R for Data Science

Center for Health Data Science

Who Are We?

Center for Health Data Science

- Data Science 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 ☕
- 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/

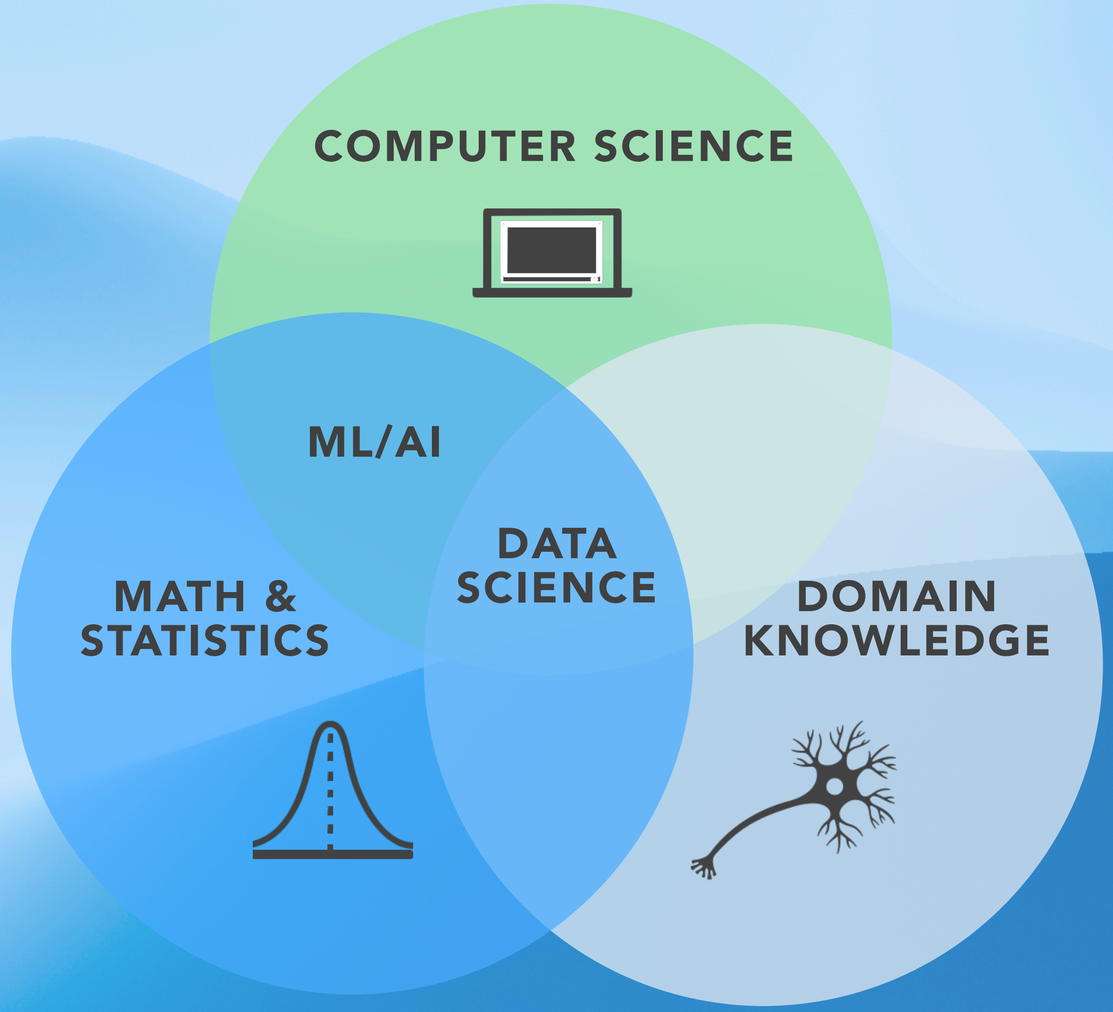
Program

Day 1		Day 2		Day 3	
08:30-09:00	Installation Issues + Coffee				
09:00-09:15	Introduction to Course	09:00-09:15	P3B: Exploratory Data Analysis	09:00–09:45	P5A Regression Models
09:15-09:30	P0: Intro to R and Quarto	09:15-10:15	E3B: Exploratory Data Analysis	09:45-10:30	E5A: Regression Models
09:30-10:00	E0: Intro to R and Quarto	10:15-10:30	Break	10:30-10:45	Break
10:00-10:30	P1: Cleanup & Summary Stats	10:30-11:15	P4A: Conditionals & Loops	10:45-11:15	E5A continued
10:30-10:45	Break	11:15-12:00	E4A: Conditionals & Loops	11:15-12:00	P5B: Other Models
10:45-12:00	E1: Cleanup & Summary Stats	12:00-13:00	Lunch	12:00-13:00	Lunch
11:15-12:00	P2: Data Wrangling	13:00-13:30	E4A continued	13:00-14:15	E5B: Other Models
12:00-13:00	Lunch	13:30-14:15	P4B: Functions	14:15-14:30	Course evaluation
13:00-14:15	E2: Data Wrangling	14:15-14:30	E4B: Functions	14:45-15:00	Break
14:15-14:30	Break	14:30-14:45	Break	13:00-15:30	Bring Your Own Dataset
14:30-15:00	P3A: Exploratory Data Analysis	14:45-16:00	E4B continued	16:00	Mingling Session
15:00-16:00	E3A: Exploratory Data Analysis	16:00	See you tomorrow!		
16:00	See you tomorrow!				

