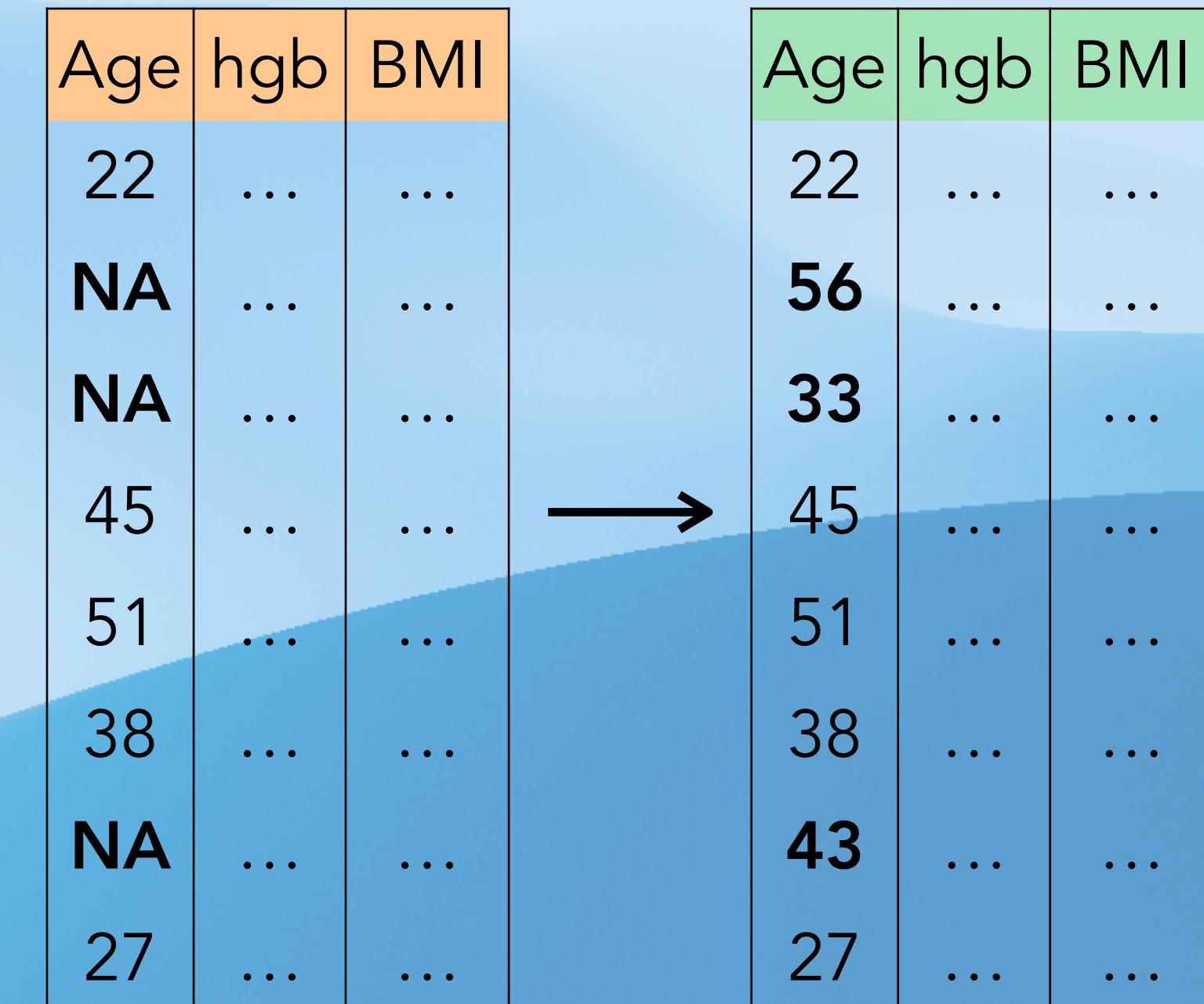
- Data points have the same probability of being missing
- I.e. lab tool error

## MNAR:

- Missing data point depends on the missing value itself
- I.e. patient with depression skip mental health surveys

# Missing Value Imputation

- Why impute and not just remove?
- What type of missing that can be imputed:
  - MCAR = Yes
  - MAR = Yes (be careful, include the right predictors!)
  - MNAR = No! Missingness is related to the unobserved data
- Methods:
  - K-Nearest Neighbors (KNN)
  - Multiple Imputation by Chained Equations (MICE)
  - Probability Methods (Maximum Likelihood)
  - ML methods: Generative Models (VAEs, GANs)



The diagram illustrates the process of missing value imputation. On the left, a sparse matrix shows three rows of data for variables Age, hgb, and BMI. The first row has values 22, ..., ... for Age, hgb, and BMI respectively. The second row has NA, ..., ... for the same variables. The third row has NA, ..., ... for the same variables. An arrow points from this sparse matrix to a denser matrix on the right. In the denser matrix, the first row now has values 22, ..., ... for Age, hgb, and BMI. The second row has 56, ..., ... for Age, hgb, and BMI. The third row has 33, ..., ... for Age, hgb, and BMI. The fourth row has 45, ..., ... for Age, hgb, and BMI. The fifth row has 51, ..., ... for Age, hgb, and BMI. The sixth row has 38, ..., ... for Age, hgb, and BMI. The seventh row has NA, ..., ... for Age, hgb, and BMI. The eighth row has 27, ..., ... for Age, hgb, and BMI.

Age	hgb	BMI
22	...	...
NA	...	...
NA	...	...
45	...	...
51	...	...
38	...	...
NA	...	...
27	...	...

→

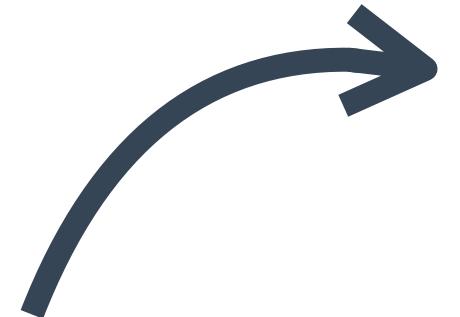
Age	hgb	BMI
22	...	...
56	...	...
33	...	...
45	...	...
51	...	...
38	...	...
43	...	...
27	...	...

# Long and wide format

```
tree_long <- tree_wide %>%
  pivot_longer(cols = starts_with("Site"),
               names_to = "Site",
               values_to = "Average diameter (cm)")
```

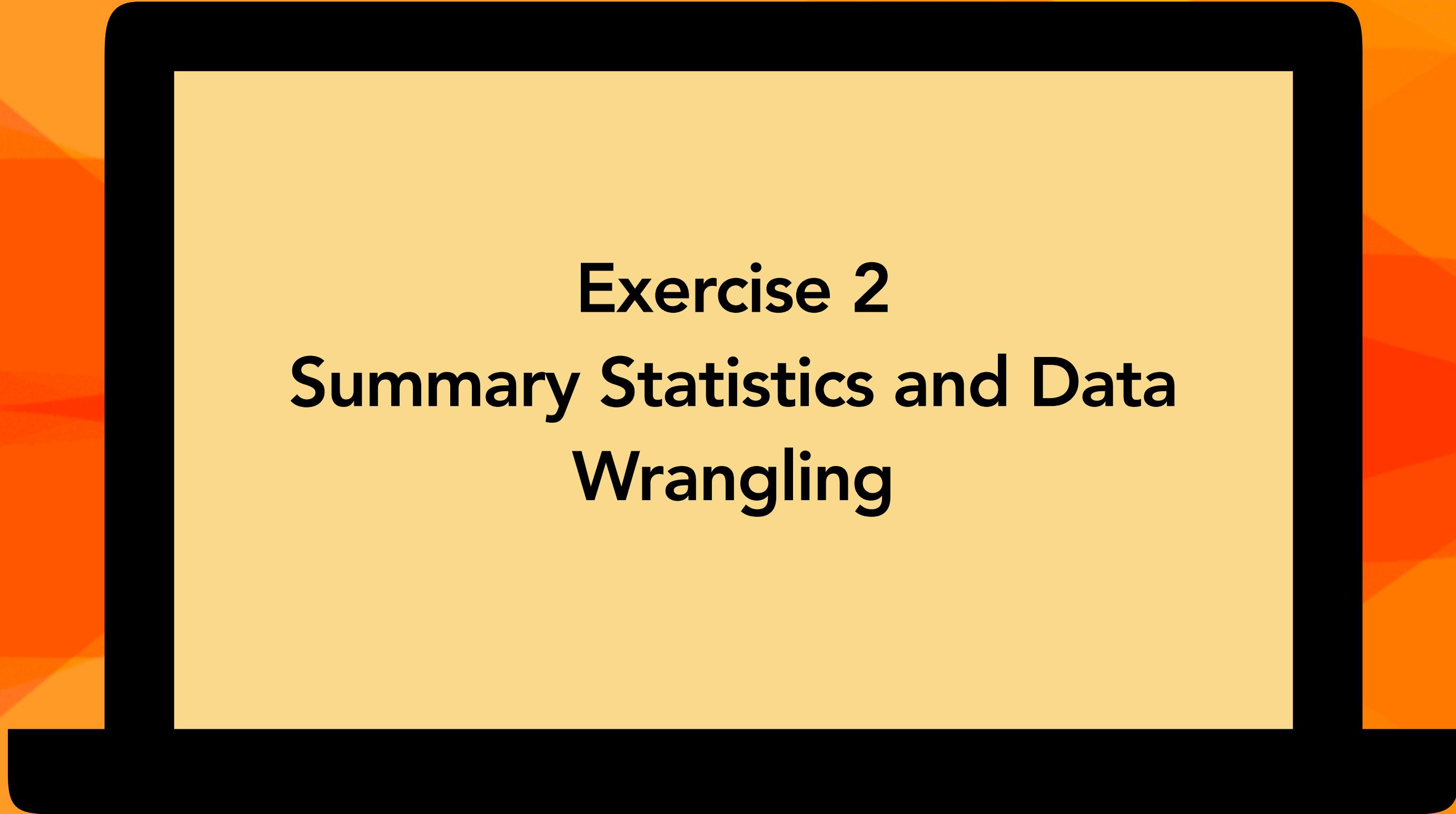
Type species	Site A	Site B	Site C	Site D
Acer rubrum	15	8	30	27
Quercus alba	29	17	14	42
Pinus teada	10	19	25	23

```
tree_wide <- tree_long %>%
  pivot_wider(names_from = Site,
              values_from = `Average diameter (cm)`)
```



Tree species	Site	Average diameter (cm)
Acer rubrum	Site A	15
Acer rubrum	Site B	8
Acer rubrum	Site C	30
Acer rubrum	Site D	27
Quercus alba	Site A	29
Quercus alba	Site B	17
Quercus alba	Site C	14
Quercus alba	Site D	42
Pinus teada	Site A	10
Pinus teada	Site B	19
Pinus teada	Site C	25
Pinus teada	Site D	23





## **Exercise 2**

### **Summary Statistics and Data Wrangling**

# **Part 3**

# **Exploratory Data Analysis**

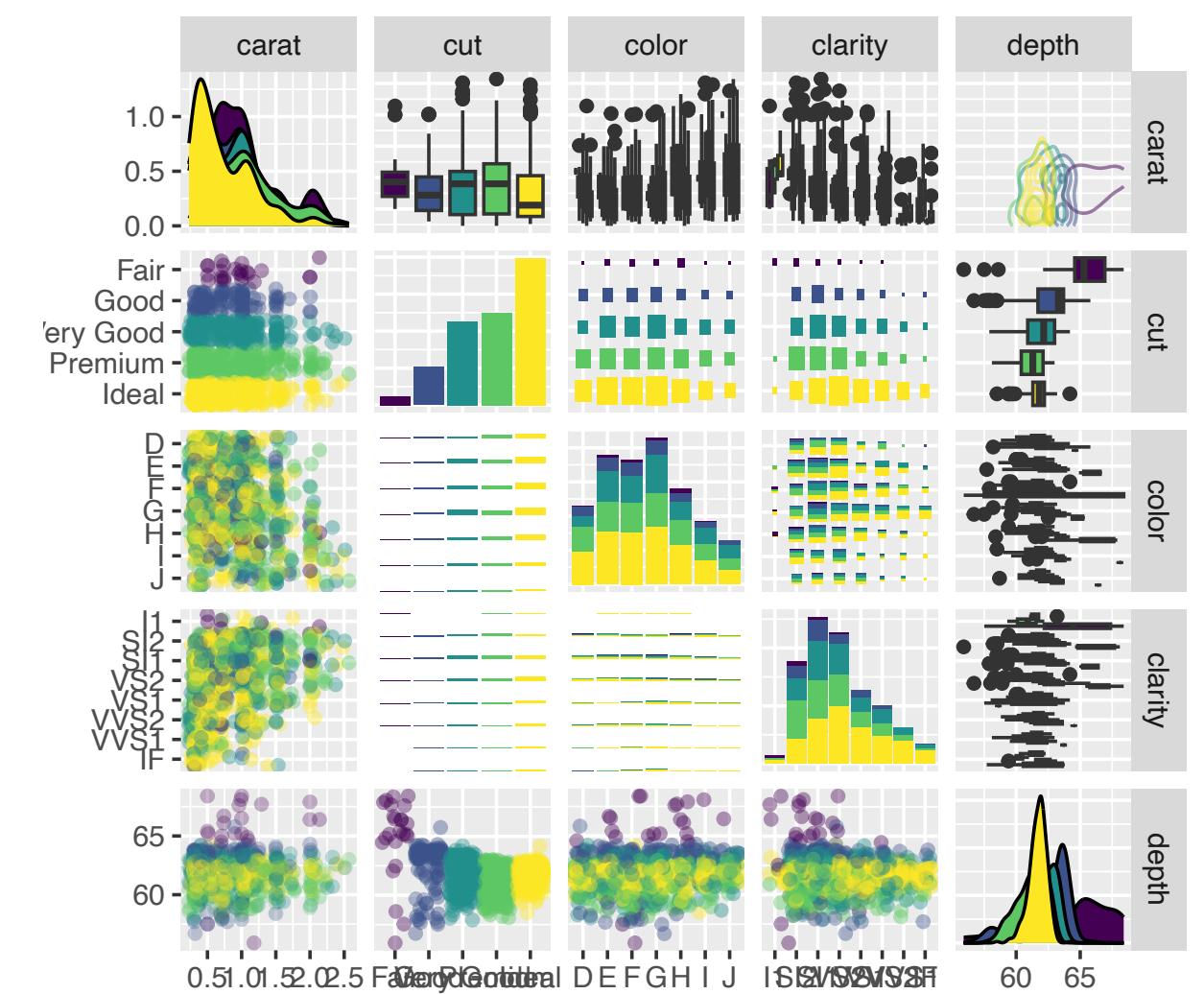
# Why Multivariate EDA

- The Challenge of Complexity (Reality of High-Dimensional Data)

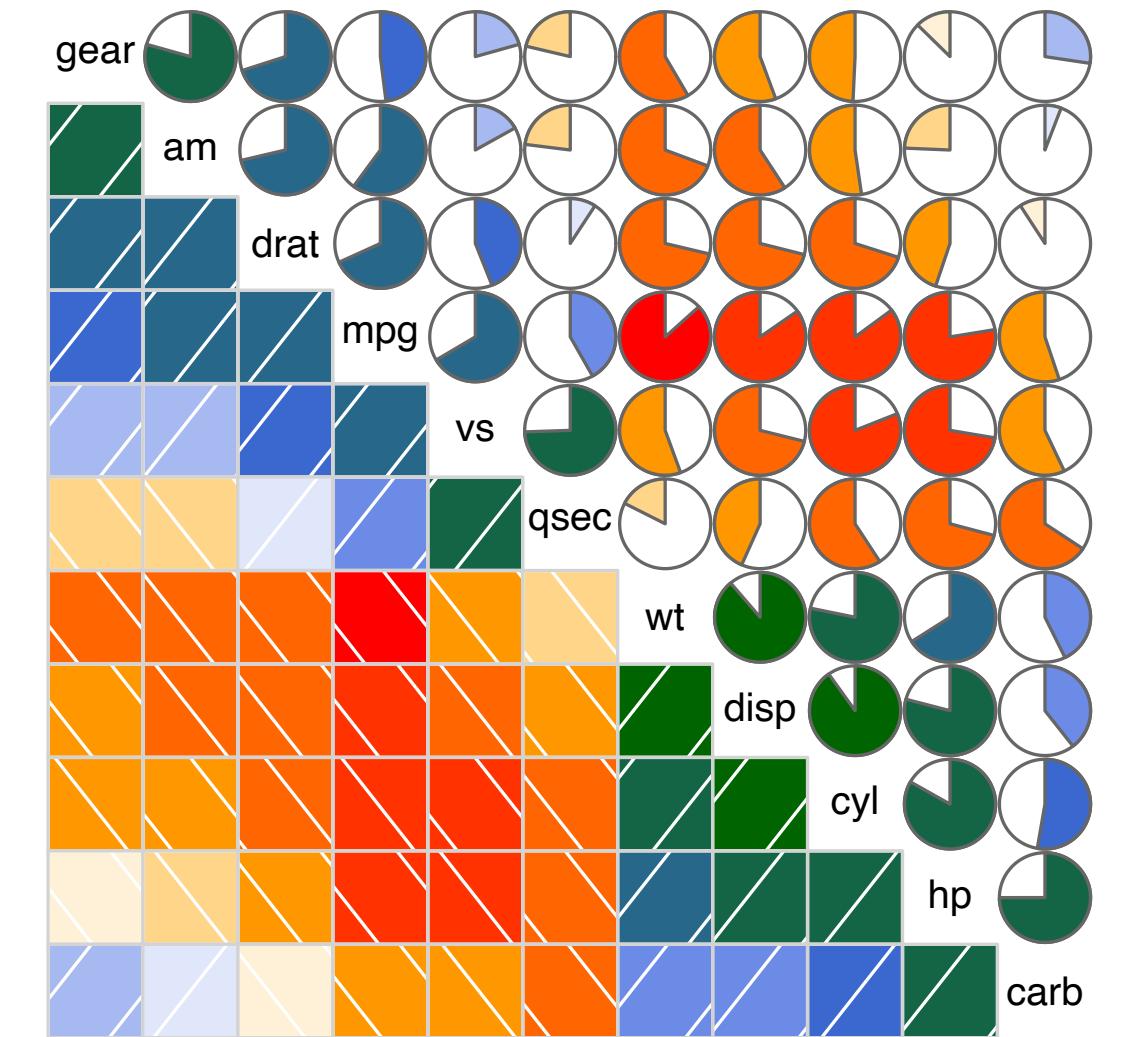
- As the number of variables increases, especially if you have dozens of gene expression values, categorical groups, and clinical variables - simple visualizations become:

- Trends become hard to interpret

Scatter plots crowded



Correlation plots dense and overwhelming

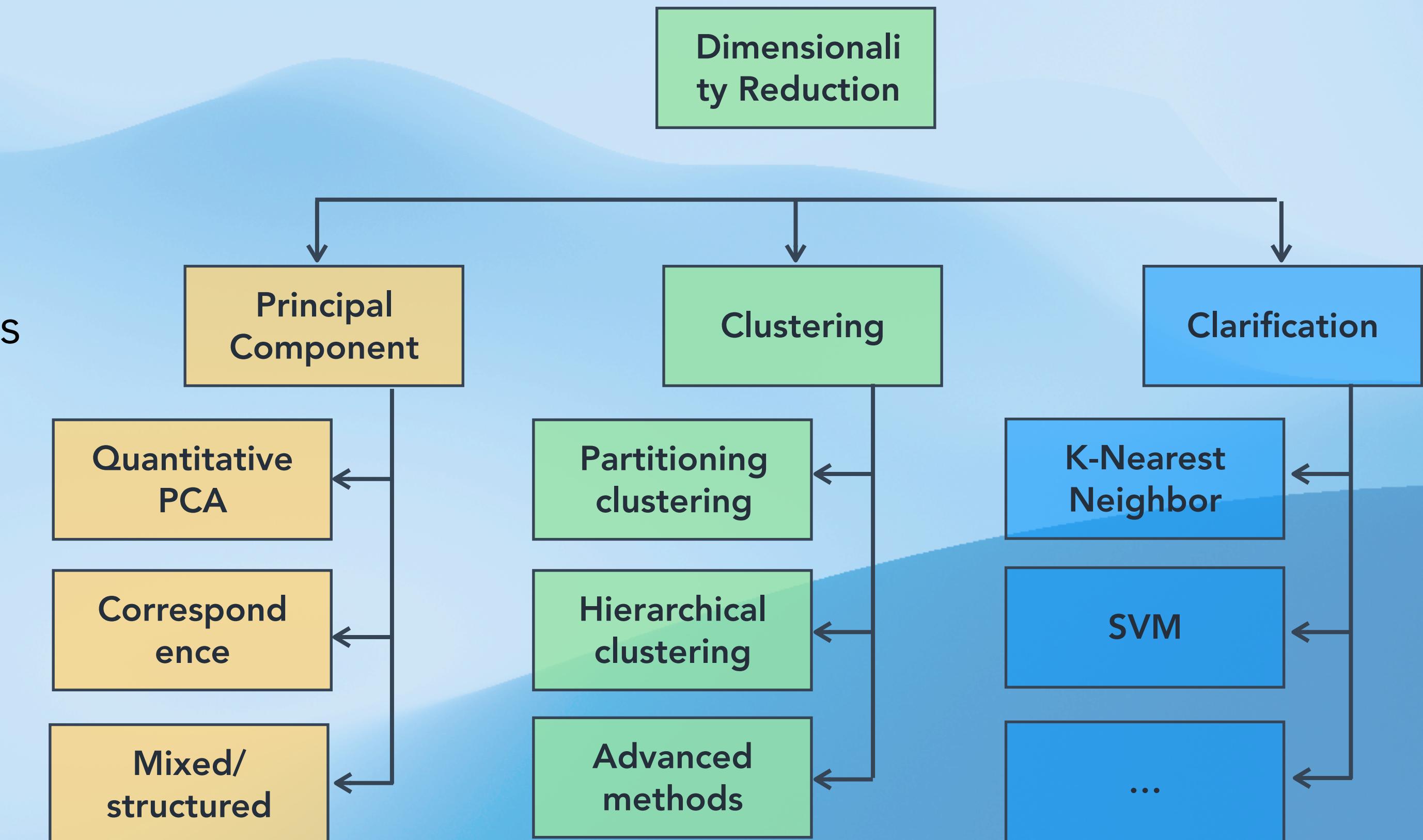


# Dimensionality Reduction

When visual exploration becomes too noisy, we need to simplify:

These help compress information, preserve structure, and reveal key drivers in the data.