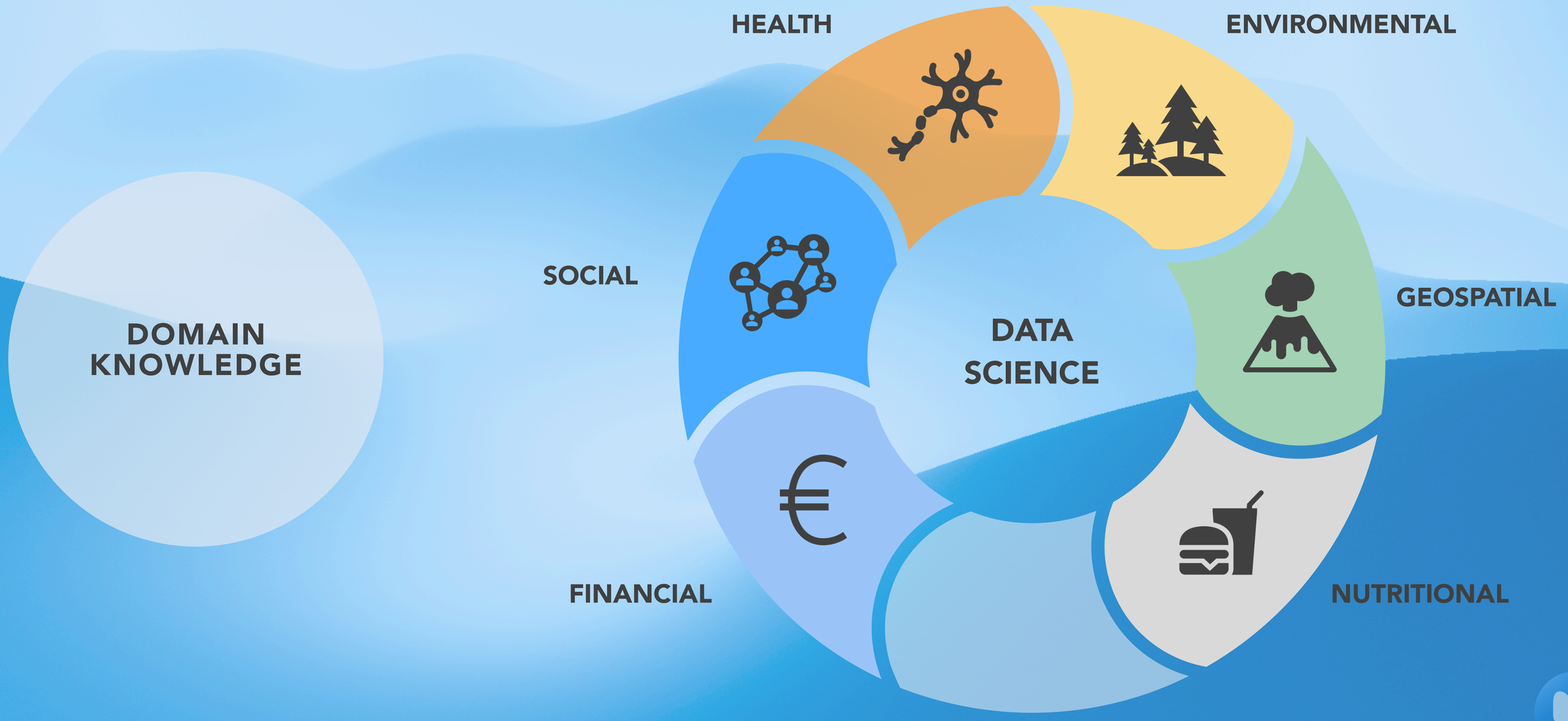
What is Data Science?

- A **cross-disciplinary** undertaking that draws on many disciplines and is in turn becoming part of many disciplines.

- Data matters, one size analysis does not fit all.



Flavors of Data Science?



Analysis of Health Data

Research /Clinical Studies

Registries & Questionnaires
Pharmacological Databases

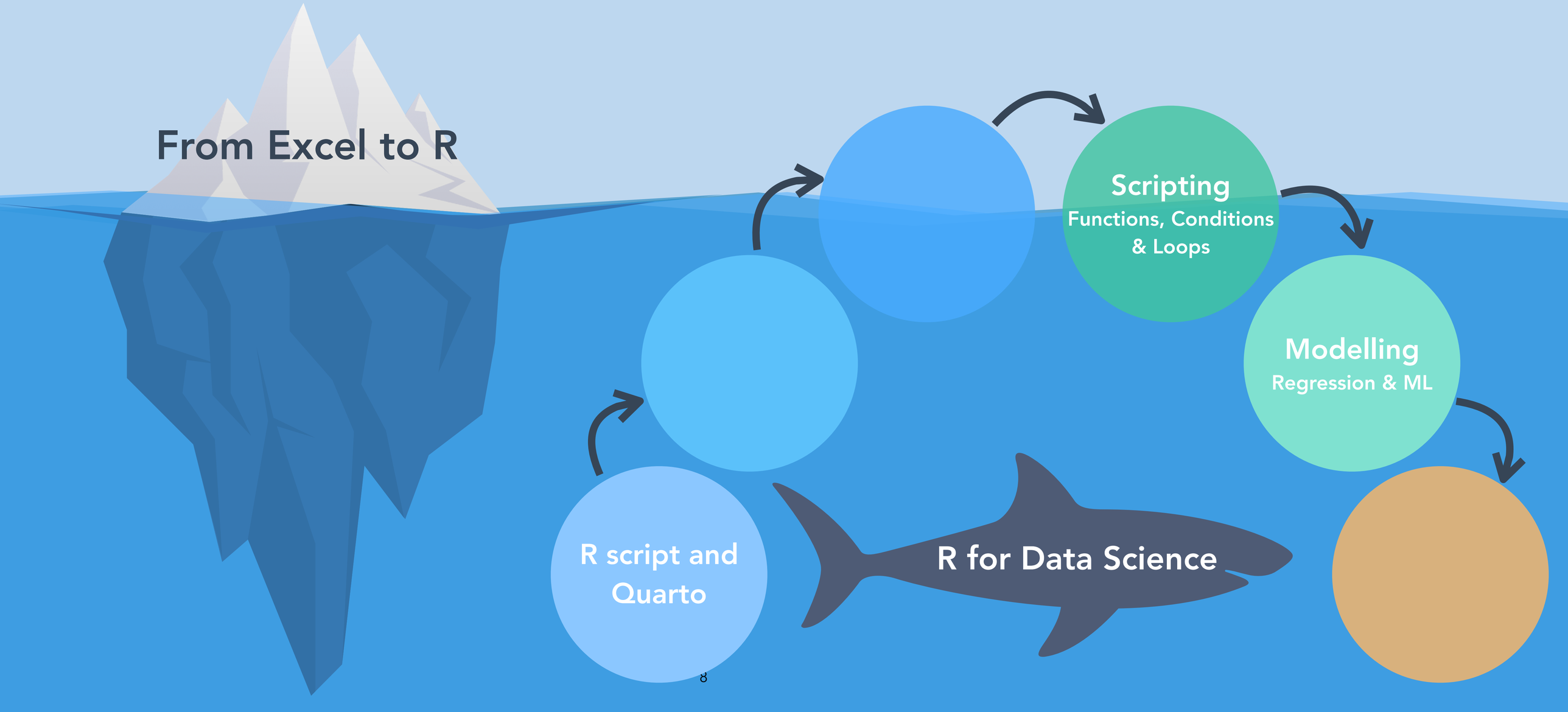
Biological (omics) Databases
Medical Image Data

Wearable Devices
Search Engine Data

'Future' Data

DATA SCIENCE
SKILLSET

What Will We Learn



Part 0
R script and Quarto

R script

- Flat script
  - Fast, less moving parts then 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

Quarto

Report

R_script_example.R

```r
##########################################
# R for Data Science - How to R script #
# Author: DataLab HeaDS                #
# Date: 8 Novmeber 2024                #
##########################################

##################################################
################# Load Packages ##################
##################################################
library(tidyverse)
library(readxl)

##################################################
################### Load Data ####################
##################################################
diabetes <- read_excel('~/Desktop/DataLab/R4DataScience/data/diabetes_toy.xlsx')

##################################################
################# Inspect Data ###################
##################################################

# Check dimensions of data
dim(diabetes)

# Check structure of data
str(diabetes)

# Check for NA's in each column
colSums(is.na(diabetes))

##################################################
########### Exploratory Data Analysis ############
##################################################

# Plot distribution of BMI
diabetes %>%
  ggplot(aes(x = BMI)) +
  geom_histogram(bins = 10)
```

Quarto_example.qmd

```yaml
---
title: "R for Data Science - How to R script"
format: html
author: DataLab HeaDS
editor: visual
---
```

Load Packages

```r
library(tidyverse)
library(readxl)
```

Load Data

```r
diabetes <- read_excel('~/Desktop/DataLab/R4DataScience/data/diabetes_toy.x
```