- Principal Component Methods
- Clustering
- Classification
- Regressions



# Data Transformation Techniques

**Real-world data is often messy:** skewed, non-linear, or heteroscedastic (i.e., variance changes with the mean).

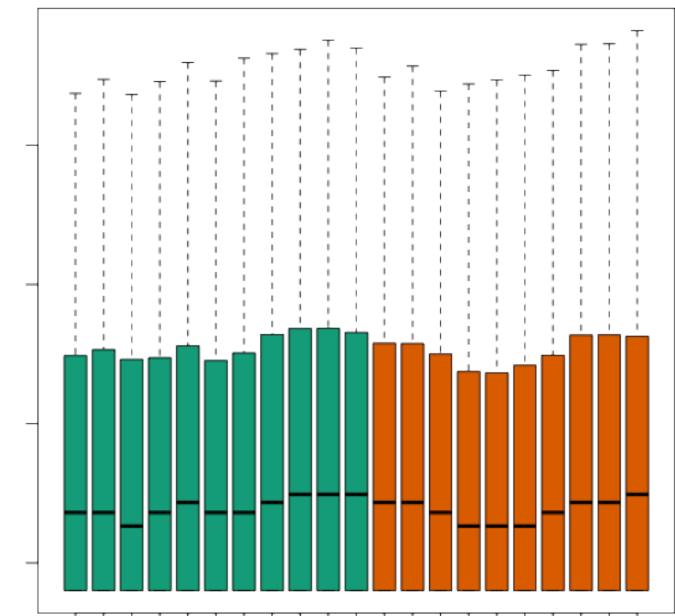
Depending on our data's characteristics and goals, we may need to

- Scale
- Transform
- Stabilize variance

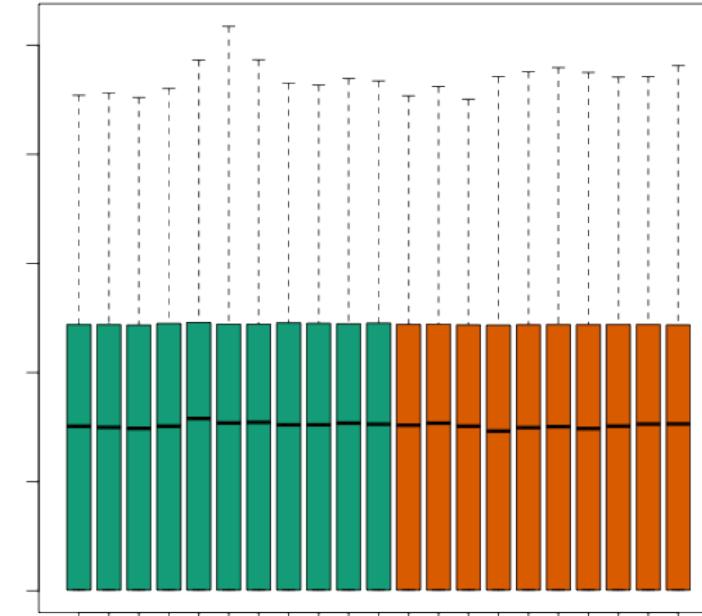
To accurate analysis and modeling.

## Normalization

Raw Samples

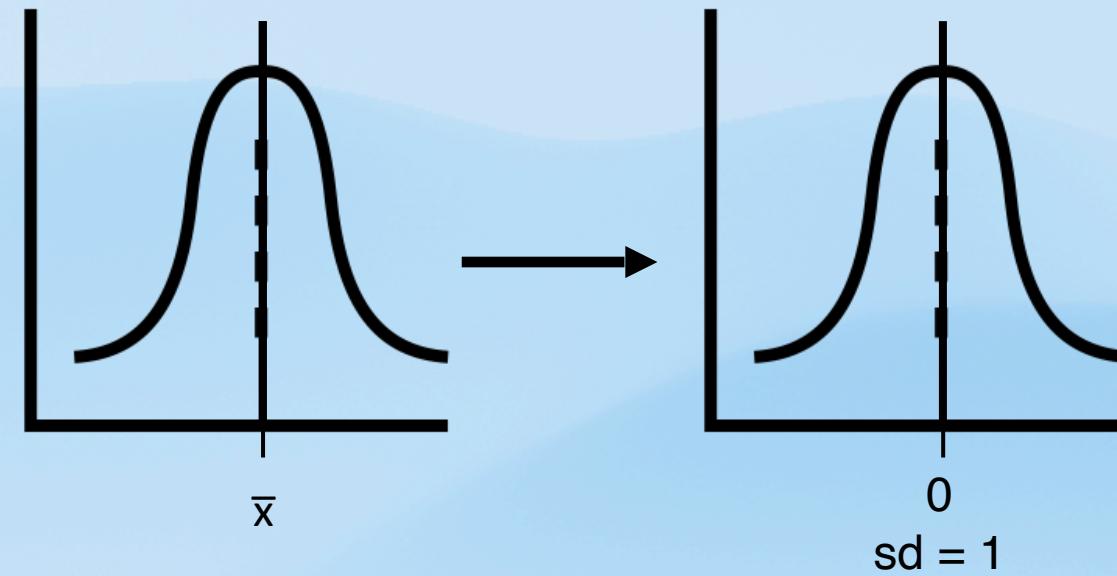


Normalized Samples



- Can use for not normally distributed data.
- Variables do not get zero centered.
- Normalization within a range (max, min).
- Affected by outliers.

## Standardization

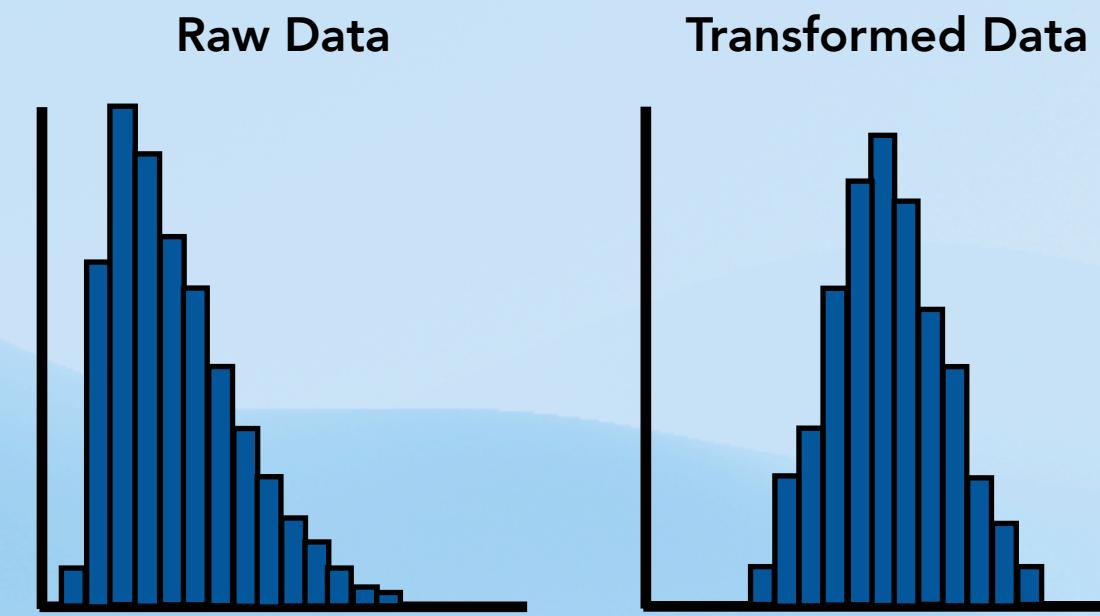


- For normally distributed data
- Not constrained to a range.
- Not affected by outliers.

**Min-Max scaling** - Rescales variables to 0–1 range

**Z-score scaling** - Rescales variables to mean 0 and SD 1

## Transformation



- Attempt to make data normally distributed - reduce skewness
- Squeezes outliers for less impact on model.

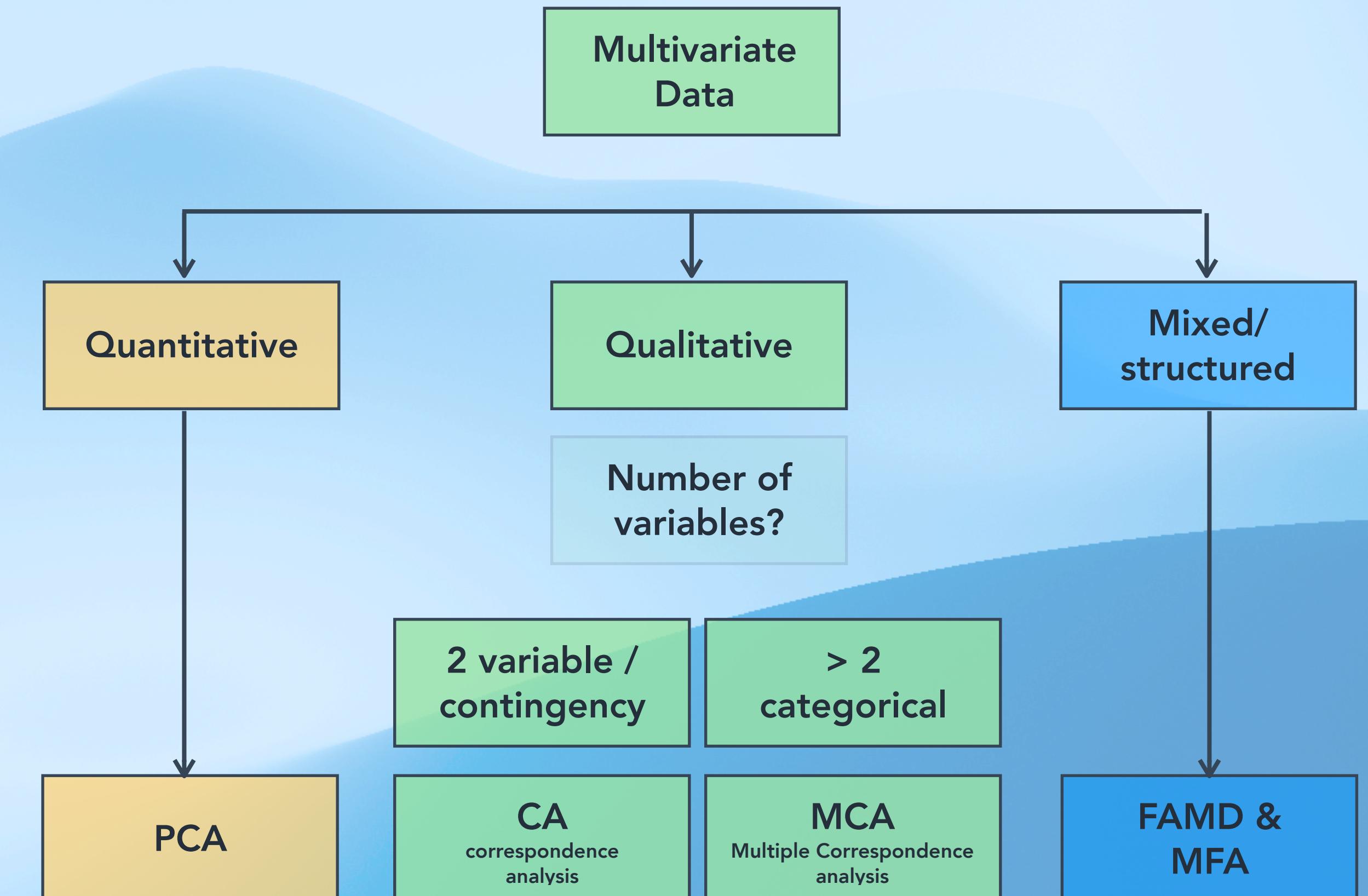
**Log Log2 Log10** - Compresses large values, reduces right skew

**Square-root** - Similar to log but gentler

**Square/power** - Expands right tail, reduces left skew.  
For left-skewed variables

# Principal Component Methods

- Classical methods
  - PCA, CA and MCA
  - (continuous variables, contingency table and qualitative variables, respectively).
- Advanced methods -
  - FAMD, MFA and HMFA -
  - a mix of variables (qualitatives & quantitatives)

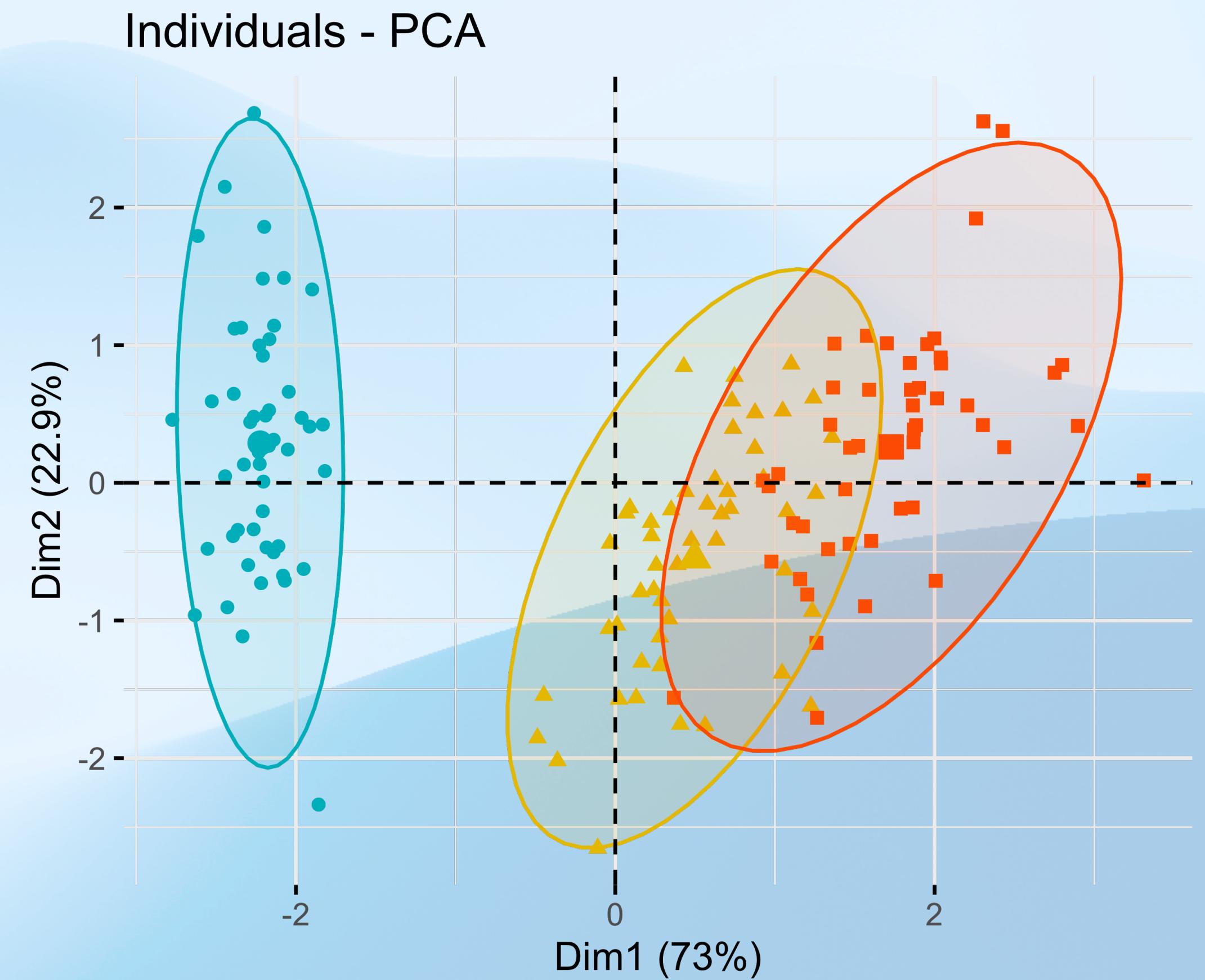


# Principal Component Analysis (PCA)

- Linear dimensionality reduction:
- High → Low dimensional space
- Find new dimensions that capture as much variation as possible
- Shows structure of the data across all variables

	Alc.	Alkalini	Flabon	Color	...
Indiv1	1.51	-1.17	1.03	0.25	...
Indiv2	0.25	-2.48	0.73	-0.29	...
Indiv3	1.50	-0.96	0.97	0.57	...
...	...	...	...	...	...

Good and simple guide to PCA in R: [https://cran.r-project.org/web/packages/ggfortify/vignettes/plot\\_pca.html](https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html)

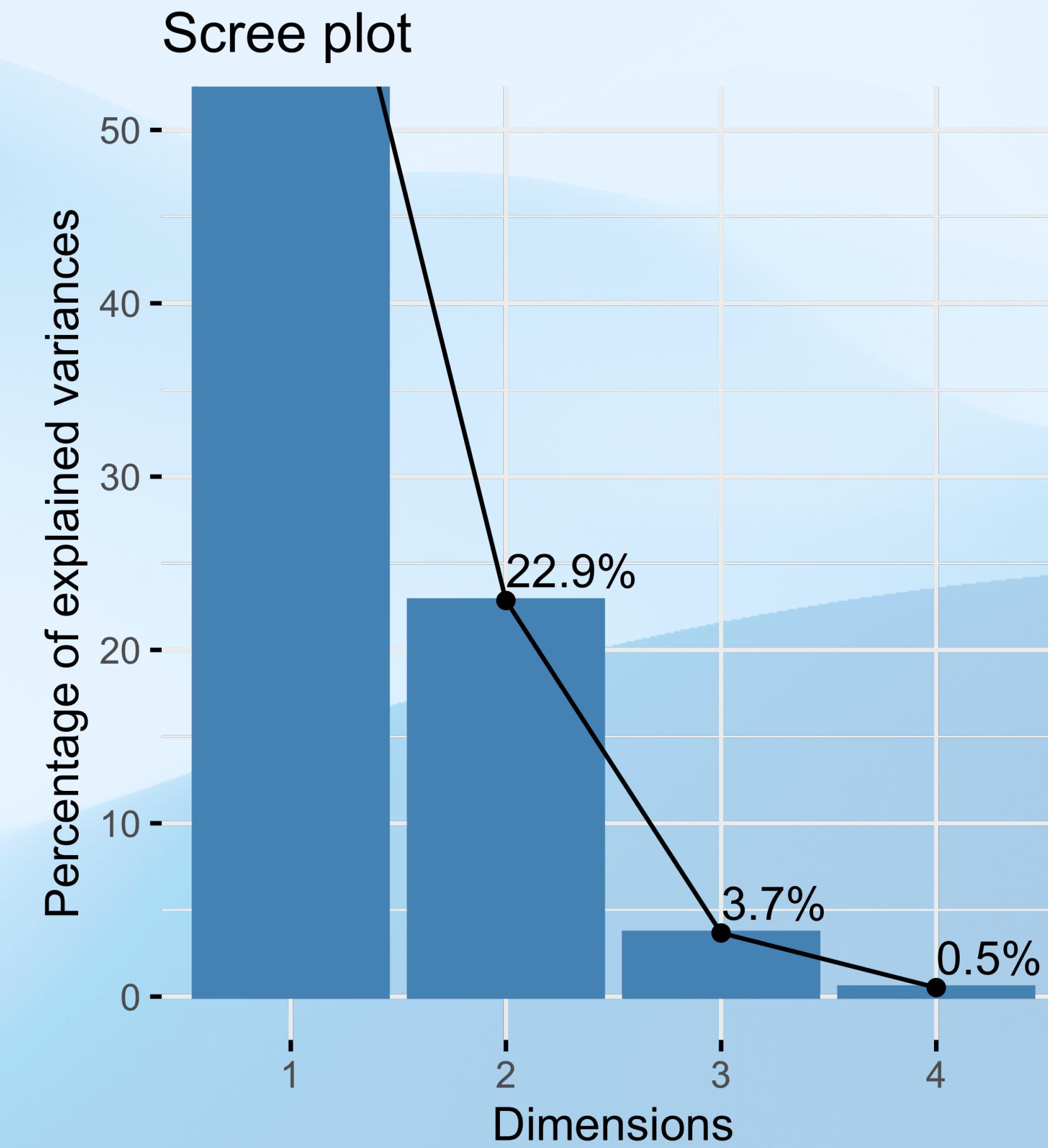


# Principal Component Analysis (PCA)

- Linear dimensionality reduction:
- High → Low dimensional space
- Find new dimensions that capture as much variation as possible
- Shows structure of the data across all variables

	Alc.	Alkalini	Flabon	Color	...
Indiv1	1.51	-1.17	1.03	0.25	...
Indiv2	0.25	-2.48	0.73	-0.29	...
Indiv3	1.50	-0.96	0.97	0.57	...
...	...	...	...	...	...