Inspect Data

Check dimensions of data

```r
dim(diabetes)
```

Check structure of data

```r
str(diabetes)
```

Check for NA's in each column

```r
colSums(is.na(diabetes))
```

Exploratory Data Analysis

Plot distribution of BMI

```r
diabetes %>%
  ggplot(aes(x = BMI)) +
  geom_histogram(bins = 10)
```

Render .html .pdf

R for Data Science - How to Quarto

AUTHOR
DataLab HeaDS

Load Packages

```r
library(tidyverse)
library(readxl)
```

Load Data

```r
diabetes <- read_excel('~/Desktop/DataLab/R4DataScience/data/diabet
```

Inspect Data

Check dimensions of data

```r
dim(diabetes)
```

```
[1] 100   8
```

Check structure of data

```r
str(diabetes)
```

```
[100 × 8] (S3: tbl_df/tbl/data.frame)
                        : chr [1:100] "ID_035" "ID_020" "ID_090" "ID_
g_Blood_Sugar           : num [1:100] 157 146 128 134 157 101 144 14
eal_Blood_Sugar         : num [1:100] 204 178 177 206 206 250 204 18
                        : num [1:100] 5.8 6.3 6.6 8 6.9 7.4 6.9 5.9
                        : num [1:100] 36 70 63 49 56 39 59 43 57 56
                        : chr [1:100] "Male" "Male" "Male" "Male" ...
                        : num [1:100] 33.7 26.5 26 22.9 25.7 24.4 25
_Pressure               : num [1:100] 122 120 137 126 115 121 116 11
```

Check for NA's in each column

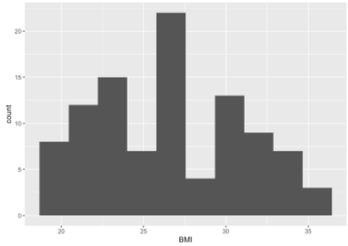
```r
colSums(is.na(diabetes))
```

```
                  ID   Fasting_Blood_Sugar Post_Meal_Blood_Sugar
                   0                     0                    0
               HbA1c                   Age                  Sex
                   0                     0                    0
                 BMI         Blood_Pressure
                   0                     0
```

Exploratory Data Analysis

Plot distribution of BMI

```r
diabetes %>%
  ggplot(aes(x = BMI)) +
  geom_histogram(bins = 10)
```



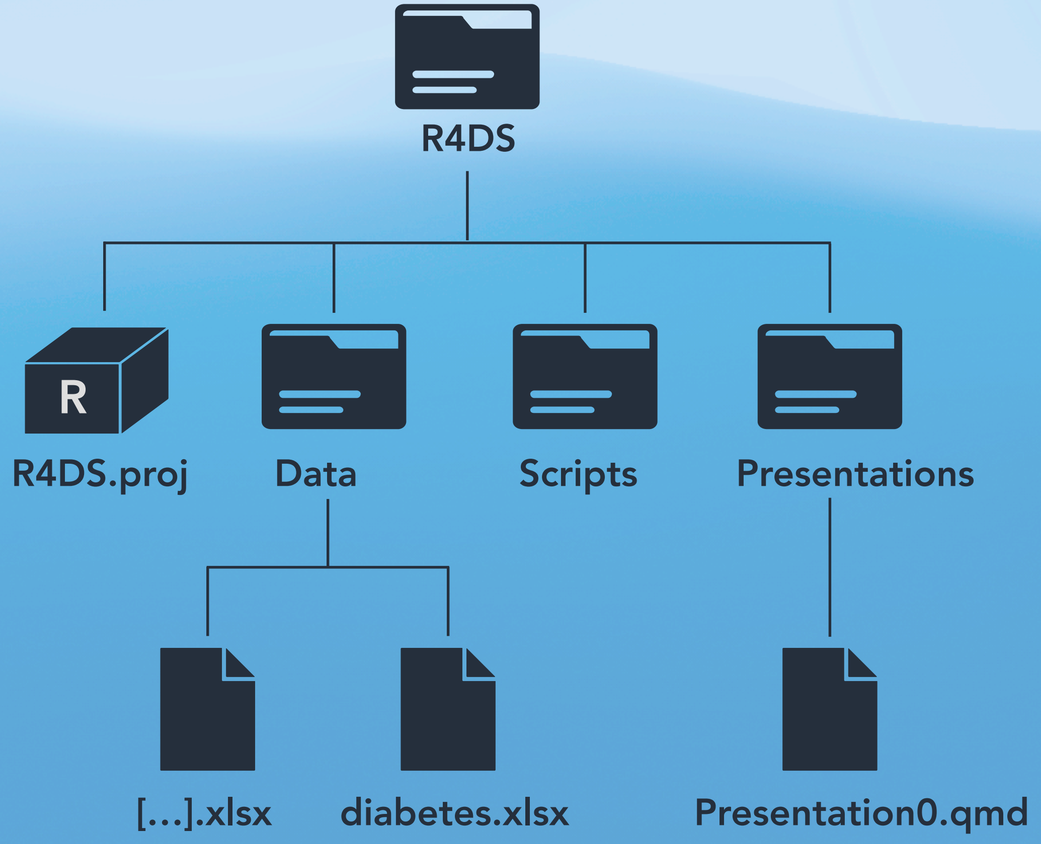
R project

R4DS.Rproj

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

.Rproj.user

- Hidden directory to store of temporary files.



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 %>%
- Legible & intuitive syntax
- Functions assume **tibble**
- Can get slow for very big datasets

Cheat Sheet - Base R and Tidyverse

	Base R	Tidyverse
Select columns	`df[, c("col1", "col2")]`	`df %>% select(col1, col2)`
Select one column	`df["col1"]`	`df %>% select(col1)`
Access one column as vector	`df$col1` `df[["col1"]]`	`df %>% pull(col1)`
Add new column	`df$col_new <- list_new`	`df <- df %>%` `mutate(col_new = list_new)`
Filter rows	`df[df$col1 < 10 ,]`	`df %>% filter(col1 < 10)`
Remove rows with NA's	`df[complete.cases(df_baseR), ]`	`df %>% drop_na()`
Access unique values of column	`unique(df$col1)`	
Access count of unique values of column	`table(df$col1)`	