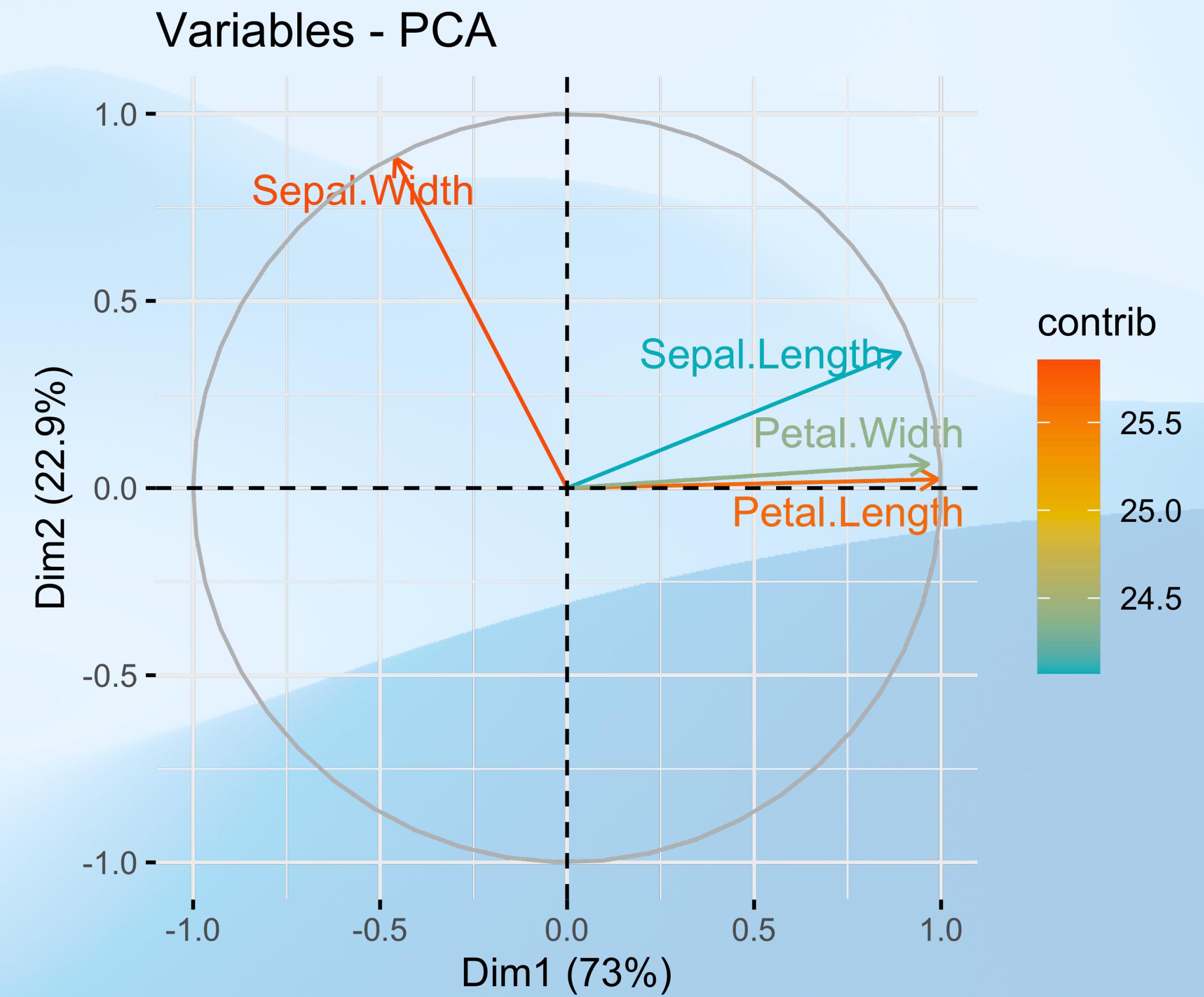
Good and simple guide to PCA in R: [https://cran.r-project.org/web/packages/ggfortify/vignettes/plot\\_pca.html](https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html)



# Biplot: PCA plot with loadings

- Loadings: Arrows represent the direction and magnitude of each variable's contribution to the PC's.
- The length of the arrow indicates the strength of the variable's contribution.
- The direction of the arrow shows how the variables correlates with each PC.

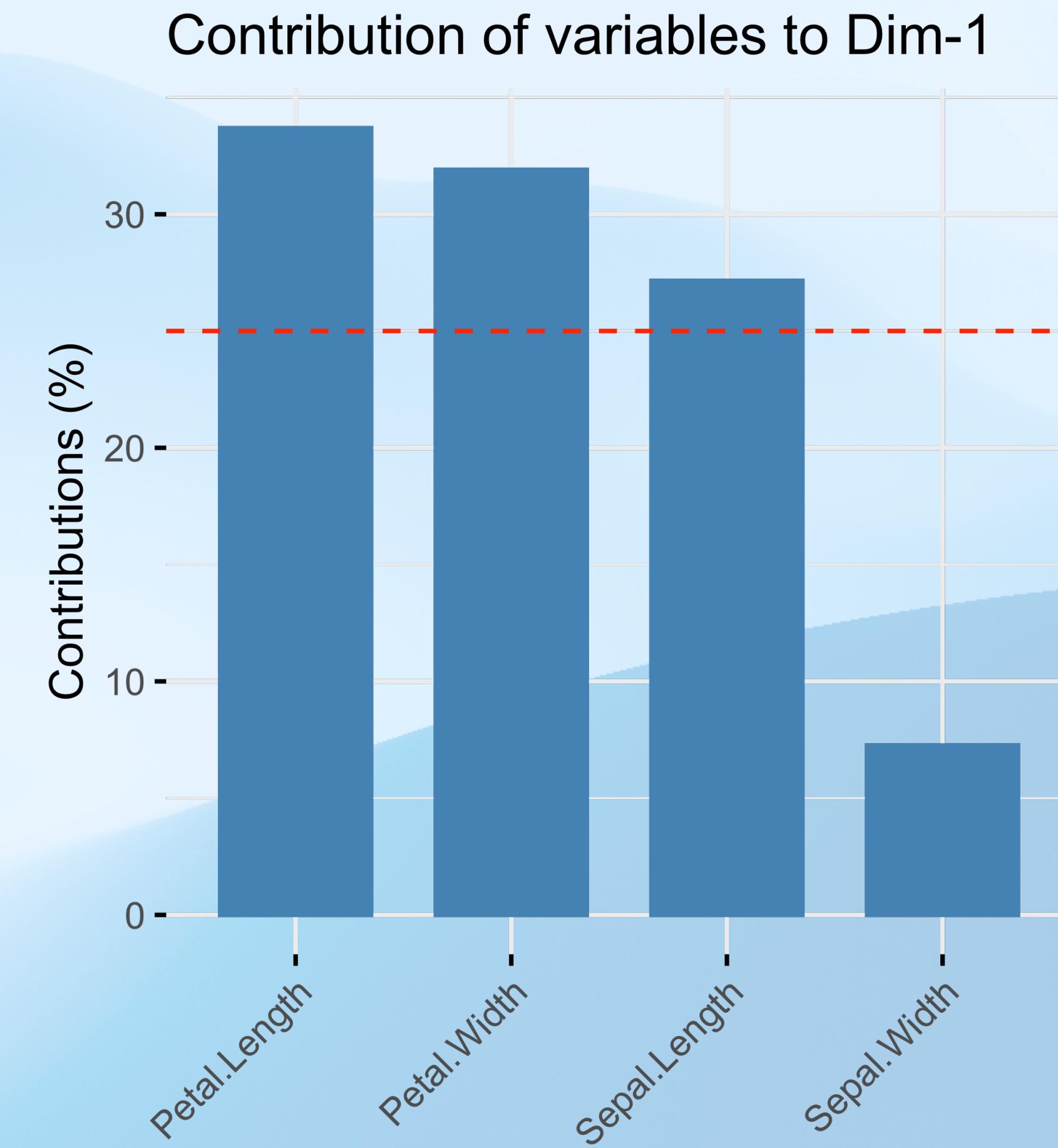
	PC1	PC2
Sepal Length	+	-
Sepal Width	-	-
Petal Length	+	-
Petal Width	+	-



# Biplot: PCA plot with loadings

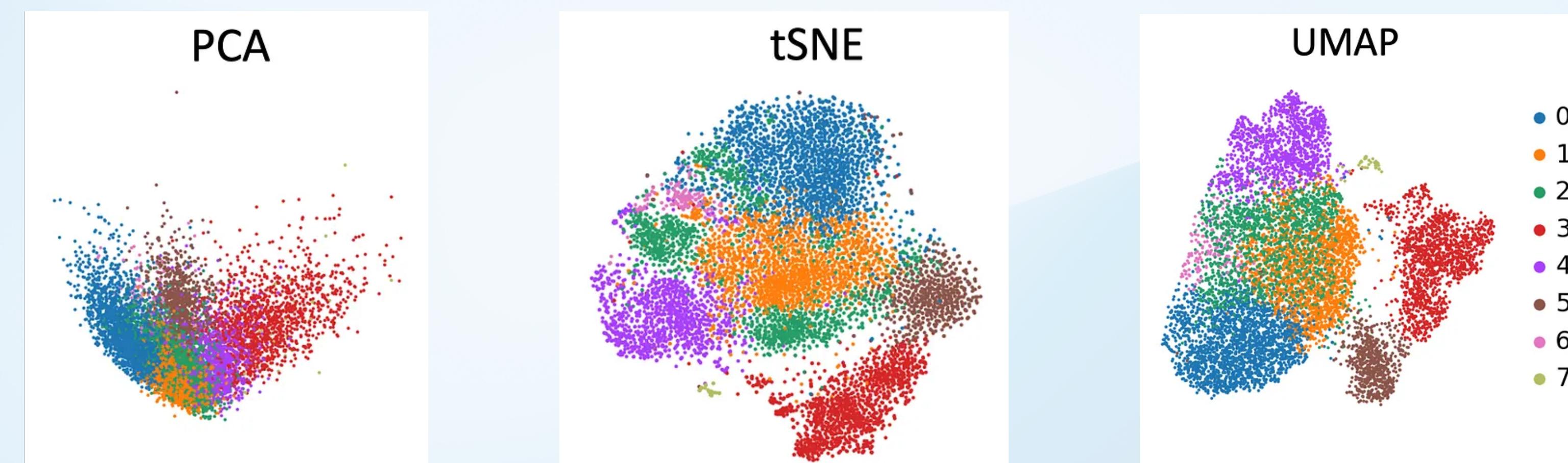
- Loadings: Arrows represent the direction and magnitude of each variable's contribution to the PC's.
- The length of the arrow indicates the strength of the variable's contribution.
- The direction of the arrow shows how the variables correlates with each PC.

	PC1	PC2
Sepal Length	+	-
Sepal Width	-	-
Petal Length	+	-
Petal Width	+	-



# Correlation Clustering

- DR methods are unsupervised. They can be either linear or non-linear.
- Linear projection: PCA = principal component analysis
- Non-linear projection:
- UMAP = Uniform Manifold Approximation and Projection
- t-SNE = t-distributed stochastic neighbor embedding



# **Exercise 3**

## **Exploratory Data Analysis**

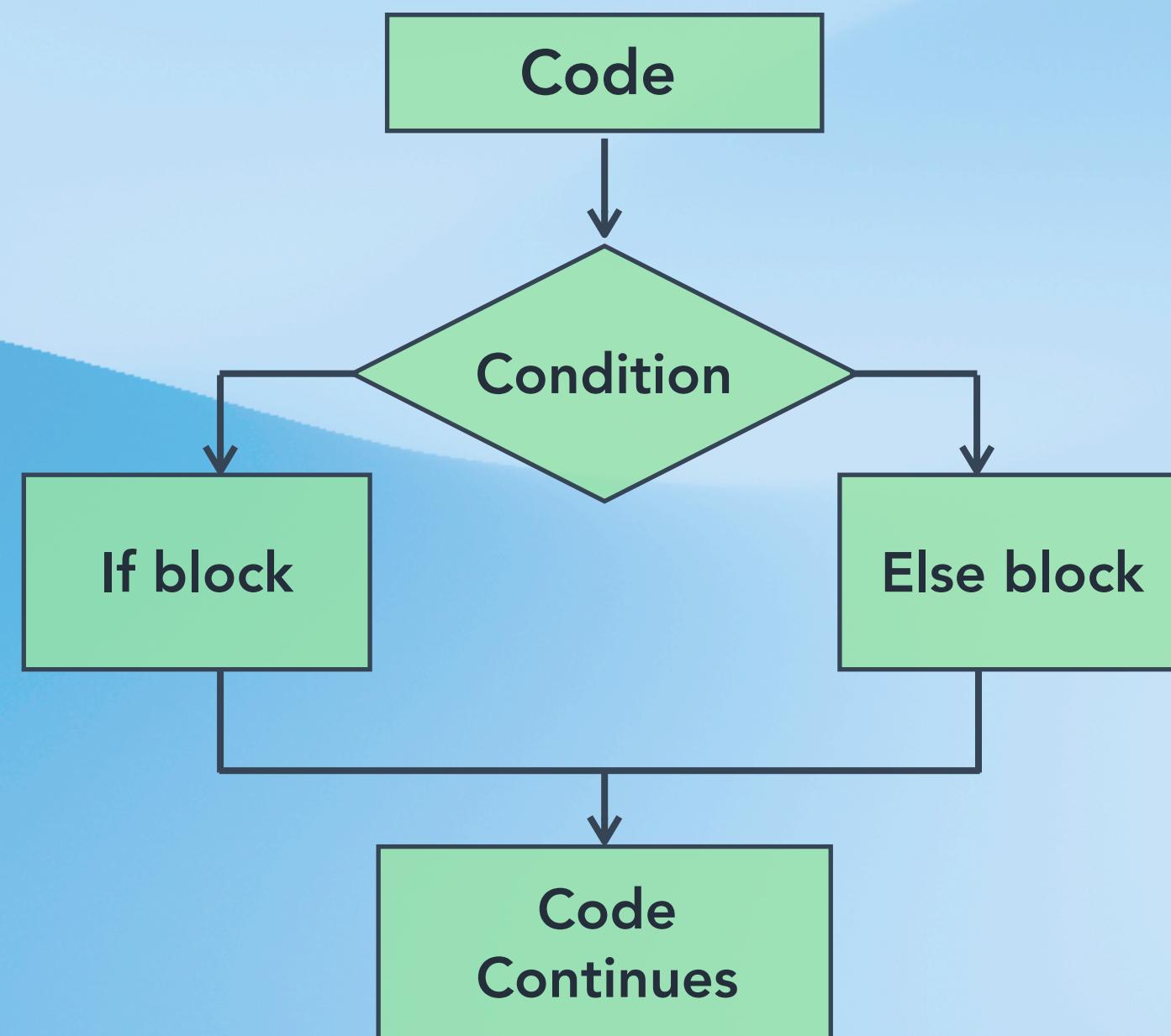
# **Part 4A**

# **Conditions and For-loops**

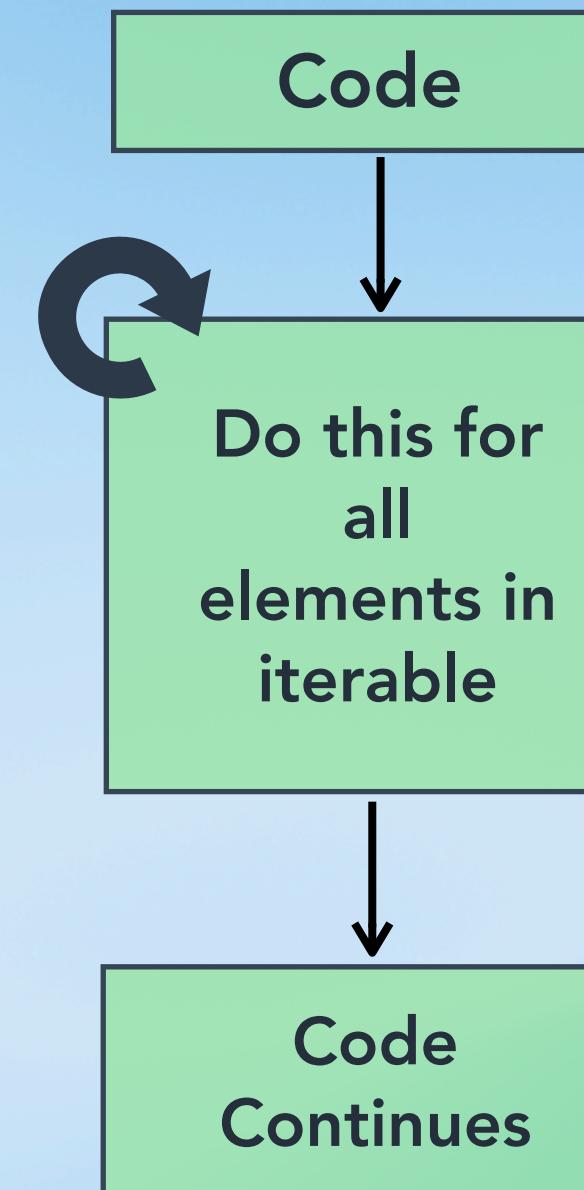


# Scripting in R

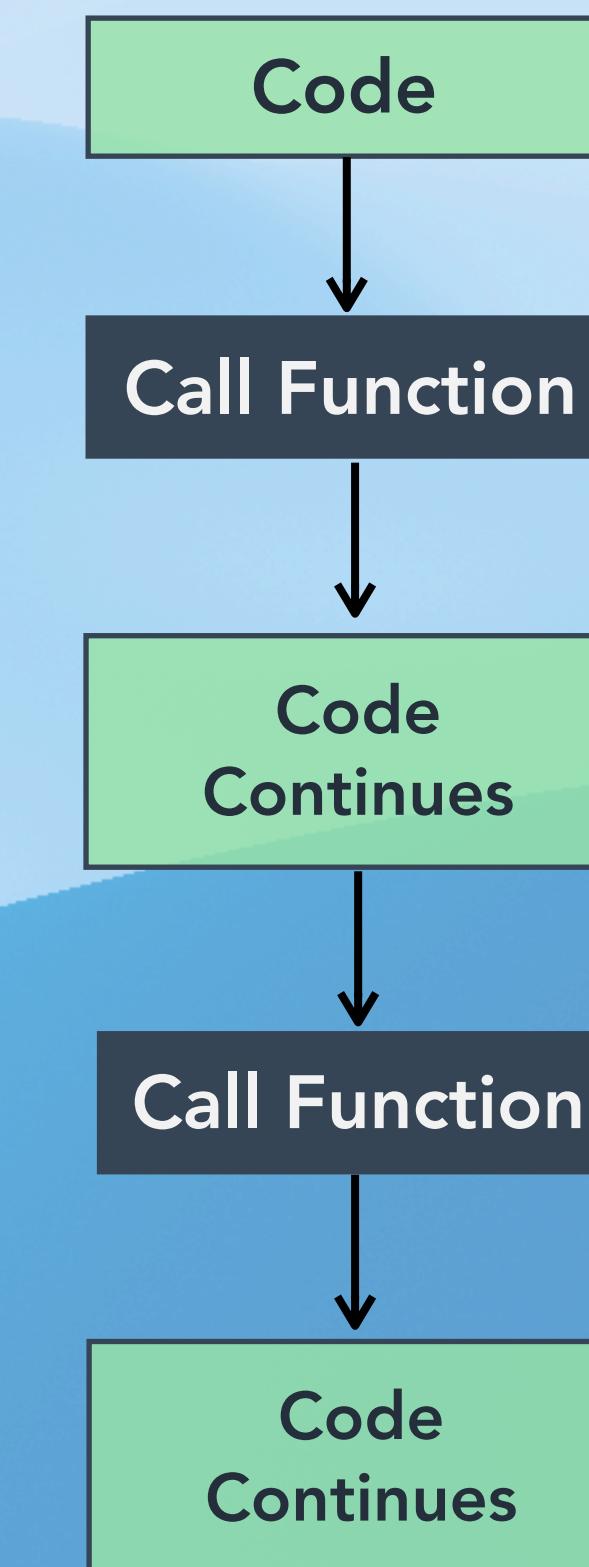
## Conditionals



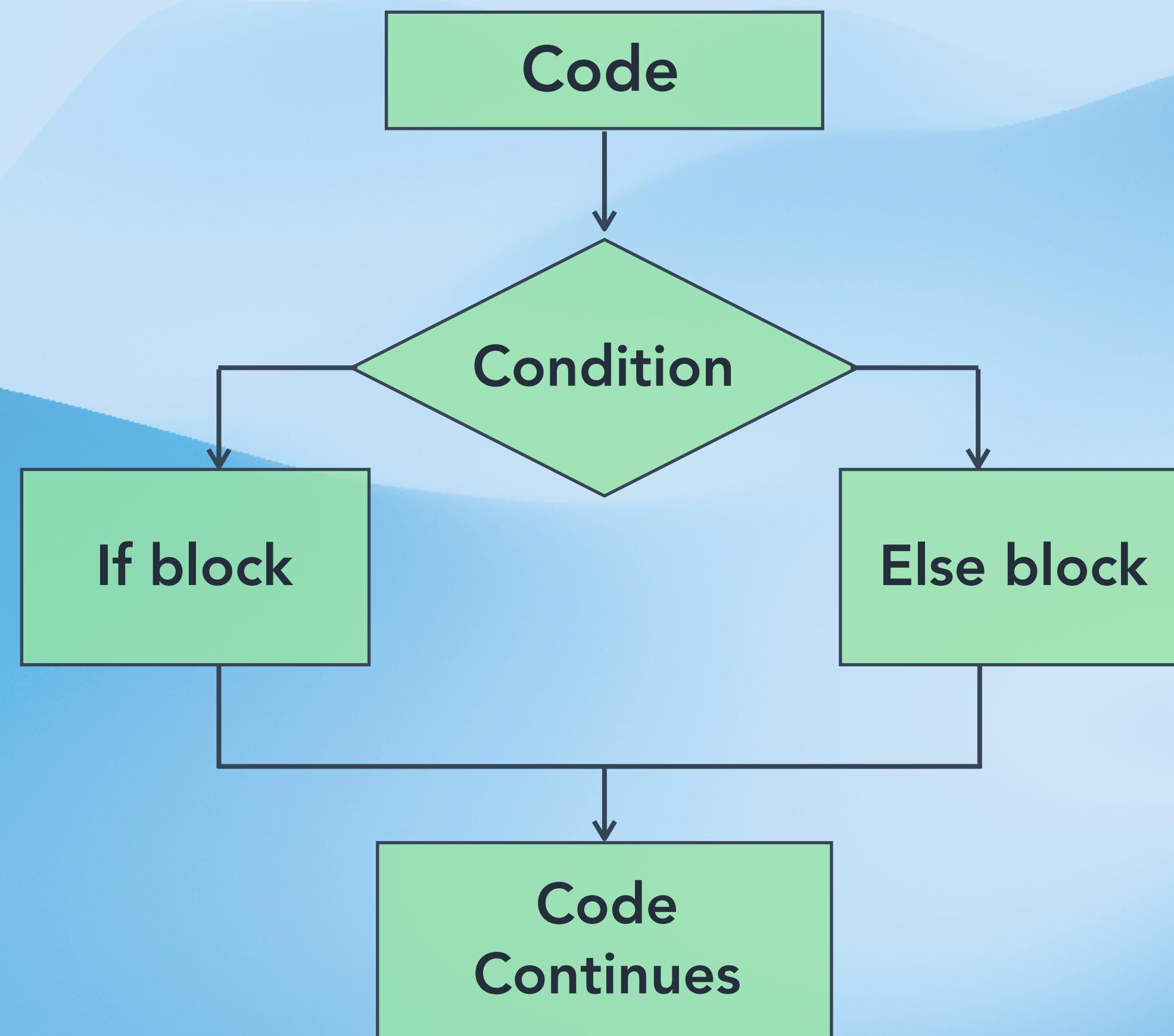
## Loop



## Functions



# Conditionals (if-else statements)



Symbol	Meaning
> , <	Greater, Smaller
==	Equal to
!	Not
%in%	Occurs in
is.[type]	Check variable type
&	And
	Or

Conditions can be complex:

```
if (age > 30 & !(smoker) | hospital == 'Herlev' )
```

# Conditionals (if-else statements)

- Conditional logic: Execute code blocks based on whether a condition is TRUE or FALSE.

## R syntax

```
if (CONDITION_1) {  
    OPERATION_1  
} else if (CONDITION_2) {  
    OPERATION_2  
} else if (CONDITION_3) {  
    OPERATION_3  
} else {  
    OPERATION_4  
}
```

## Common conditions

var1 > var2

var1 < var2

var1 == var2

var1 != var2 # Not equal to

var1 - var2 == var3

is.numeric(var1), is.character(var1), ...

## Combine conditions

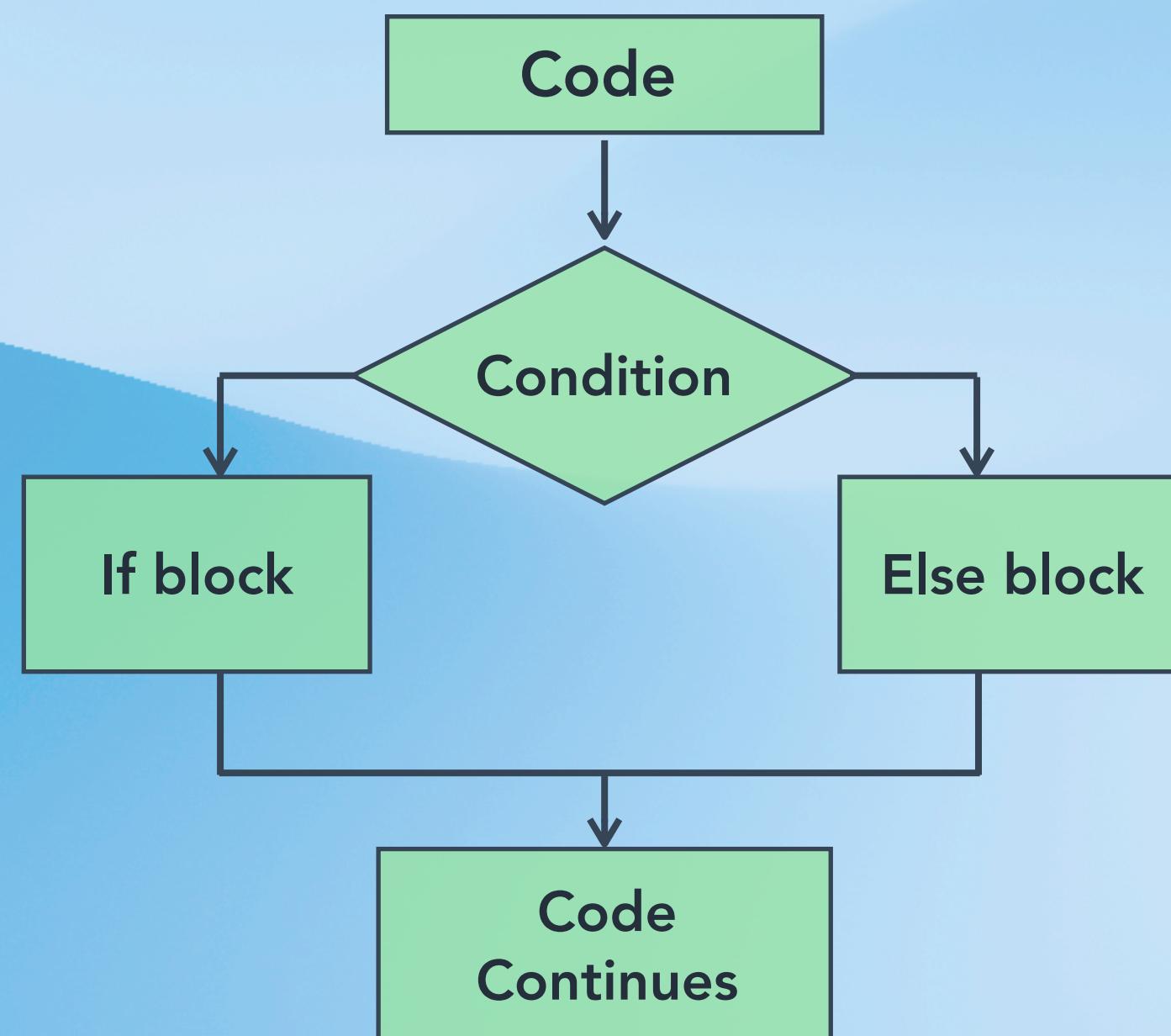
var1 > var2 & var1 < var3

var1 == var2 | var1 == var3

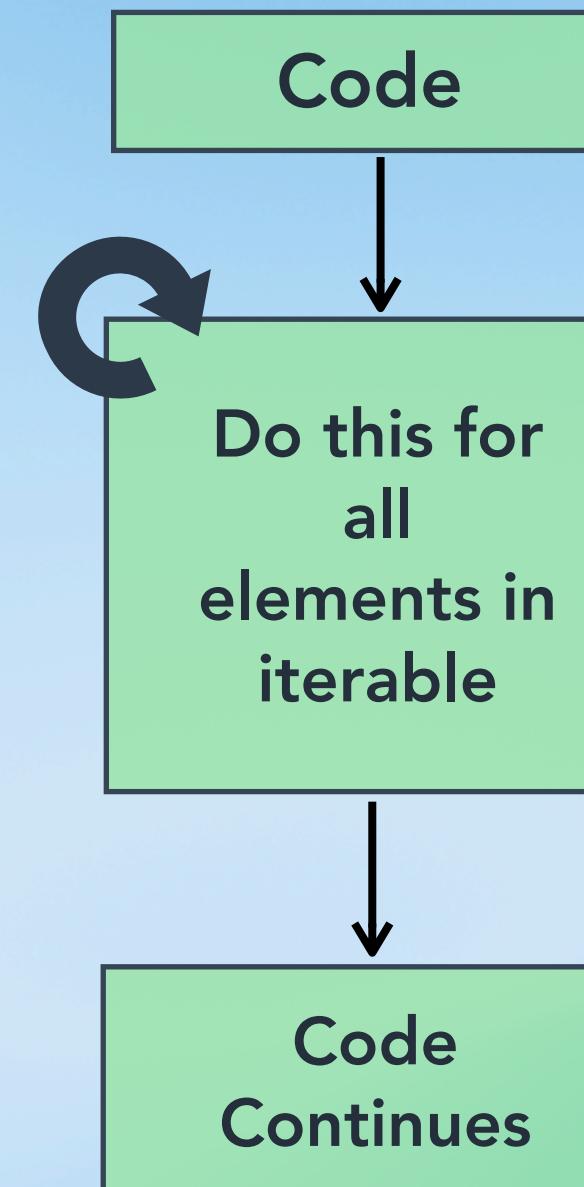
(var1 > var2 & var1 < var3) | (var1 == var2)

# Scripting in R

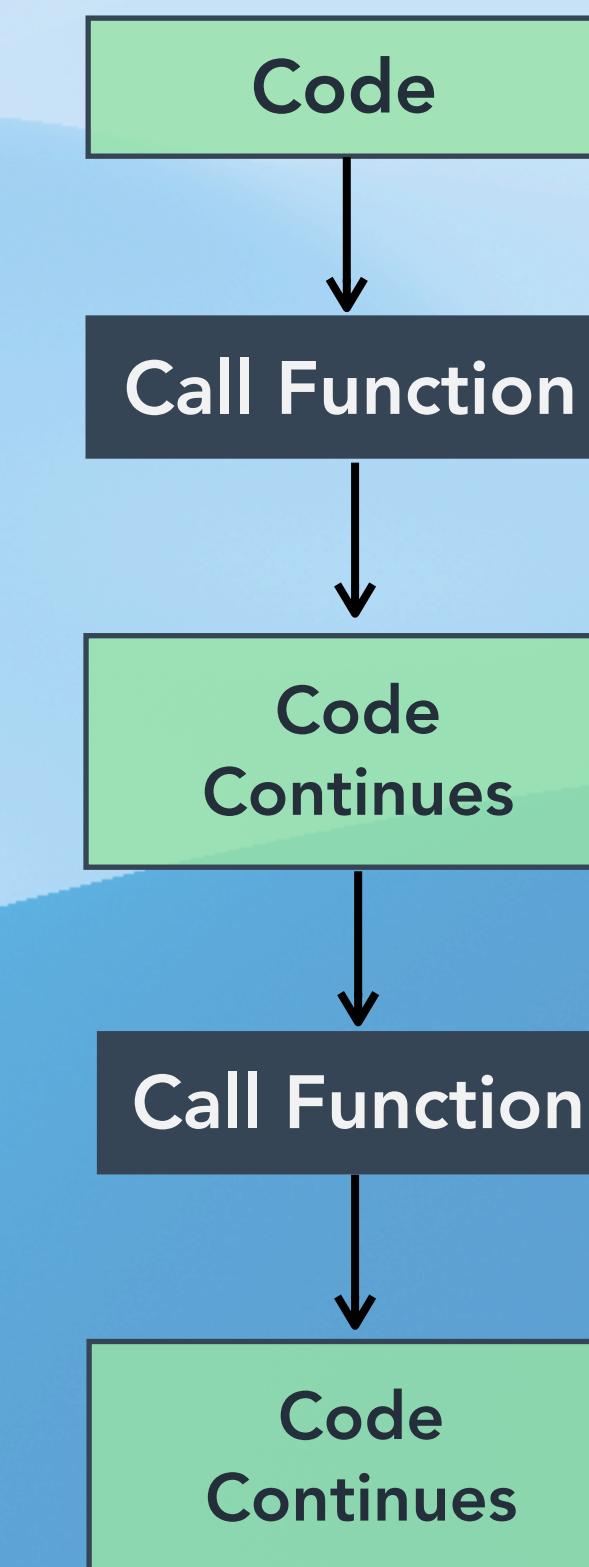
## Conditionals



## Loop



## Functions



# Looping in R (for-loop)

## Looping in R:

- Repeat a block of code for each item in a sequence.
- Iterating over objects.
- i is often used when looping through indexes.
- Different ways to write a for-loop

## Pseudo code:

```
# var iterates over each item in list.  
for (var in list) {  
    OPERATION_USING_var  
}
```

```
> names <- c("Henrike", "Thilde", 'Anders', "Helene", "Inigo", "Diana")  
>  
> for (i in names) {  
+   print(i)  
+ }  
[1] "Henrike"  
[1] "Thilde"  
[1] "Anders"  
[1] "Helene"  
[1] "Inigo"  
[1] "Diana"
```

---

```
> for (i in 1:length(names)) {  
+   print(names[i])  
+ }  
[1] "Henrike"  
[1] "Thilde"  
[1] "Anders"  
[1] "Helene"  
[1] "Inigo"  
[1] "Diana"
```

# Looping in R (while-loop)

- Loops can be combined with conditionals
- While-loop is powerful BUT might crash your R-session if not written correctly

## Pseudo code:

```
# While condition(s) holds true,  
do something to variable (var)  
  
while (var < x) {  
    OPERATION_USING_var  
}
```

```
> count <- 1  
  
> while (count <= 10) {  
+   print(count/5)  
+   count <- count + 1  
+ }  
[1] 0.2  
[1] 0.4  
[1] 0.6  
[1] 0.8  
[1] 1  
[1] 1.2  
[1] 1.4  
[1] 1.6  
[1] 1.8  
[1] 2
```

```
> count <- 1  
  
> while (count <= 10) {  
+   if (count == 5) {  
+     print("Half way there!")  
+     count <- count + 1  
+   }  
+   print(count/5)  
+   count <- count + 1  
+ }  
[1] 0.2  
[1] 0.4  
[1] 0.6  
[1] 0.8  
[1] "Half way there!"  
[1] 1.2  
[1] 1.4  
[1] 1.6  
[1] 1.8  
[1] 2
```

# Not All R Code is Created Equally

- Looping in R can be slow if data is large!
- Alternative is the family of **apply functions**:  
apply a function to every; value, list element  
or row/column.
- These functions are written in  code - fast!

## Apply Functions

```
# Input is a matrix, specify rows or cols. Output is a list:  
apply(df, 1, FUN = mean)
```

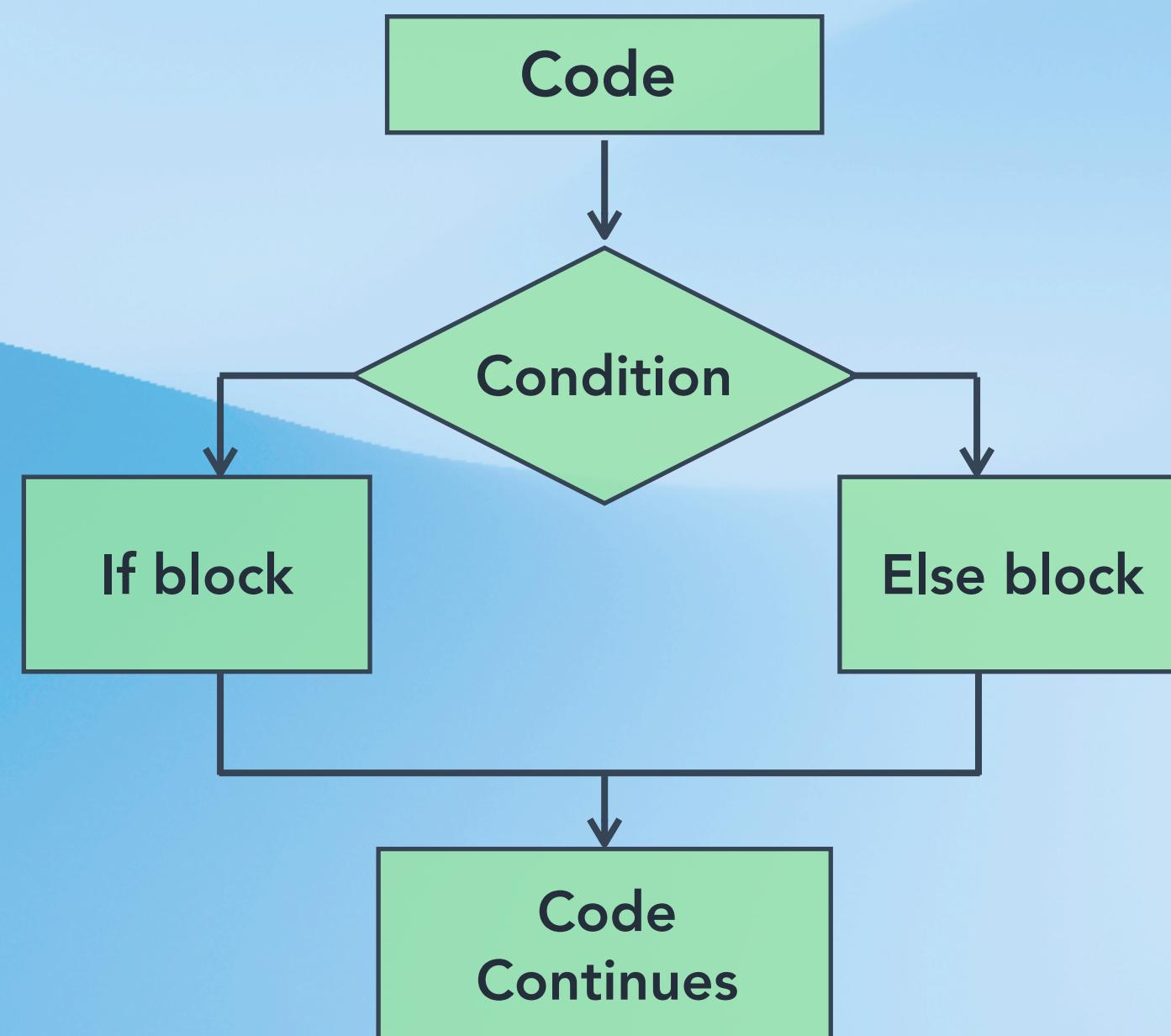
```
# Input is a single list. Output is a list:  
lapply(input_list, FUN = mean)
```

```
# Input is a single list/vector. Output is a vector:  
vapply(input_list, FUN = mean, FUN.VALUE=as.numeric(1))
```