Exercise 0
R script and Quarto

Part 1
Data Cleanup

Base R and Tidyverse

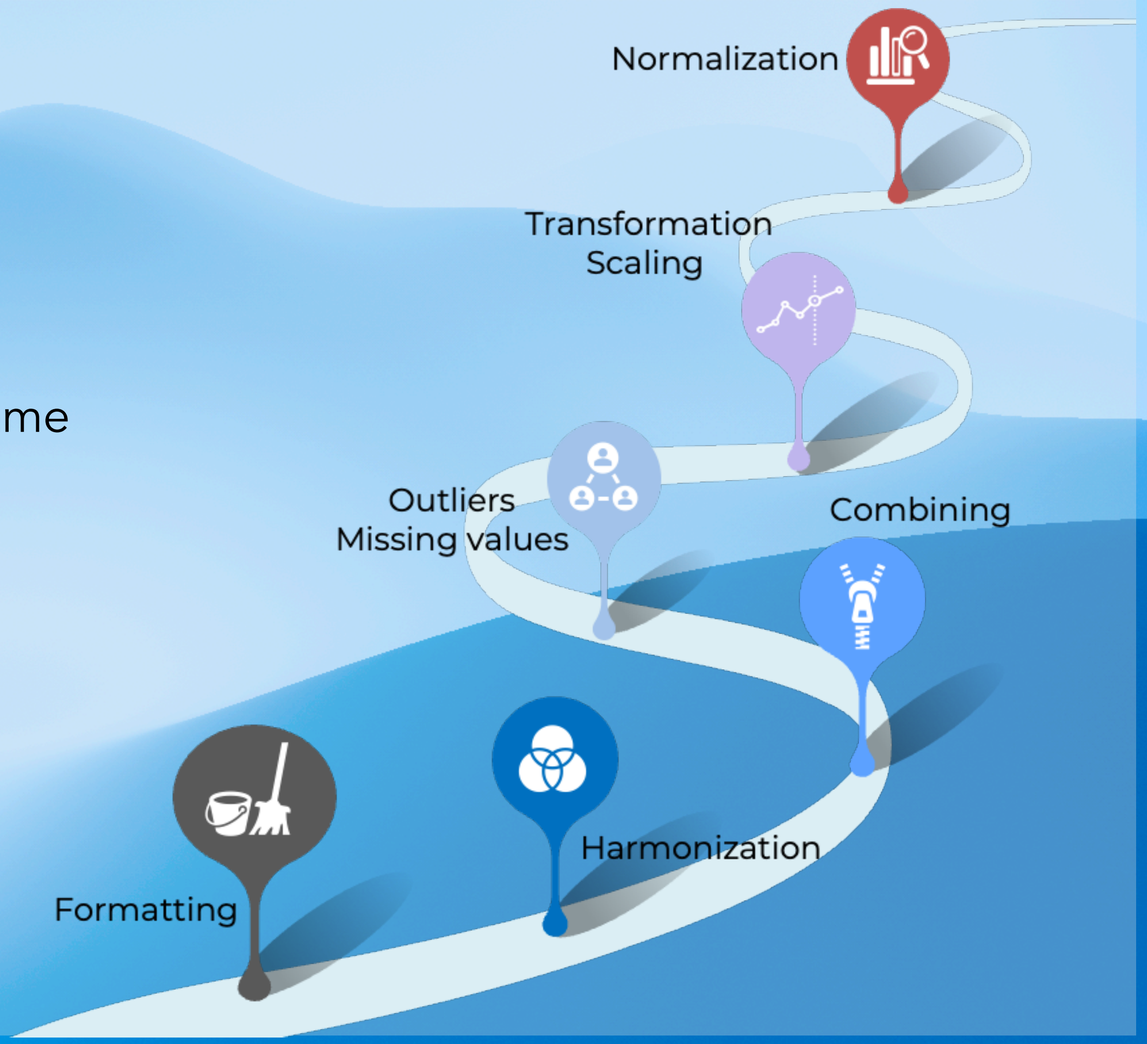
Data Cleanup

We want to clean up our data:

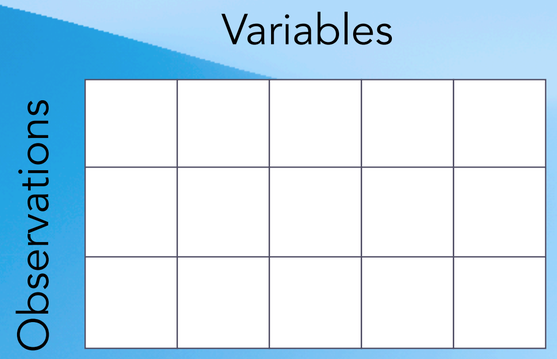
- Understand the structure of the data

- Fix Data Types
  - Ensure categorical variables are spelled the same
  - Change one type of variable to another type
- Extract, merge, add or remove variables

- Removing Duplicates
- Dealing with Outliers
- Identify Missing Values



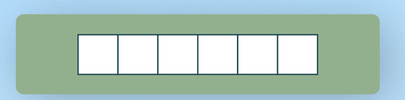
R Data Types & Structures



Data Structures

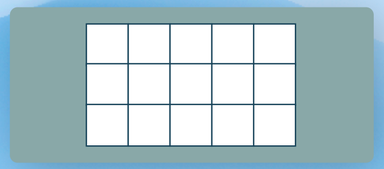
Vector

Age

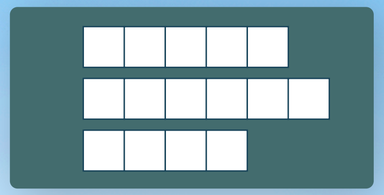


Data Frame

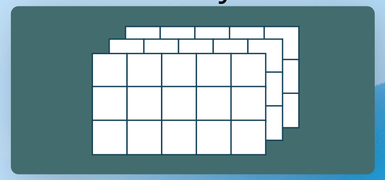
Age
Height
BMI



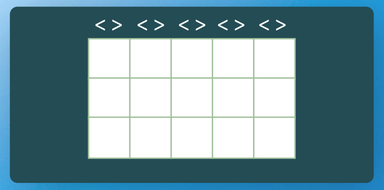
List



Array



Tibble



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

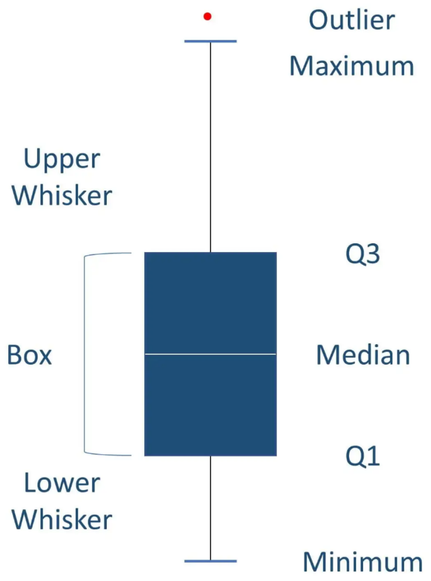
TRUE, FALSE

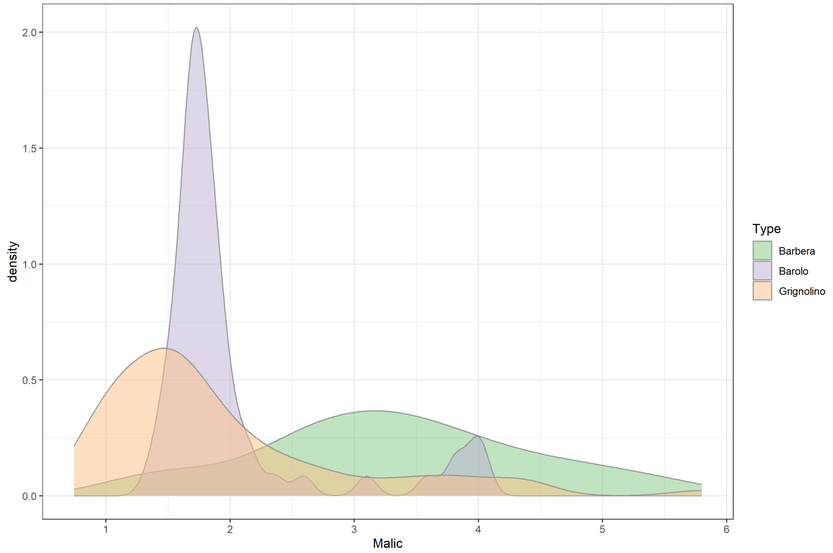
Basic Summary Stats

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





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)*

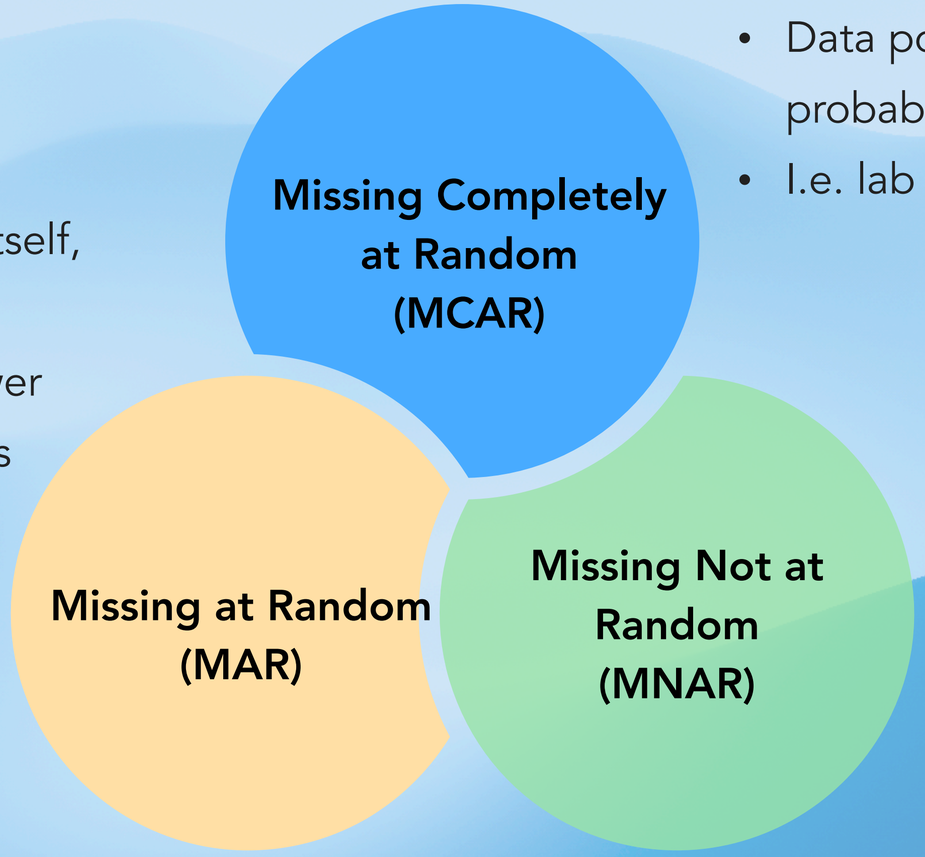
Types of Missing Values

MCAR:

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

MAR:

- Missing data point is not related to missing data itself, but to other variables.
- I.e. Health records in lower median income countries



MNAR:

- Missing data points are related to the reason why data are missing.
- I.e. Assessing depression through questionnaires

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)

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

→

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

Cheat Sheet - String manipulation

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

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

Exercise 1
Data Cleanup

Outlier Samples

- 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

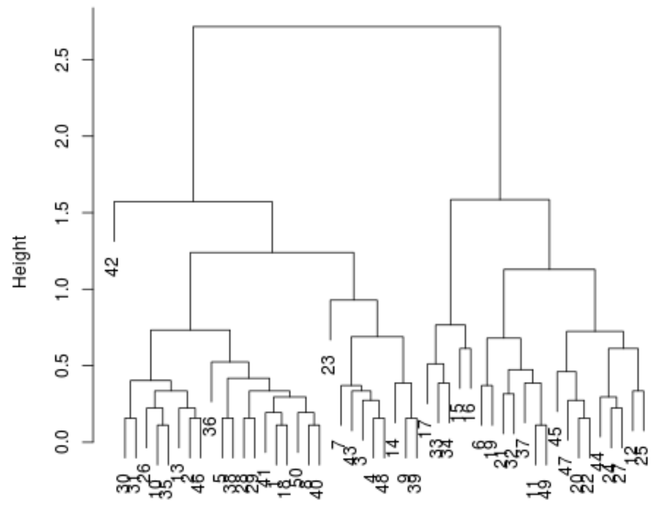
- When is something and outlier?
- If it doesn't make sense, you can:
  - Cap them at a reasonable threshold
  - Transform them (e.g., log transformation).

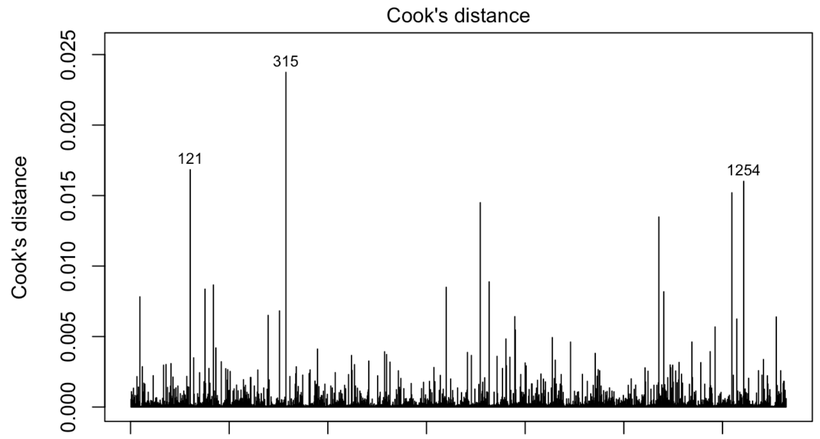
Outliers - Visual Inspection & Stats Test

- **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





Part 2
Data Wrangling & Integration

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 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

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

Nesting

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

```
tree_long_nested <- tree_long %>%
  group_by(`Type species`) %>%
  nest(Data = c(Site, `Average diameter (cm)`) %>%
  ungroup()
```

Type species	Data
Acer rubrum	<tibble>
Quercus alba	<tibble>
Pinus teada	<tibble>

```
tree_long_nested %>%
  filter(`Type species` == "Quercus alba") %>%
  pull(Data)
```