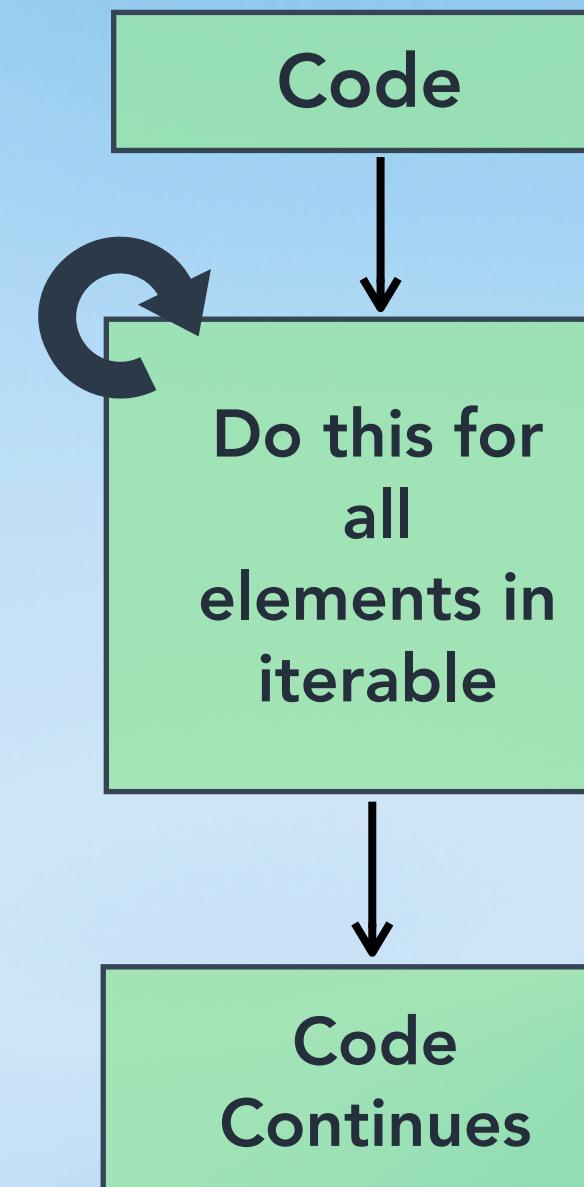
```
# Input is more than one argument. Output is list or vector:  
mapply(function(x,y) mean(c(x,y)), list1, list2)
```

# Scripting in R

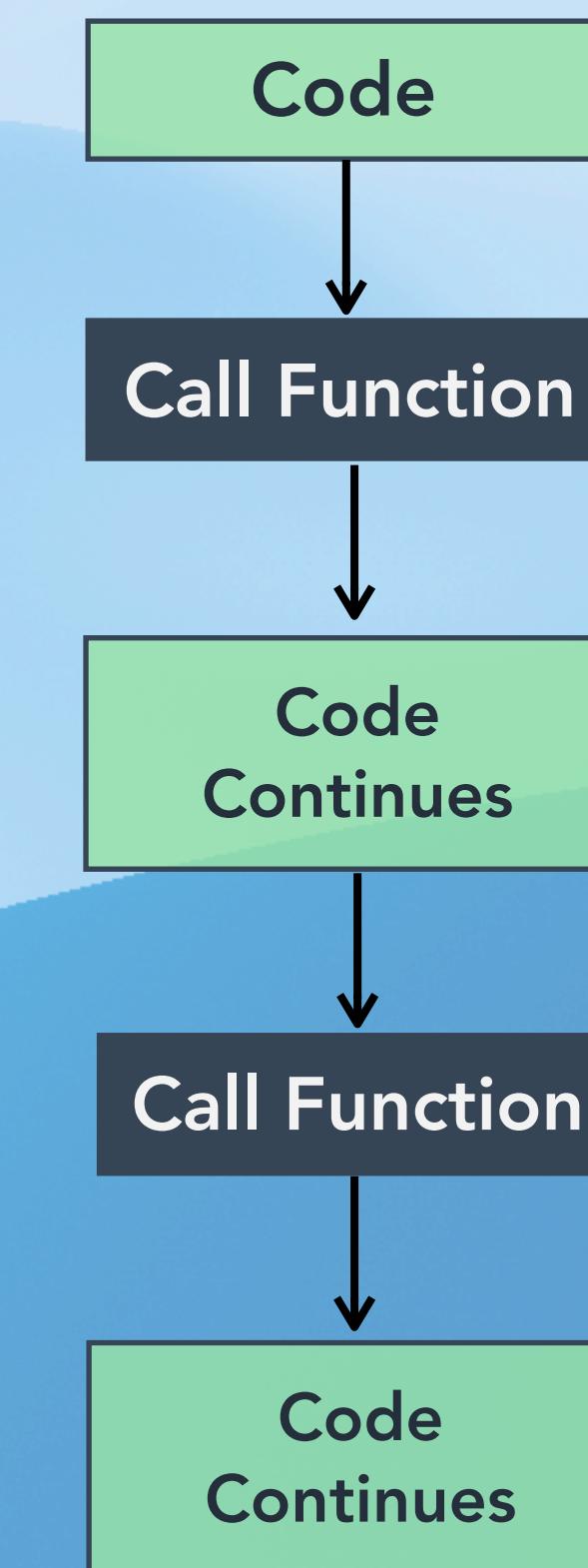
## Conditionals



## Loop



## Functions



## **Exercise 4A**

### **Conditions and For-loops**

# **Part 4B**

# **Functions**



# Functions

## Pros:

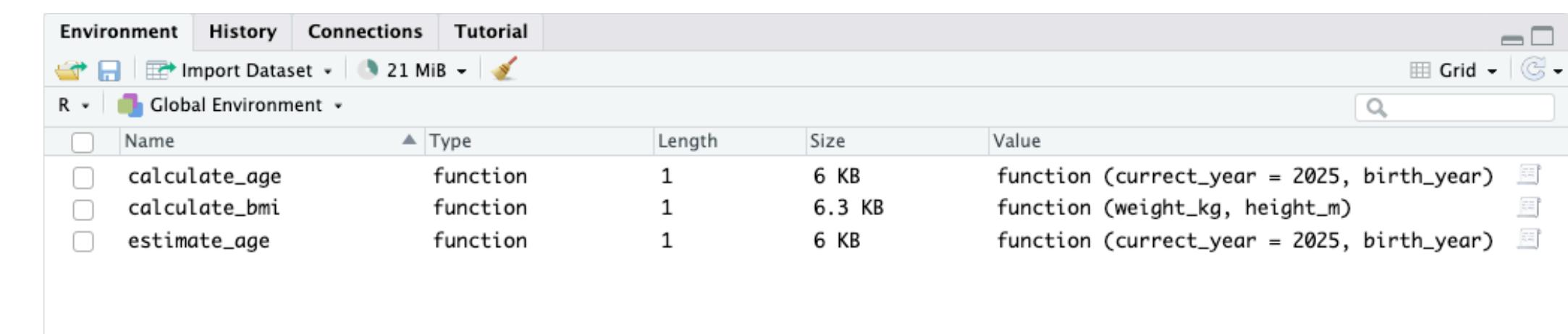
- Reusable: Write once, use many times.
- Readable: Code is clean and easy to understand.
- Modular: Breaks down complex tasks into steps.

## Code Example:

```
MyFunction <- function(x , y) {  
  z <- x + y  
  return(z)  
}
```

## Globale variables

- Defined outside functions.
- Accessible anywhere in the code.



Name	Type	Length	Size	Value
calculate_age	function	1	6 KB	function (current_year = 2025, birth_year) {
calculate_bmi	function	1	6.3 KB	function (weight_kg, height_m) {
estimate_age	function	1	6 KB	function (current_year = 2025, birth_year) {

## Local variables

- Defined inside functions.
- Only accessible within a specific function.

Globale variable

Local variable

# Functions

## Function

```
FUN_1 <- function(var1, var2){  
  result <- OPERATION_WITH_VARIABLES  
  return(result)  
}
```

Function with default variable values that can still be defined when the function is called.

```
FUN_2 <- function(var1 = "Hello", var2 = 5){  
  result <- OPERATION_WITH_VARIABLES  
  return(result)  
}
```

## Source script

```
source(functions.R)
```

## Call function (spits out the result)

```
FUN_1(var1 = 2, var2 = 9)
```

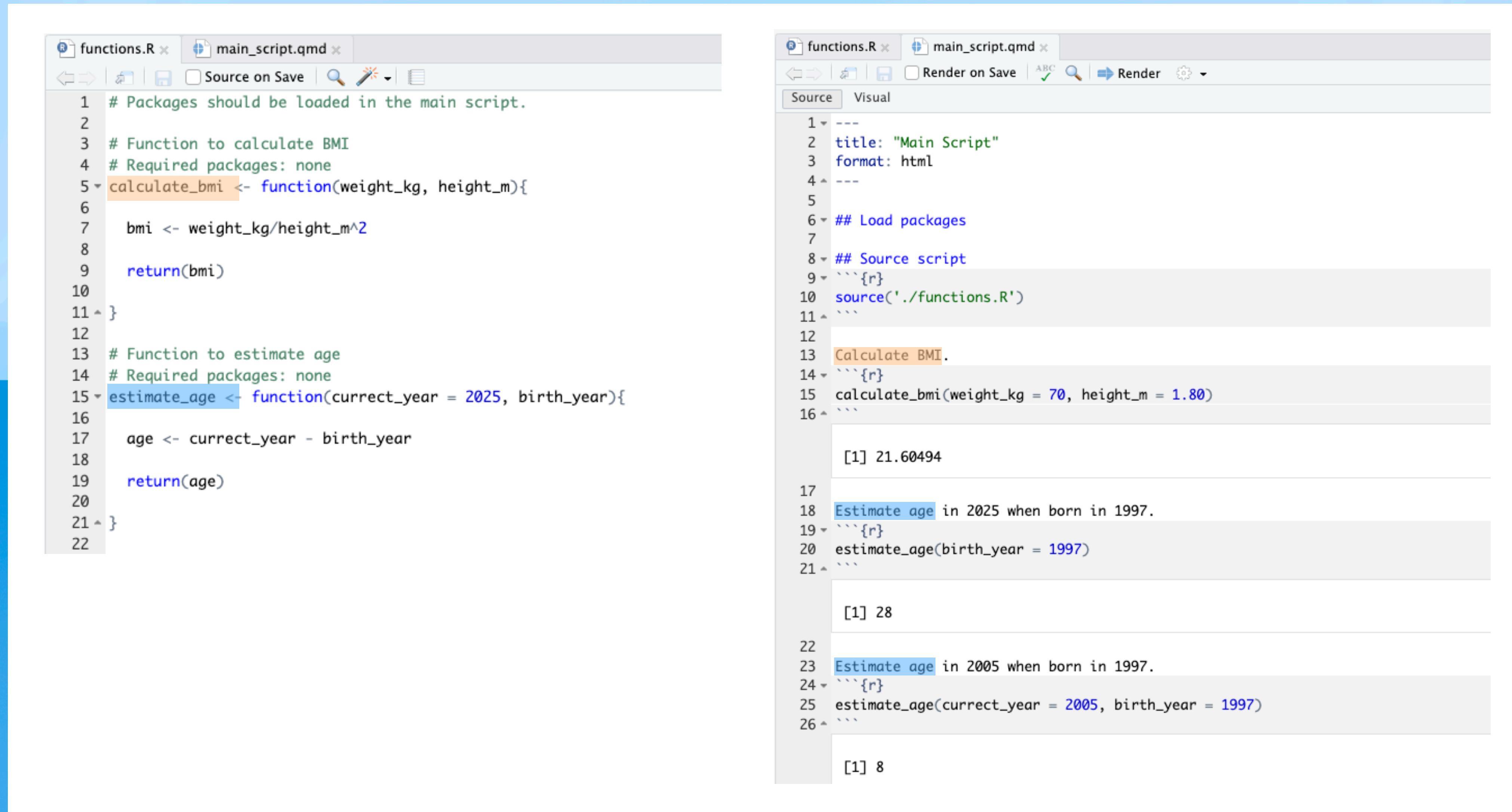
```
FUN_2(var1 = "World", var2 = 3)
```

## Save result of function call

```
RESULT_1 <- FUN_1(var1 = 2, var2 = 9)
```

```
RESULT_2 <- FUN_2(var1 = "World", var2 = 3)
```

# Functions



The image shows a screenshot of the RStudio interface. On the left, the code editor pane displays `functions.R` with the following content:

```
1 # Packages should be loaded in the main script.
2
3 # Function to calculate BMI
4 # Required packages: none
5 calculate_bmi <- function(weight_kg, height_m){
6
7   bmi <- weight_kg/height_m^2
8
9   return(bmi)
10
11 }
12
13 # Function to estimate age
14 # Required packages: none
15 estimate_age <- function(correct_year = 2025, birth_year){
16
17   age <- correct_year - birth_year
18
19   return(age)
20
21 }
22
```

On the right, the preview pane displays `main_script.qmd` with the following content:

```
1 ---  
2 title: "Main Script"  
3 format: html  
4 ---  
5  
6 ## Load packages  
7  
8 ## Source script  
9 ``{r}  
10 source('./functions.R')  
11 ``  
12  
13 Calculate BMI.  
14 ``{r}  
15 calculate_bmi(weight_kg = 70, height_m = 1.80)  
16 ``  
17  
[1] 21.60494  
18 Estimate age in 2025 when born in 1997.  
19 ``{r}  
20 estimate_age(birth_year = 1997)  
21 ``  
22  
[1] 28  
23 Estimate age in 2005 when born in 1997.  
24 ``{r}  
25 estimate_age(correct_year = 2005, birth_year = 1997)  
26 ``  
[1] 8
```

# Functions and Error Handling

The image shows two side-by-side RStudio sessions demonstrating error handling in a BMI calculation function.

**Left Session (functions.R):**

```
24 # Function to calculate BMI, with control points and error checking.
25 # Required packages: none
26 calculate_bmi_2 <- function(weight_kg, height_m) {
27   # Check if weight and height are numeric
28   if (!is.numeric(weight_kg) | !is.numeric(height_m)) {
29     stop("Both weight_kg and height_m must be numeric values.")
30   }
31
32   # Check if weight and height are positive
33   if (weight_kg <= 0) {
34     stop("Weight must be a positive value.")
35   }
36   if (height_m <= 0) {
37     stop("Height must be a positive value.")
38   }
39
40   # Calculate BMI
41   bmi <- weight_kg / height_m^2
42
43   # Check if BMI is within a reasonable range
44   if (bmi < 10 | bmi > 60) {
45     warning("The calculated BMI is outside the normal range. Please check your input values.")
46   }
47
48   return(bmi)
49
50 }
```

**Right Session (main\_script.qmd):**

```
127 The BMI function with out error handling returns a meaningless BMI value if given a negative weight.
128 ``{r}
129 calculate_bmi(weight_kg = -50, height_m = 1.80)
130 ```

[1] -15.4321

131
132 The BMI function with error handling returns an error if given a negative weight.
133 ``{r}
134 calculate_bmi_2(weight_kg = -50, height_m = 1.80)
135 ```

Error in calculate_bmi_2(weight_kg = -50, height_m = 1.8) :
  Weight must be a positive value.

136
137 The BMI function with error handling returns an warning if a BMI outside the normal range is calculated.
138 ``{r}
139 calculate_bmi_2(weight_kg = 25, height_m = 1.80)
140 ```

Warning in calculate_bmi_2(weight_kg = 25, height_m = 1.8) :
  The calculated BMI is outside the normal range. Please check your input values.
[1] 7.716049

141
142
```

## **Exercise 4B**

### **Functions**

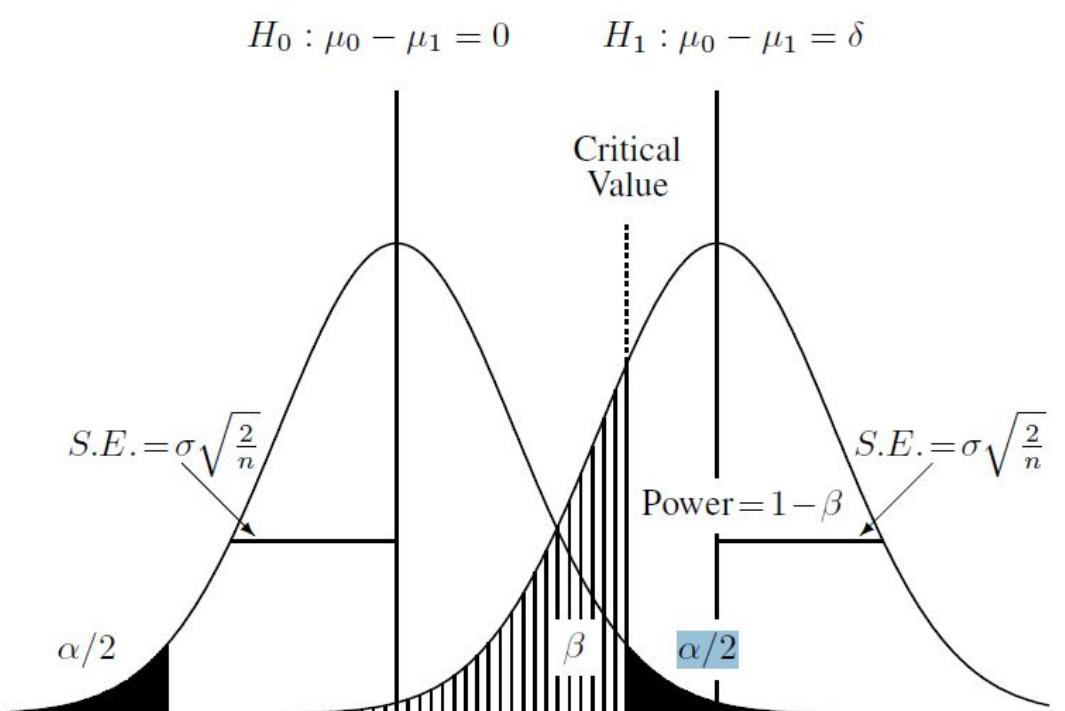
# **Part 5A**

# **Models in R**



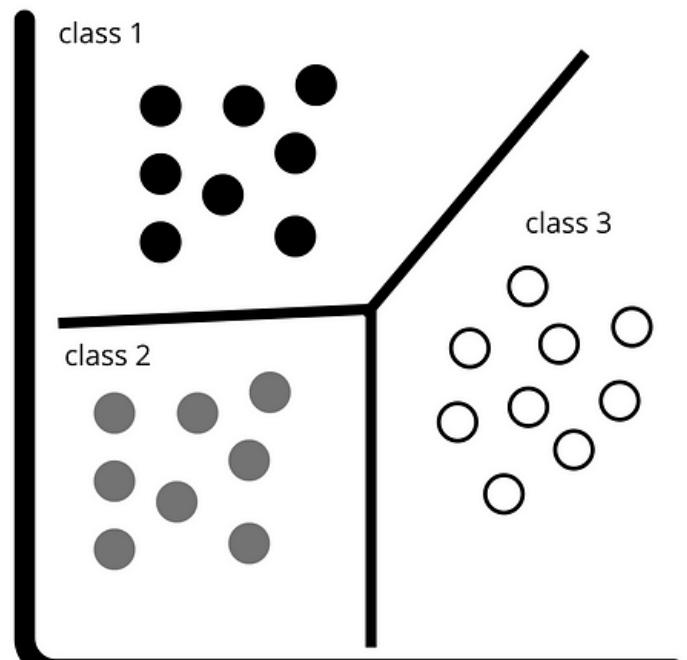
# Types of models

## Hypothesis Testing



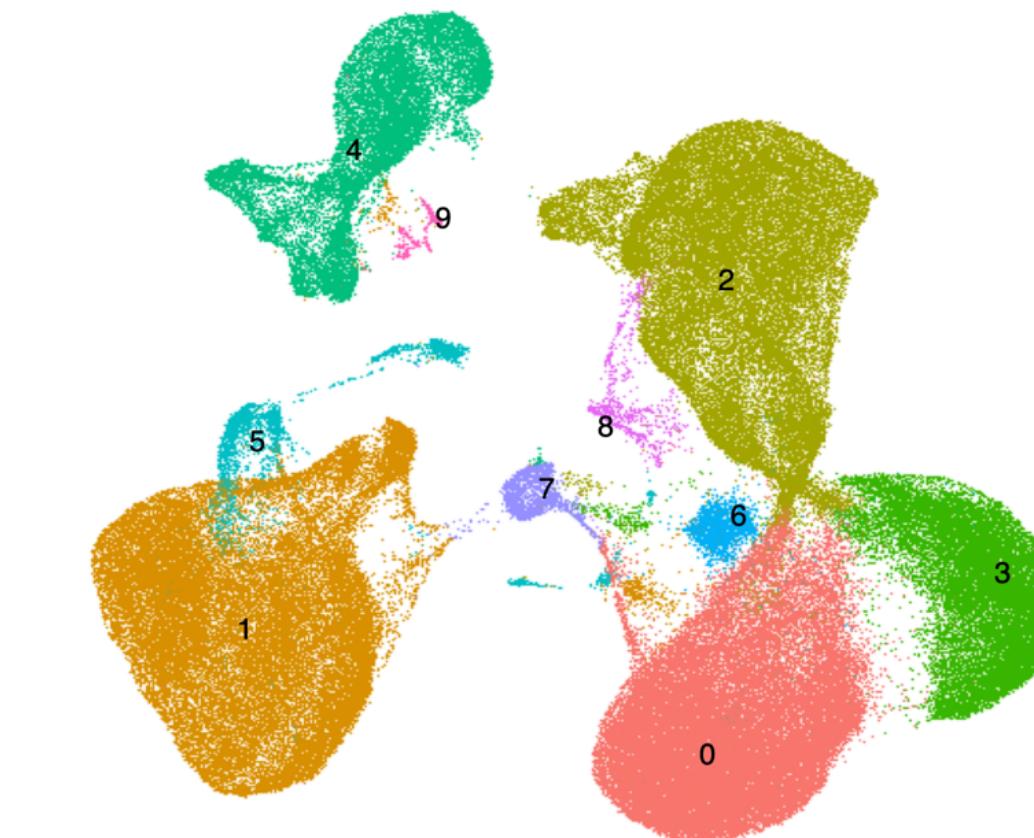
**RQ:** Is there a significant difference in the variable of interest between two or more groups?