Site	Average diameter (cm)
Site A	29
Site B	17
Site C	14
Site D	42

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()
```

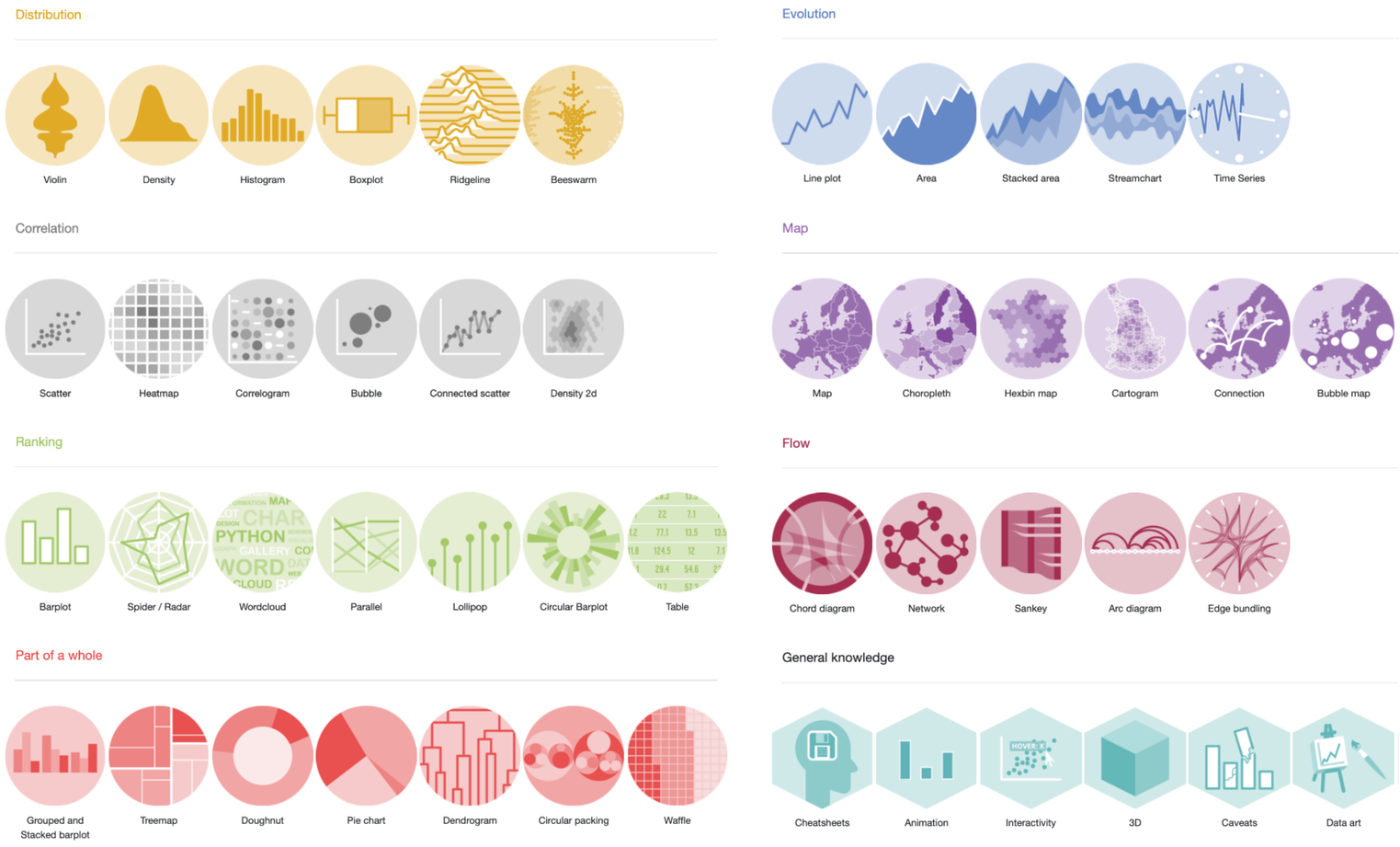
Cool New Plot?

CHART TYPES PKG BEST QUICK TOOLS ALL RELATED

The R Graph Gallery

Welcome to the R graph gallery, a collection of charts made with the R programming language. Hundreds of charts are displayed in several sections, always with their reproducible code available. The gallery makes a focus on the tidyverse and ggplot2. Feel free to suggest a chart or report a bug: any feedback is highly welcome! Note that if you like this gallery, you'll love my newsletter!

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

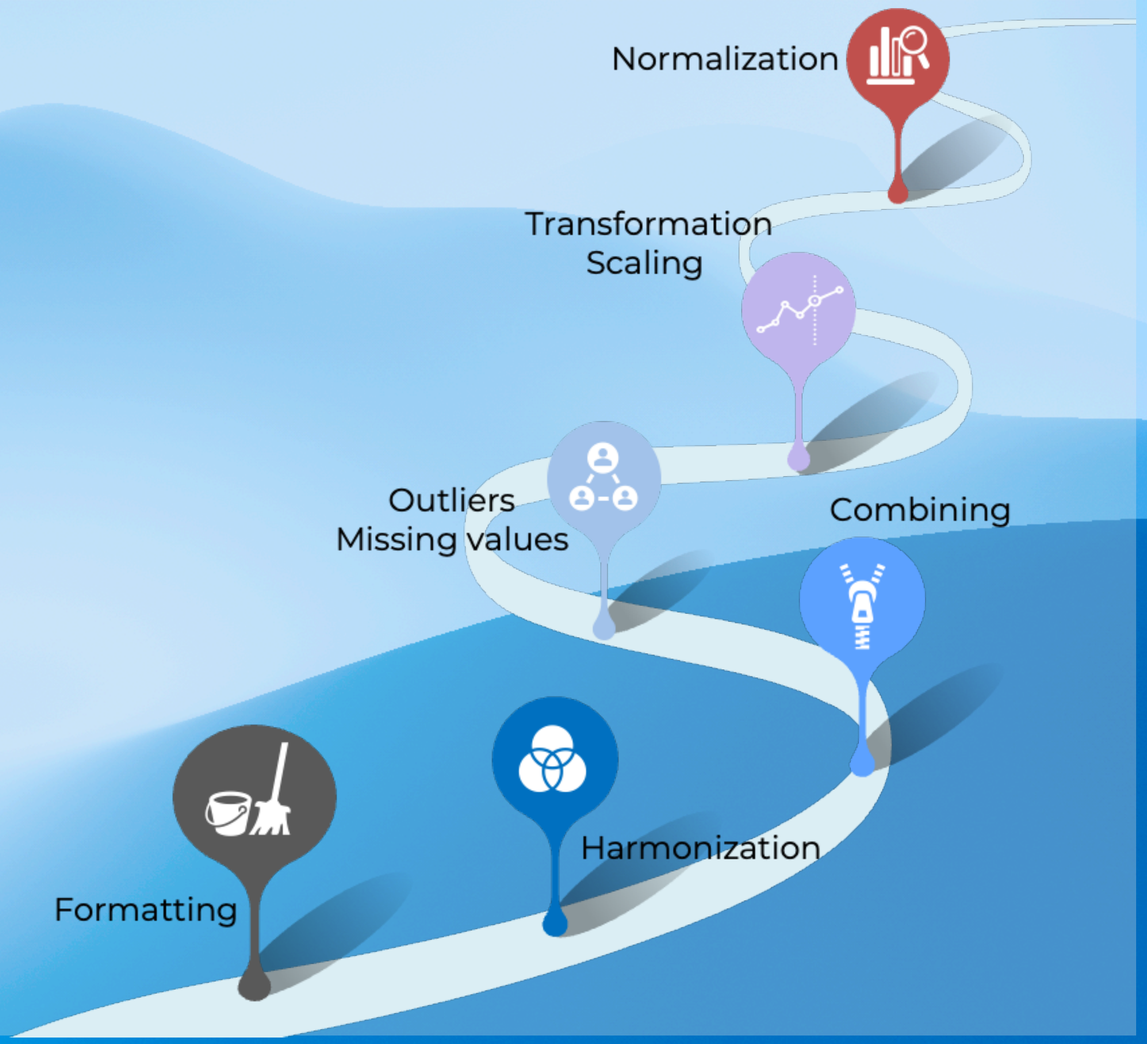


Univariate and Bivariate Analysis

Understand each variable

- Visualizing Distributions
- Categorical Data
- Dominant category or if the data is balanced

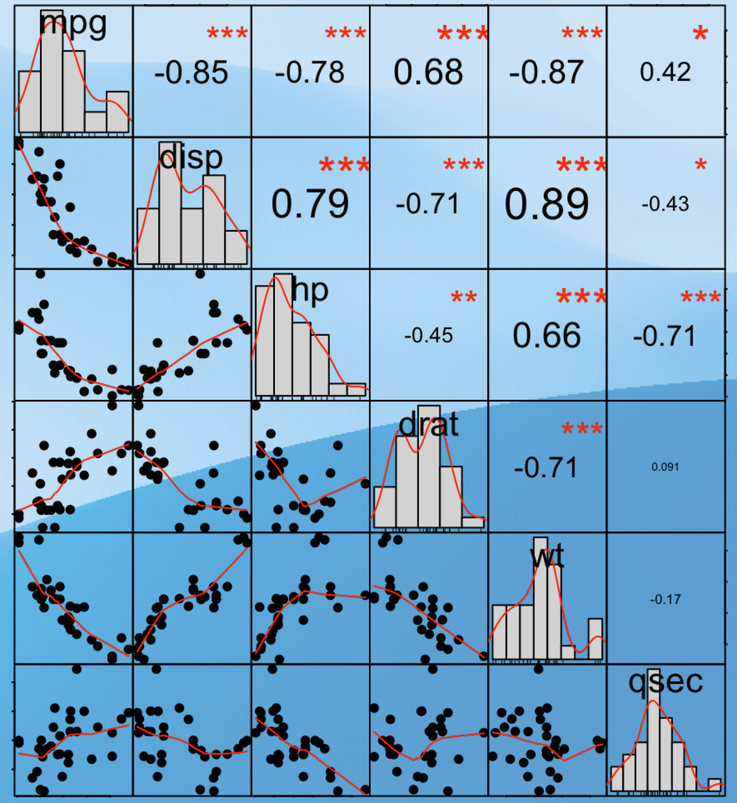
Individual and paired relationships



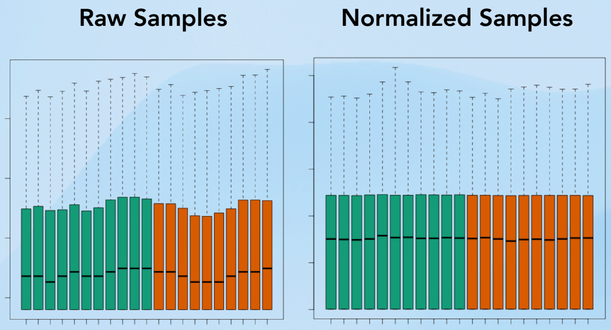