## Prediction / Classification



- RQ:** Does this patient have diabetes?
- RQ:** Can we estimate cancer risk based on genetic risk, smoking and age?

## Unsupervised Learning



- RQ:** What sets the cancer cells apart from the healthy cells?
- Are there subtypes within one cell type?

# Hypothesis Testing

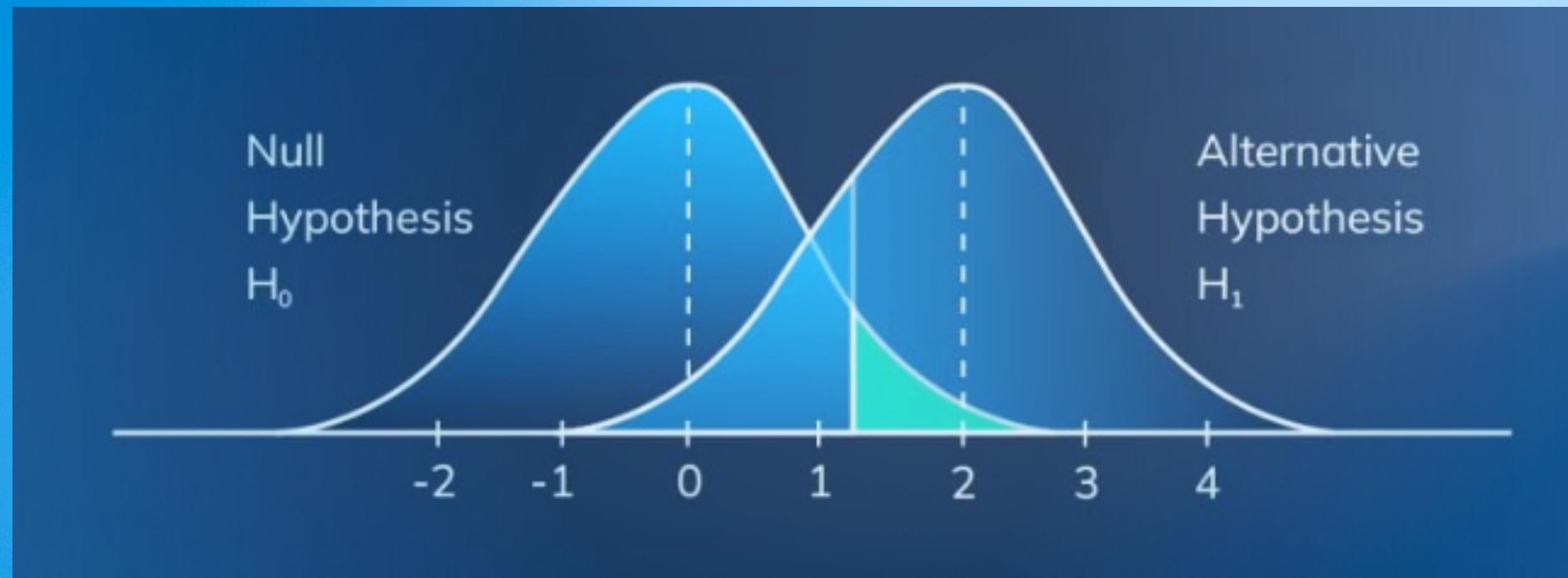
**Questions that boil down to:**

*Is there a difference in feature A between these two or more groups?*

- Is the bacterial load higher in colon swaps of cancer patients?
- Is the expression of gene A higher in tumor samples?
- Is there a difference in median height between men and women?

**Difference tests:**

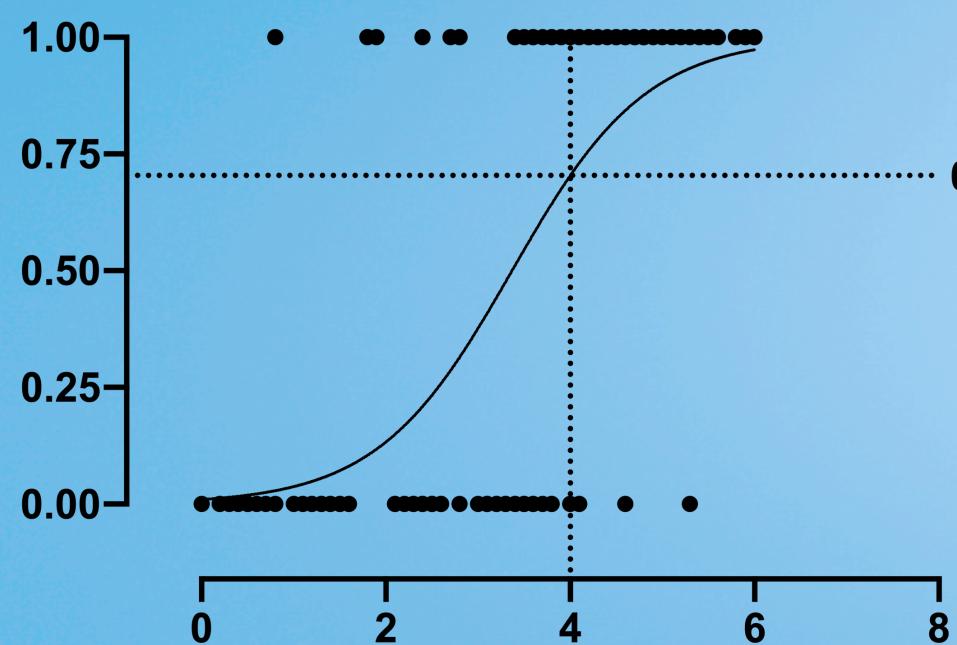
- Student's t-test
- F-test
- ANOVA / MANOVA
- Fisher's exact test



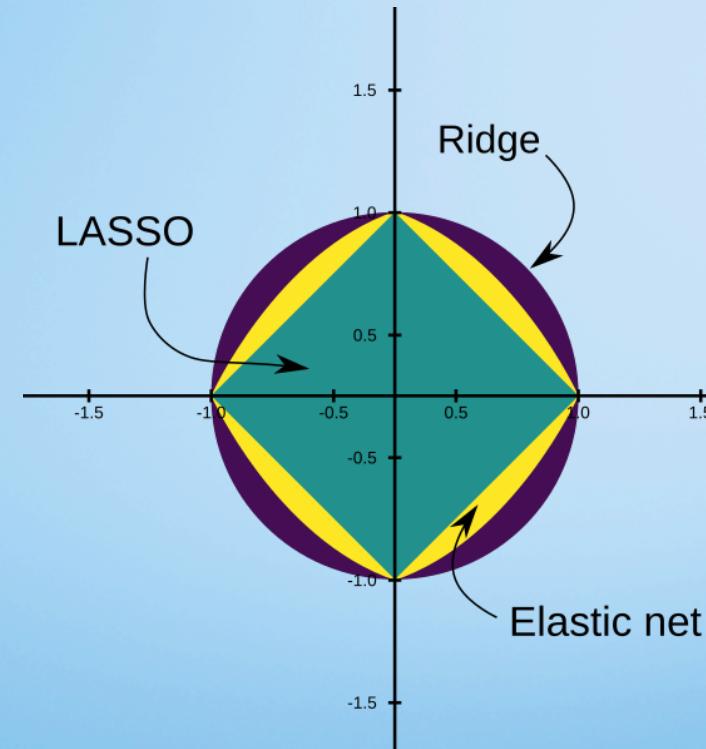
# Predictive Modelling & Classification

- Using statistical/ML models to **predict outcomes / classify new data.**
- Discovering and quantifying the relationships between predictor variables and outcome.
- **Prediction (Regression) ==** outcome is continuous (weight of newborn)
- **Classification ==** outcome is a class/group (cystic fibrosis or healthy)

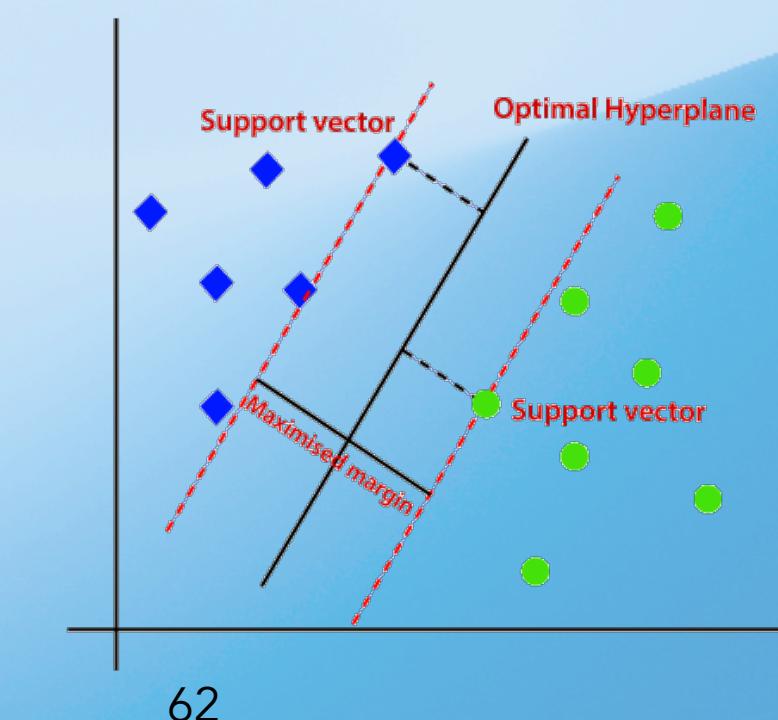
Logistic Regression



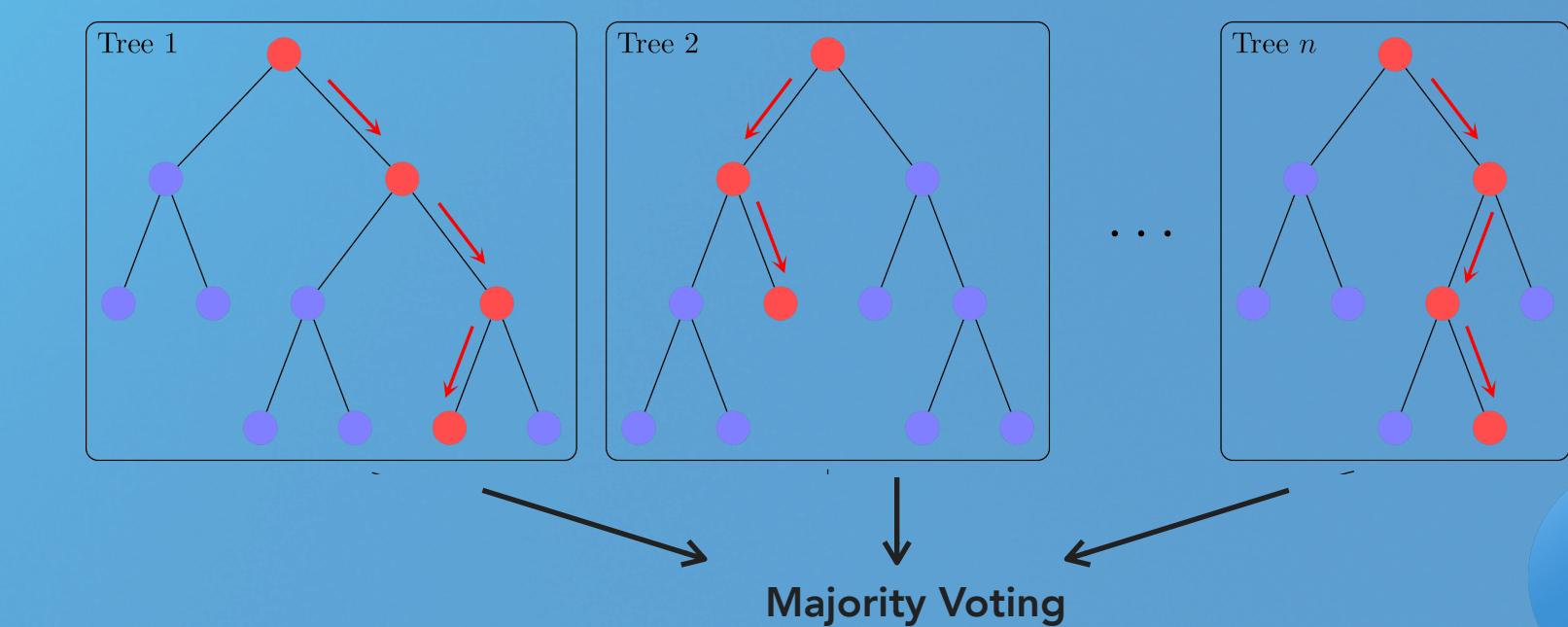
Penalized Regression



Support Vector Machine

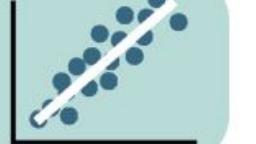
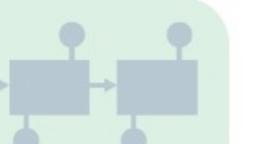


Random Forest



# Supervised Learning / Modeling

- **Supervised learning** is one category which encompasses many different models.
- **Supervised** means the ground truth is known.
- We have **labels** for the **outcome** when training our model, i.e. cancer or health sample

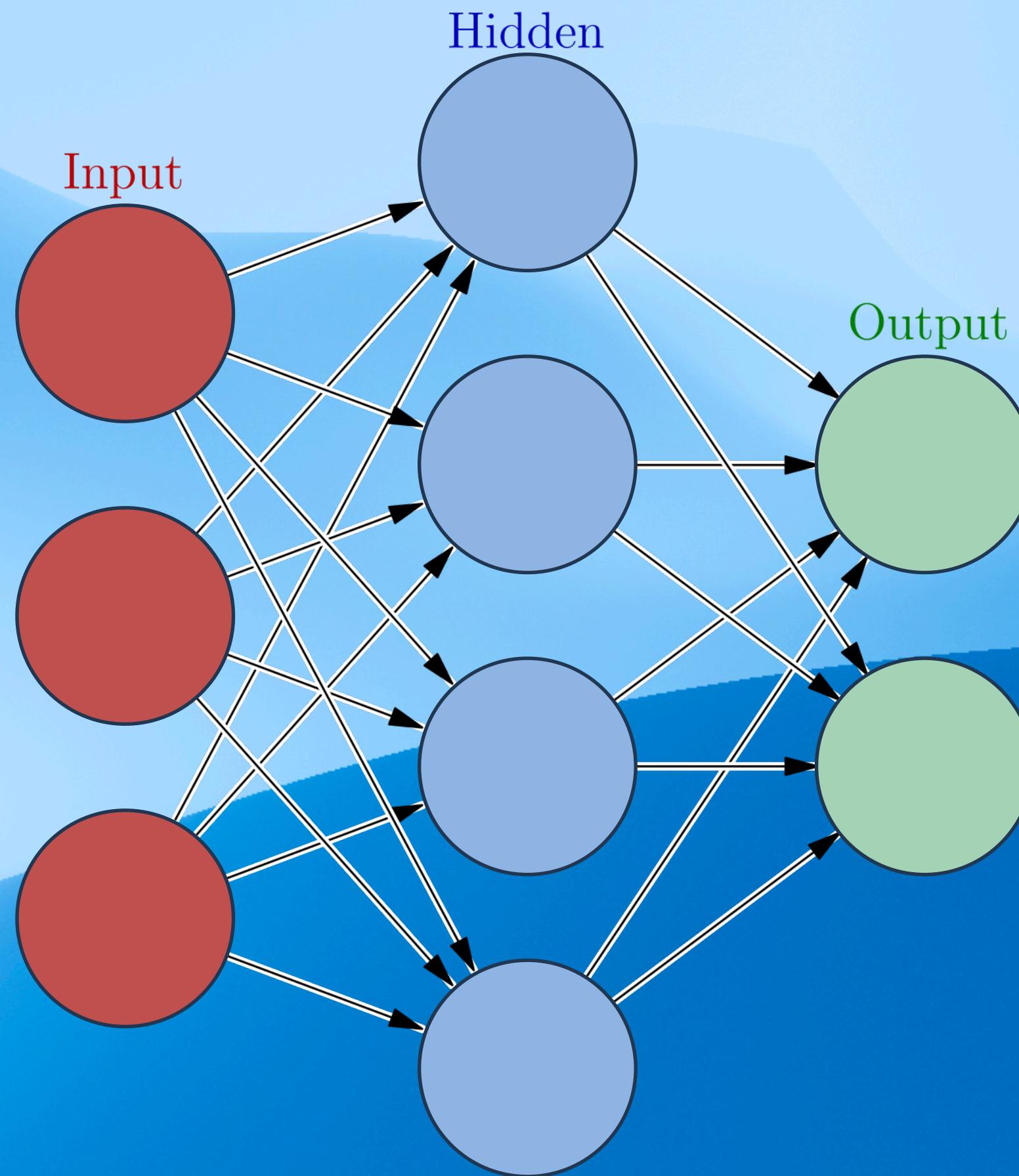
Frequently used algorithms for biomedical research	Example usage (data type)
SVM	
KNN	
Regression	
Random forest	
CNN	
RNN	

Supervised learning

Deep learning

# Deep Learning (Neural Networks)

- Neural Networks (NN) mimic the function and structure of the human brain
- NN can be either supervised or unsupervised
- We use NN for medical image analysis, genomics analysis, protein structure prediction, etc.



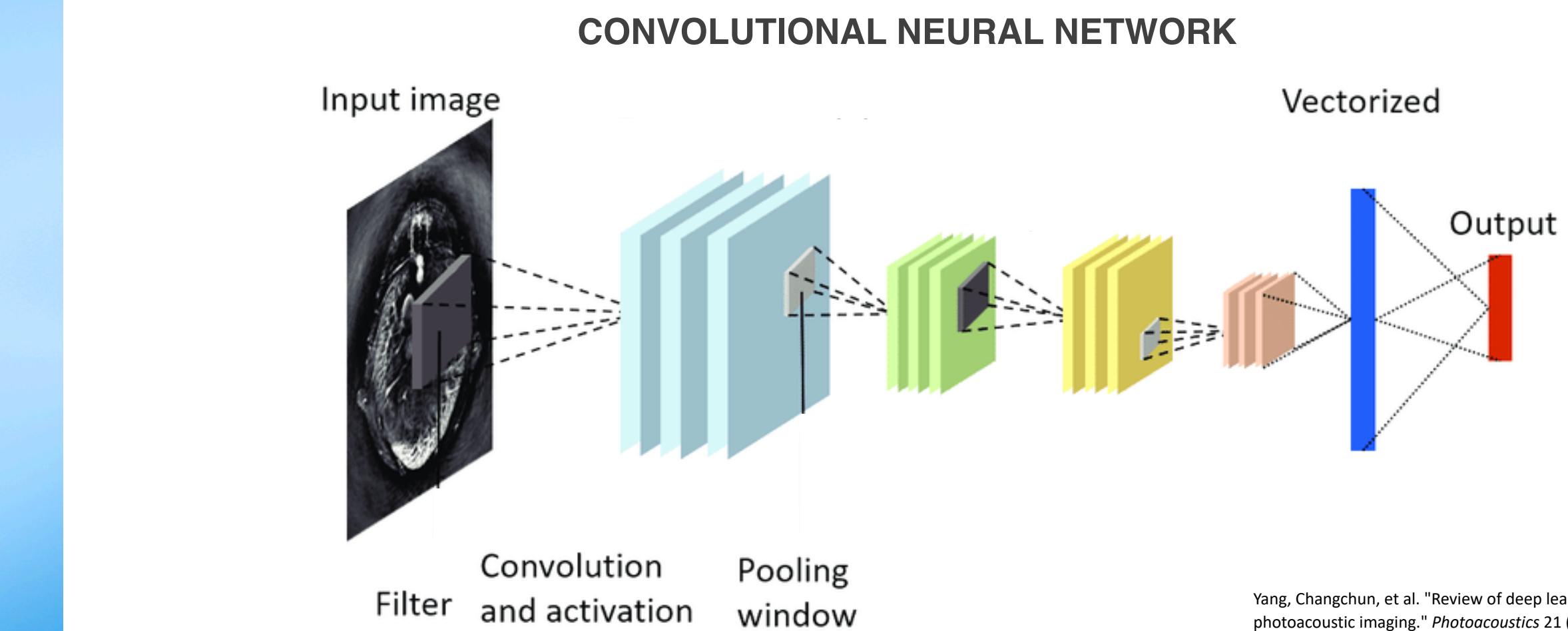
# Medical Image Analysis



- There are different architectures of neural networks (NNs).
- Different neural networks are good for different data and tasks.
- NNs for **medical image** analysis are often **convolutional neural networks (CNNs)**.
- **Transformer models** are very powerful:



- Many Data Scientists do NOT implement their own NNs, but they use a trained one.
- Can use different programming languages to implement NNs in.



Yang, Changchun, et al. "Review of deep learning for photoacoustic imaging." *Photoacoustics* 21 (2021): 100215.

# Neural Networks in R? Perhaps not...

## Deep Learning in R

**Deep Learning with H2O**

## Deep Learning in Python

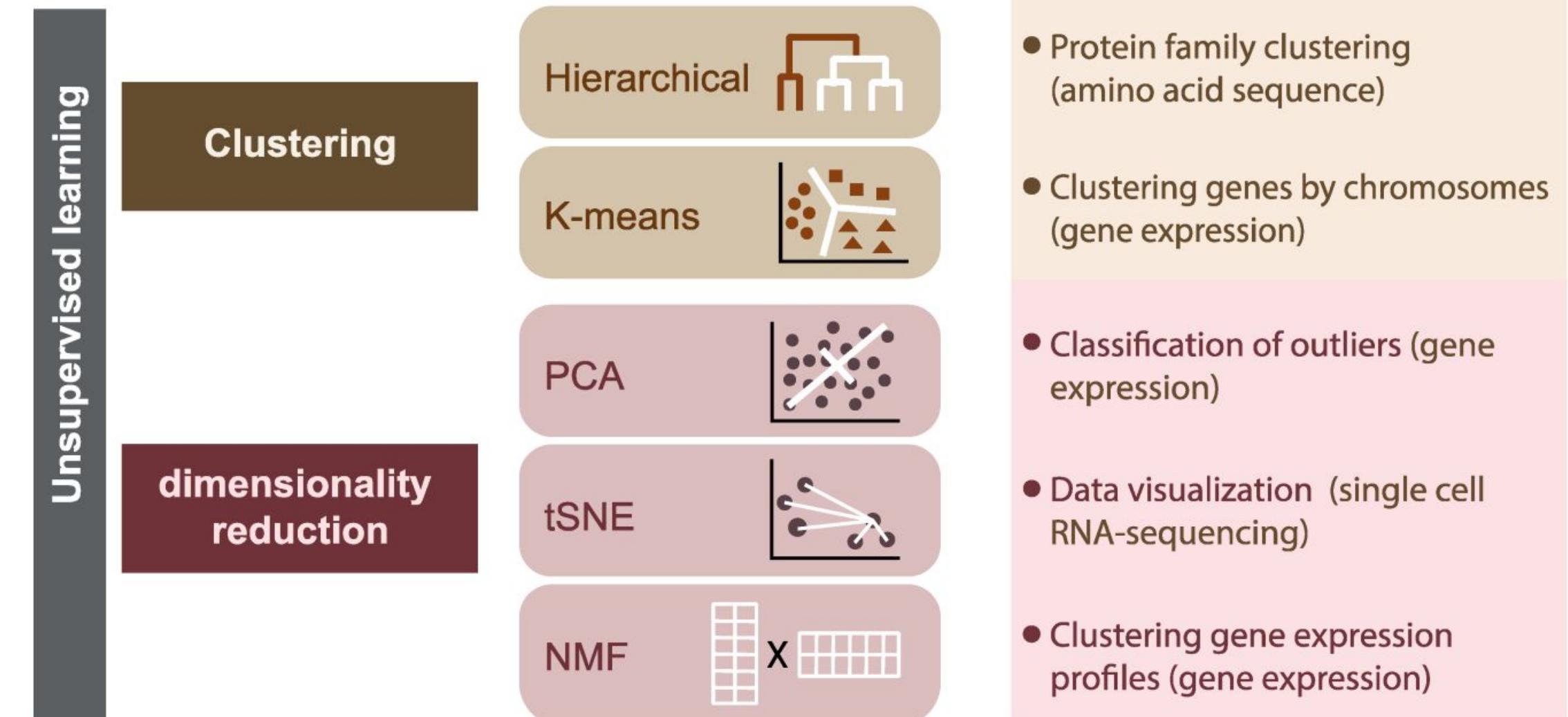


**torch for R** ECOYSTEM GET STARTED USE TORCH GET TECHNICAL RESOURCES DOCS



# Unsupervised Learning

- **Unsupervised Classification:** Group observations into clusters.
- What if we do not know the groups our data partition into, **no labels...**
- A scientific question could be: *Do our observations stratify into groups and what data characteristics drive this partitioning?*
- For this we use **unsupervised learning** methods (PCA is one example).



# CRAN & Bioconductor

- The primary package repository in the R community.
  - CRAN is a network of web servers that store identical, up-to-date, versions of code and documentation for R.
  - The repo is maintained by the designated “CRAN team”
- 
- Community-driven, open-source R packages for bioinformatics
  - RNA-seq, GWAS, epigenetic, proteomics, annotation, ect.
  - BiocManager is the install wrapper for Bioconductor:  
`biocmanager::install("package")`



# **Exercise 5A**

## **Regression Models**

# **Part 5B**

# **Model Evaluation in R**

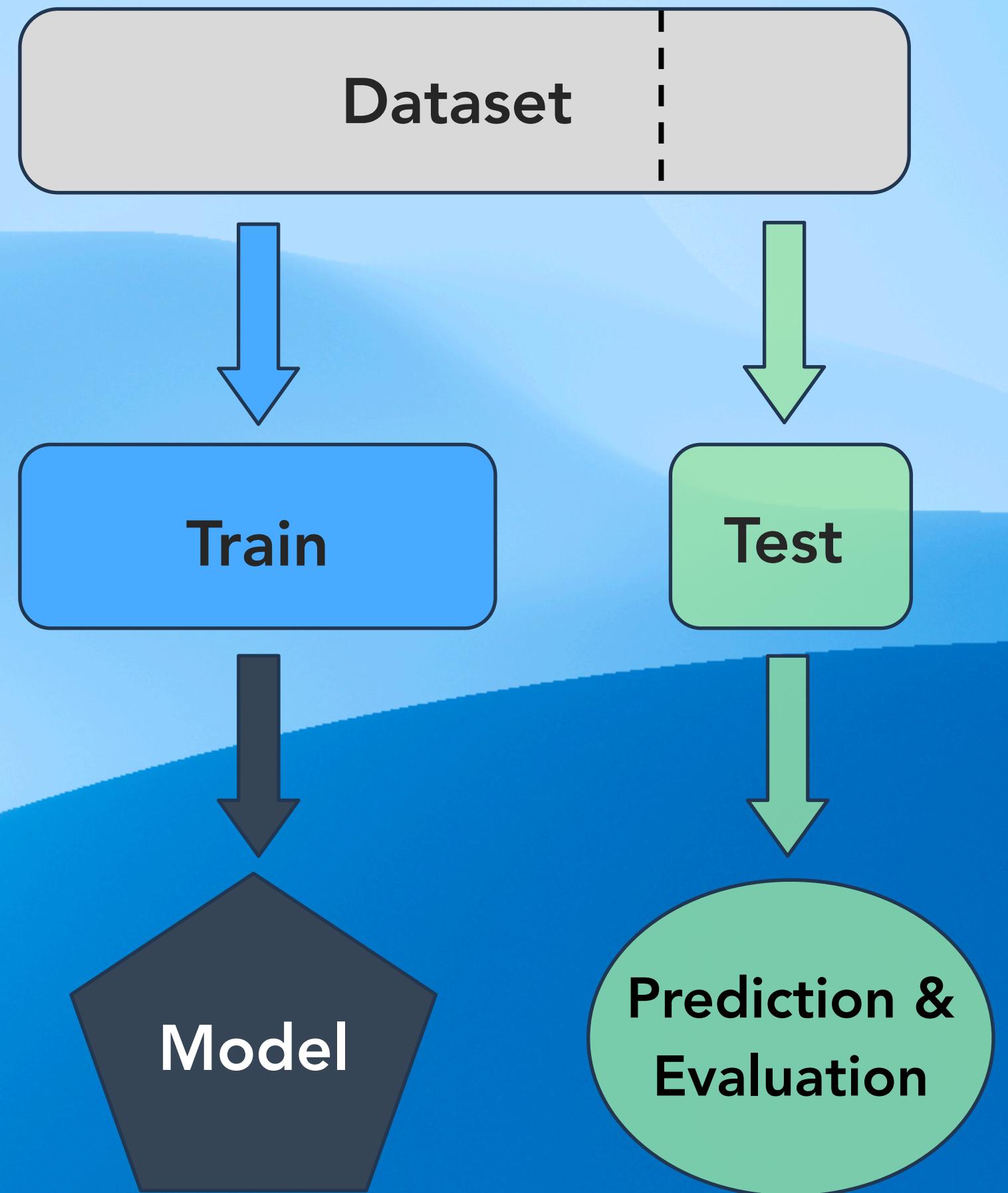


# Train & Test Dataset

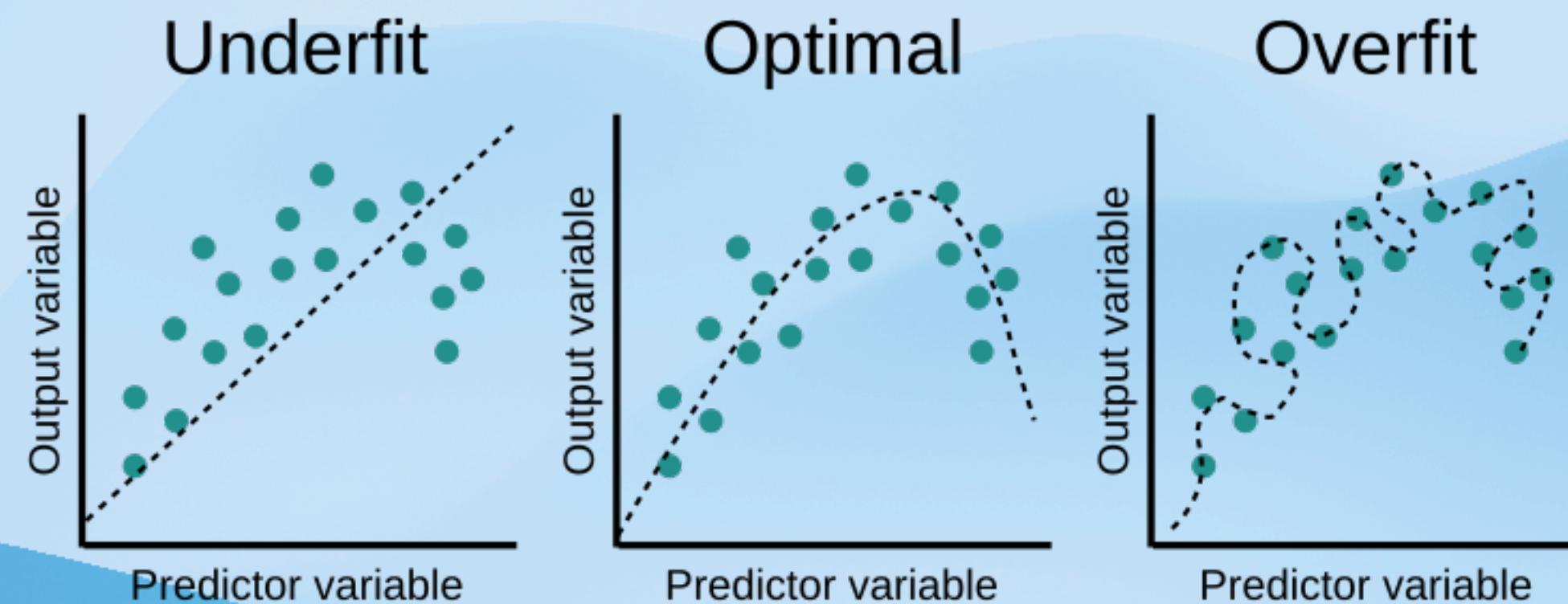
- We split our dataset into a **training set** and a **test set**
- **Training set** is used to train (estimate the parameters of) our model
- **Test set** is used to evaluate the utility of our model
- **N.B:** Sets must be balanced and contain all observed classes!

## R-syntax:

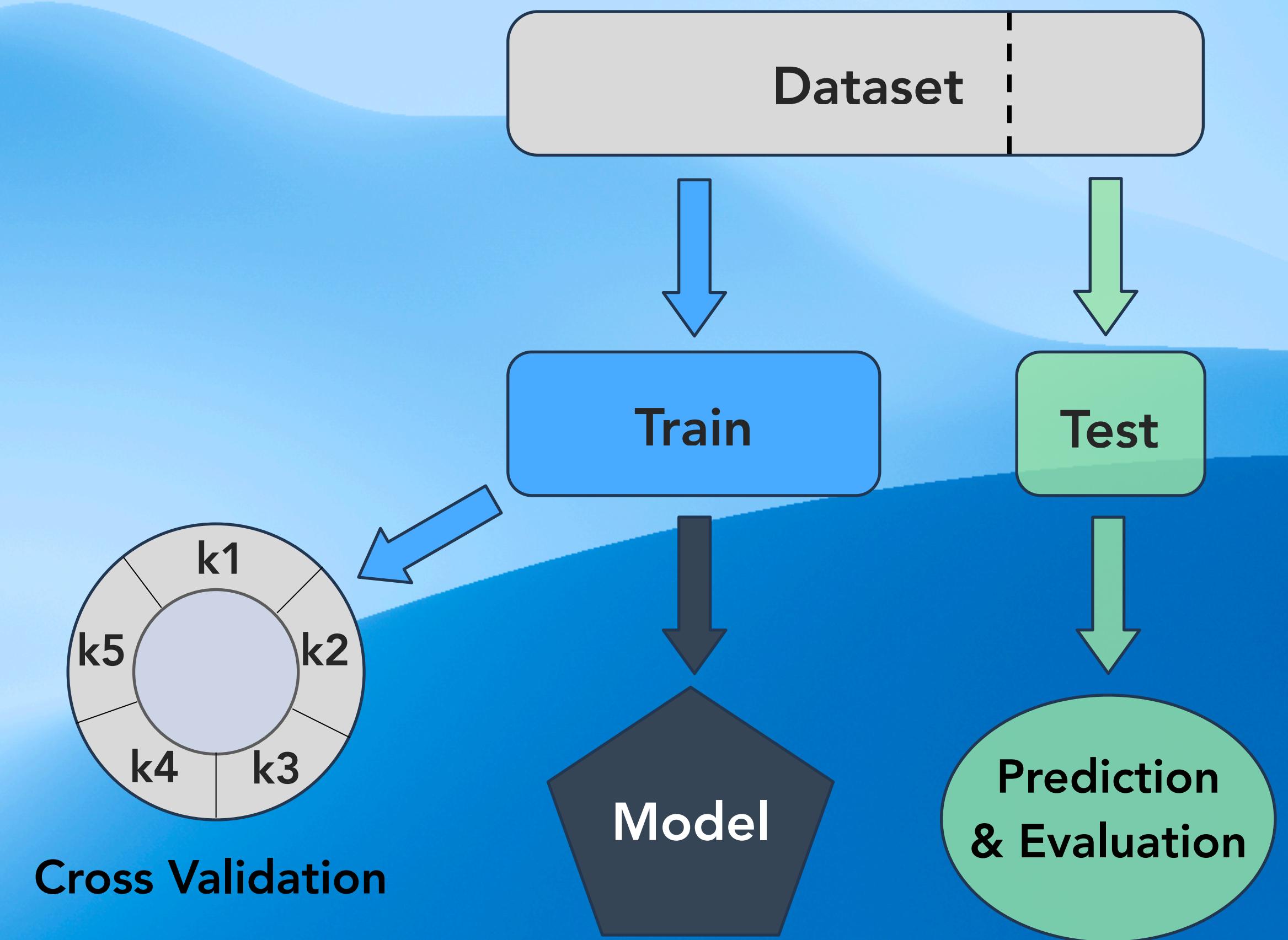
```
data <- read_csv("data/data.csv")  
  
library(caret)  
split <- createDataPartition(data$mpg, p = 0.8, list = FALSE)  
  
train_data <- data[split, ]  
test_data <- data[-split, ]
```



# Model Overfitting



- When a model follows the data too closely we get an effect known as overfitting.
  - Splitting the data into training and evaluation we partly avoid this
  - By performing **cross validation** while training we may avoid it further

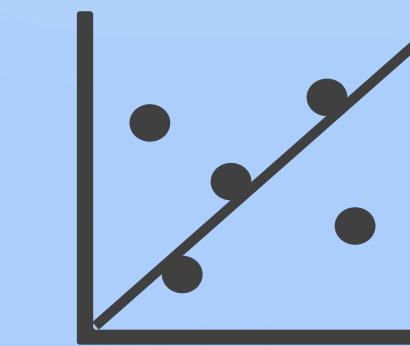


# Model Evaluation Metrics

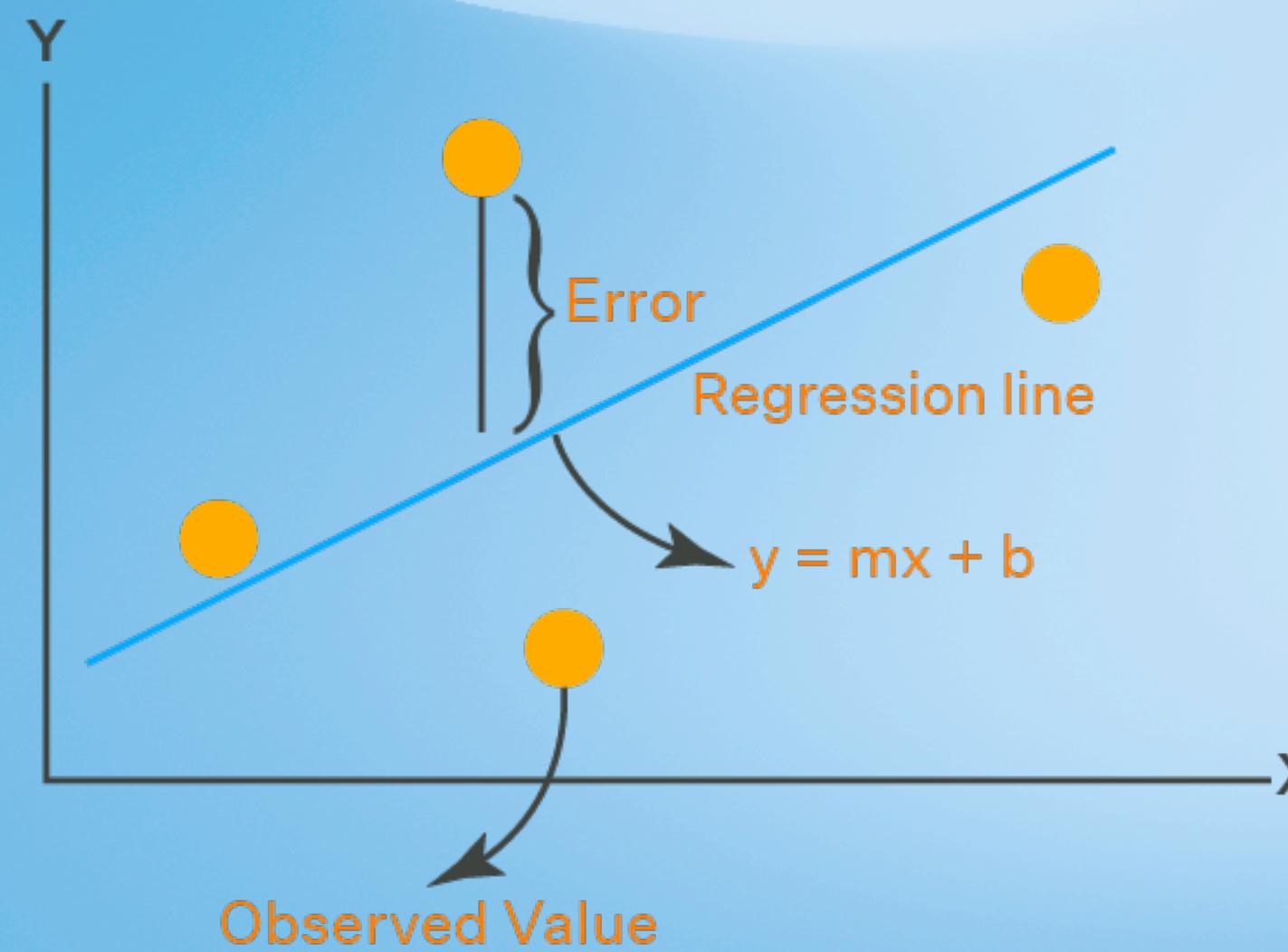
How well does the model fit the data?