Correlation

- Investigate relationship between variables:
  - Correlation between explanatory variables and outcome variable
  - Informative, uninformative, latent variable relation-ships (redundant info)
- Drives hypothesis generation!

- If two variables are highly correlated (above 0.8 or below -0.8), they might be redundant in predictive models.
- In regression models, multicollinearity inflates variance and makes coefficients unreliable.

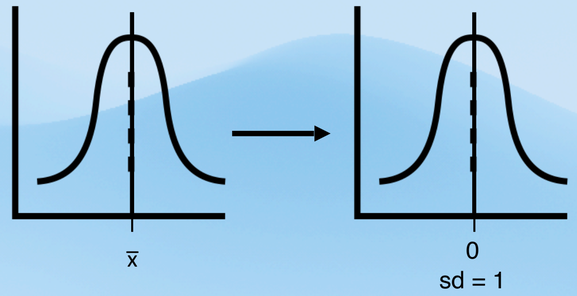


Normalization



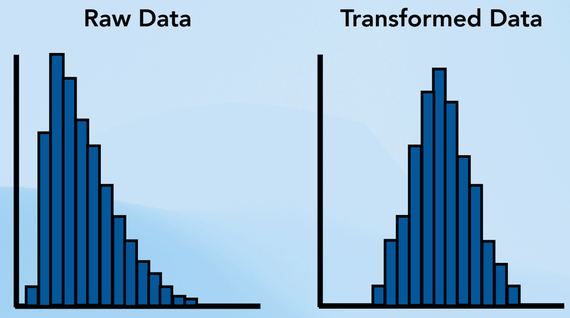
- Can use for not normally distributed data.
- Variables do not get zero centered.
- Normalization within a range (max, min).
- Affected by outliers.

Standardization



- For normally distributed data
- Scaled to a mean of 0 and standard dev. of 1.
- Not constrained to a range.
- Not affected by outliers.

Transformation



- Attempt to make data normally distributed
- Often logarithmic transformations
- Squeezes outliers for less impact on model.