- Criteria depend on model type
- Good models capture the underlying trends and characteristics

- R-squared
- MSE
- RMSE
- RMSLE

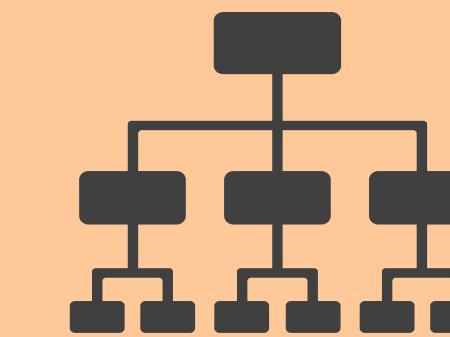


## REGRESSION



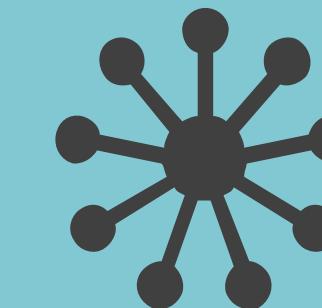
## CLASSIFICATION

- Accuracy
- Precision
- Recall
- ROC/AUC



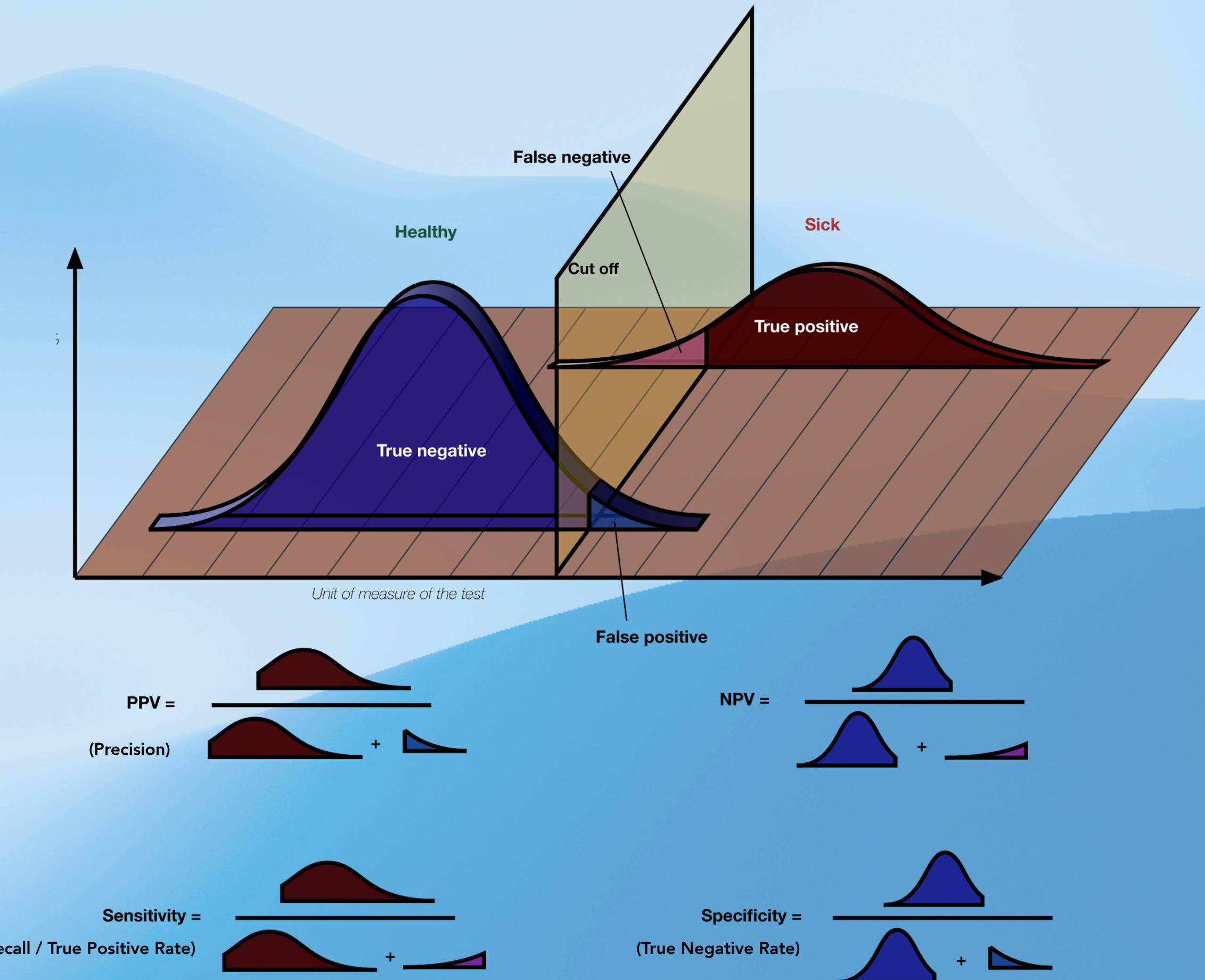
## CLUSTERING

- Silhouette
- Adj. Rand Indx.
- Gap statistic
- Davies-Bouldin



# Classifier Performance

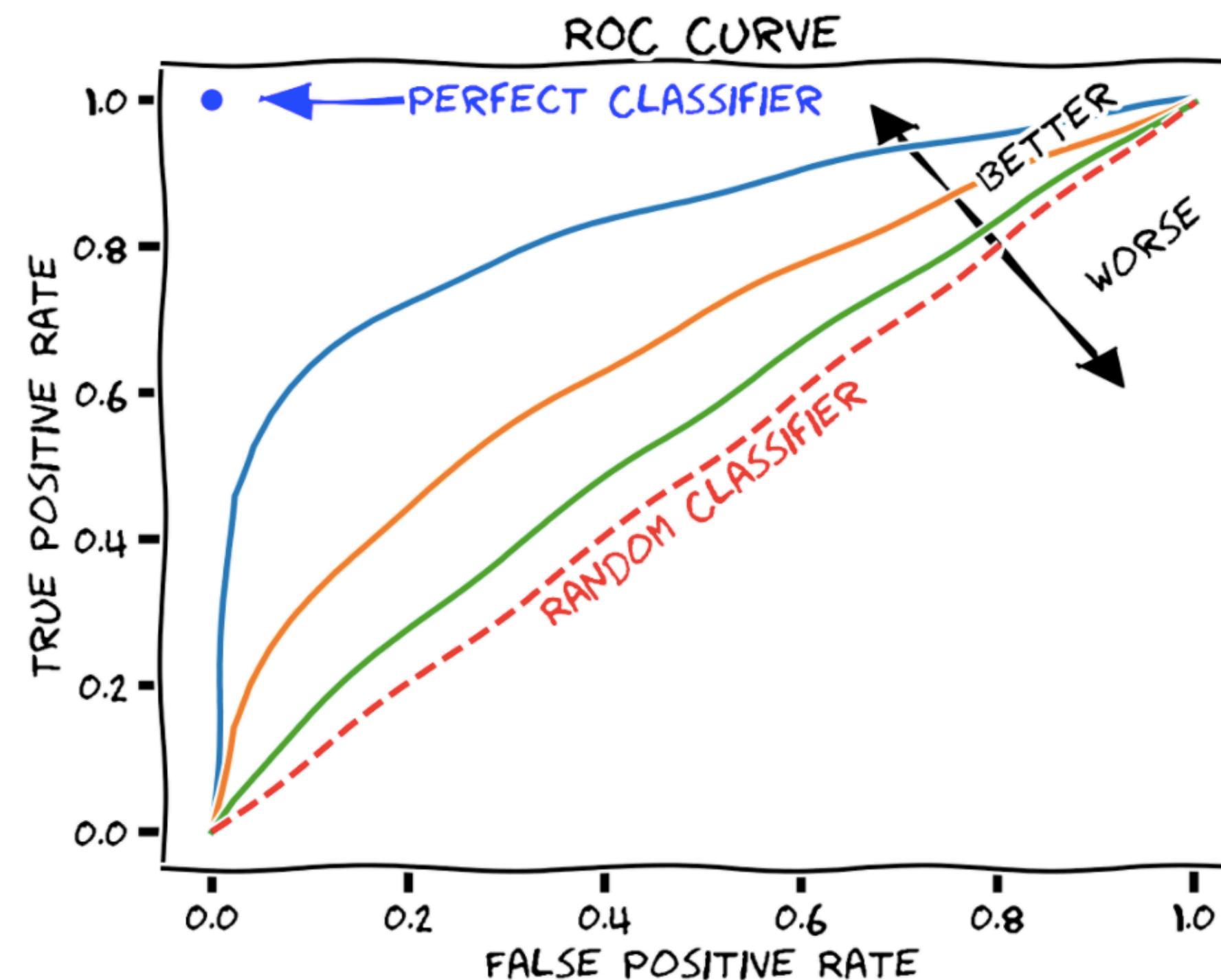
- **Recall** and **Precision** are often used as performance measures of classifiers.
- These metrics depend on the chosen cutoff.
- One can easily optimize on the number of false positives or false negatives, but not both at the same time.



# ROC Curve - Area Under the Curve (AUC)

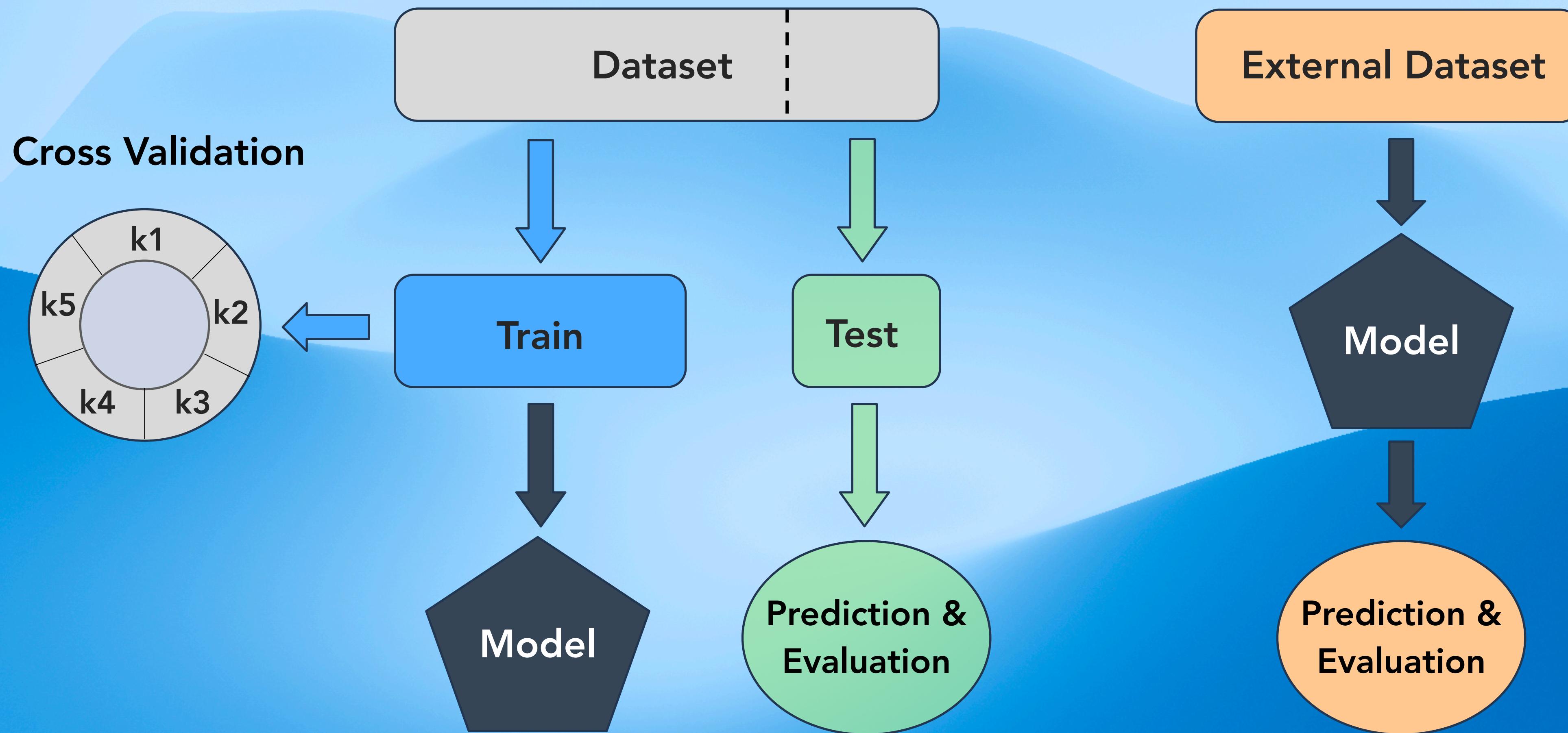
We can use the AUC to report on the goodness of the classification.

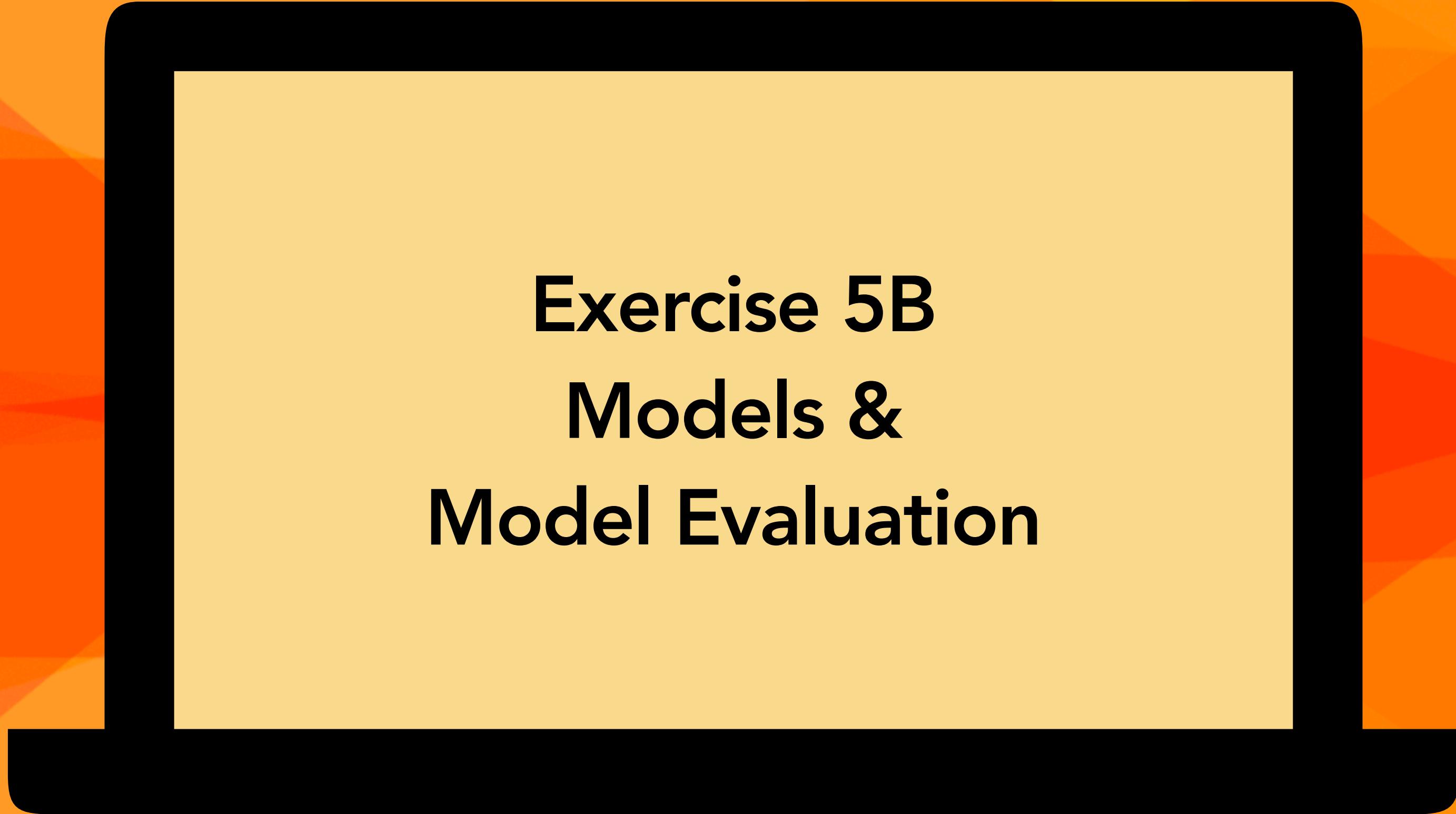
How many of the true cases have we discovered? →



- Receiver Operator Curve (ROC)
- Plotted by using every possible cutoff.
- Each True Positive Rate has an associated False Positive rate.

# Train-Test & Validation Dataset





## **Exercise 5B**

### **Models &**

### **Model Evaluation**

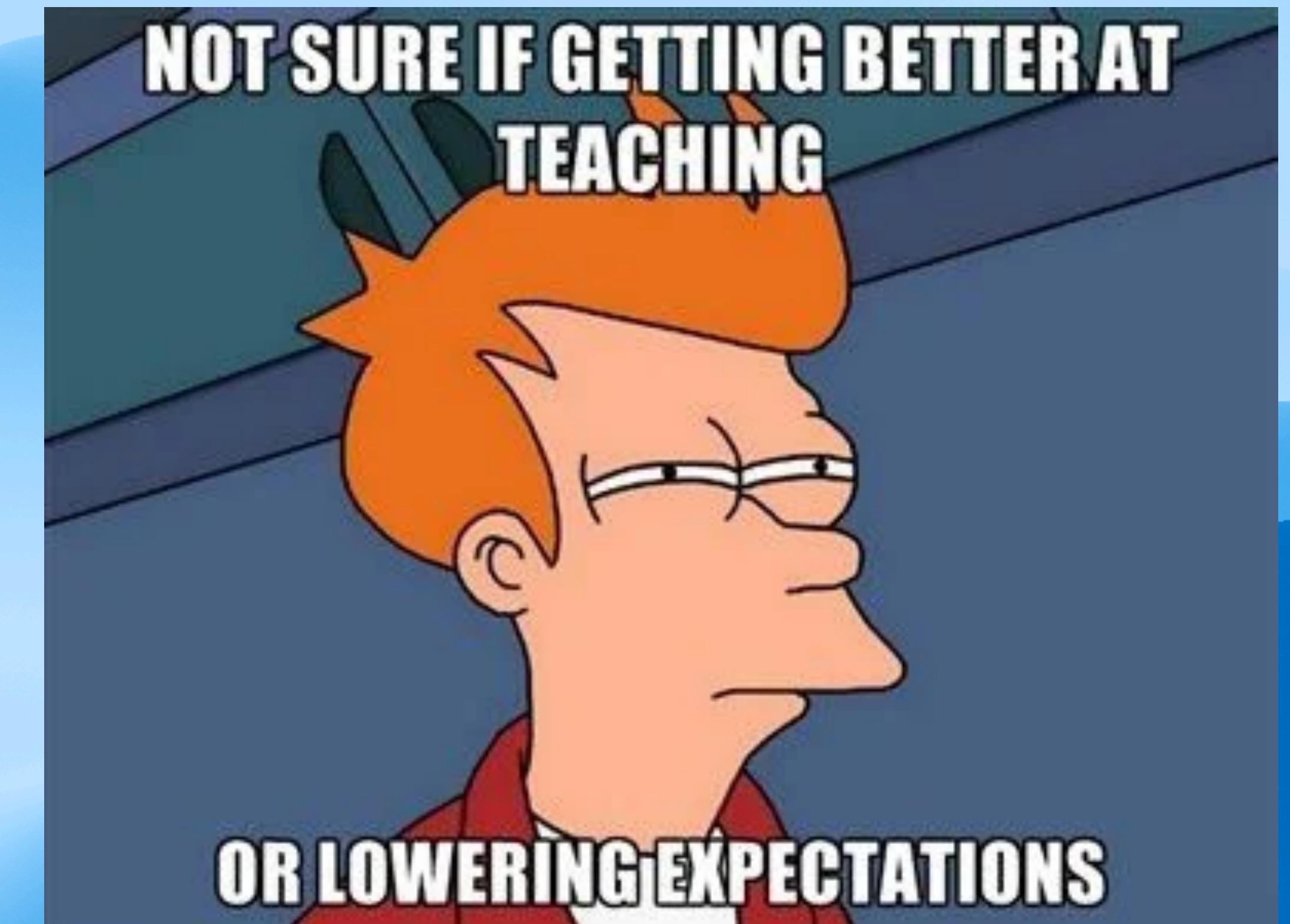
# Course Evaluation

- We would love your feedback on this course:

[https://forms.office.com/Pages/](https://forms.office.com/Pages/ResponsePage.aspx?id=kX-)

[ResponsePage.aspx?id=kX-](https://forms.office.com/Pages/ResponsePage.aspx?id=kX-)

[So6HNlkaviYyfHO\\_6kaKwKoywcVIMq5OaNsGb](https://forms.office.com/Pages/ResponsePage.aspx?id=kX-So6HNlkaviYyfHO_6kaKwKoywcVIMq5OaNsGb)  
[wftUNVA5NFBVSTE5V0hGUDFTQjIQQUxOME1](https://forms.office.com/Pages/ResponsePage.aspx?id=kX-wftUNVA5NFBVSTE5V0hGUDFTQjIQQUxOME1)  
[DNCQIQCN0PWcu](https://forms.office.com/Pages/ResponsePage.aspx?id=kX-DNCQIQCN0PWcu)



# Wrap-Up & Bring Your Own Dataset

- You now have time to:
  - Finish up exercises you may have missed!
  - Got questions about your own dataset?

Thank you for a nice course!

We end at 16.00 today -  
Stay and mingle with us and your  
other colleagues over a drink!