Exercise 2
Data Wrangling & Integration

Part 3
Exploratory Data Analysis

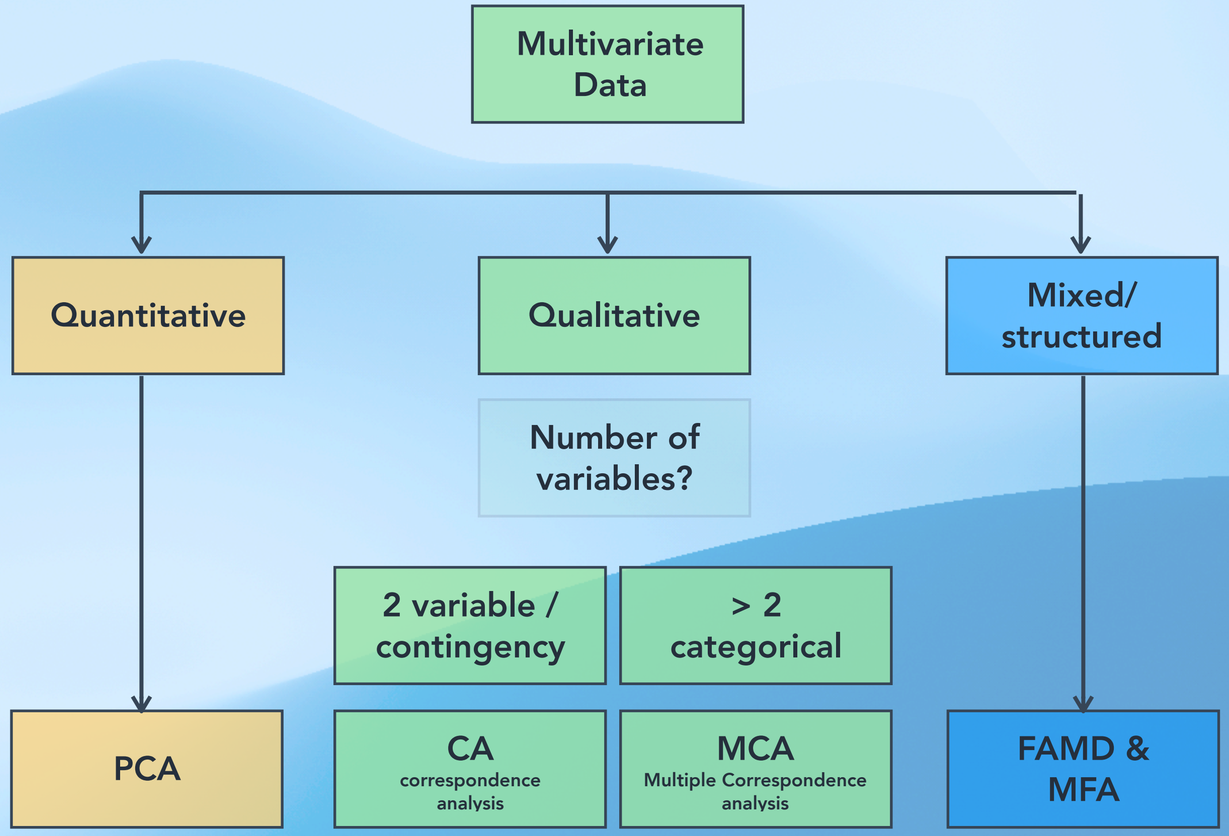
Exploratory Data Analysis (EDA)

- Identify variable type, ranges and distributions
- Identify an remedy data quality issues
- Investigate relationships between variables
- Missing values
- Outlier samples, batch effects
- Discover data patterns (groups) that inform further analysis or modeling

Principal Component Methods

- Dimension reduction
- Classical methods
- PCA, CA and MCA
- (continuous variables, contingency table and qualitative variables, respectively).

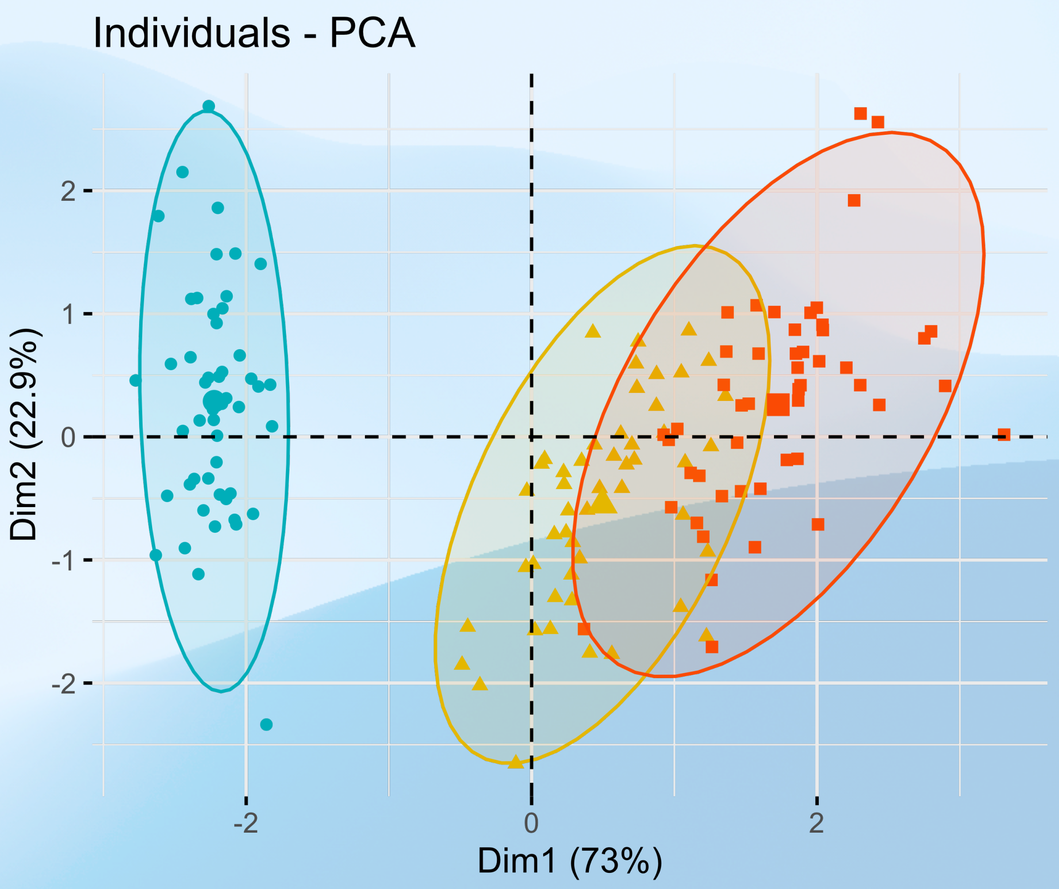
- Advanced methods -
- FAMD, MFA and HMFA -
- a mix of variables (qualitatives & quantitatives)



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	…
…	…	…	…	…	…



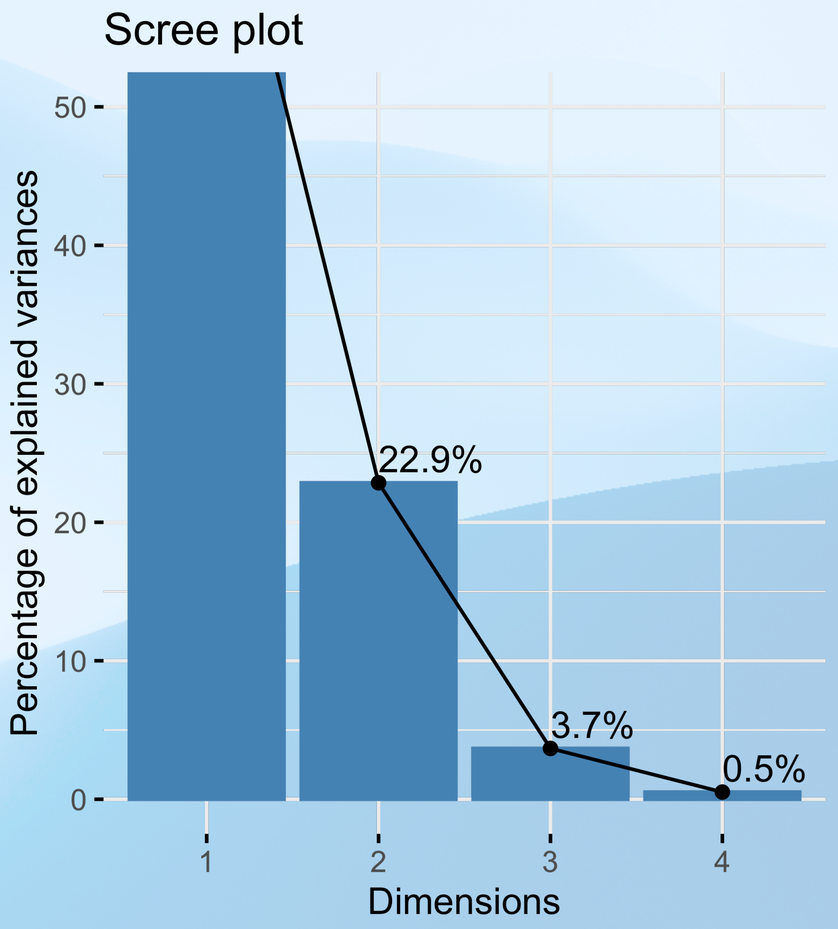
Good and simple guide to PCA in R: https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html

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	…
…	…	…	…	…	…

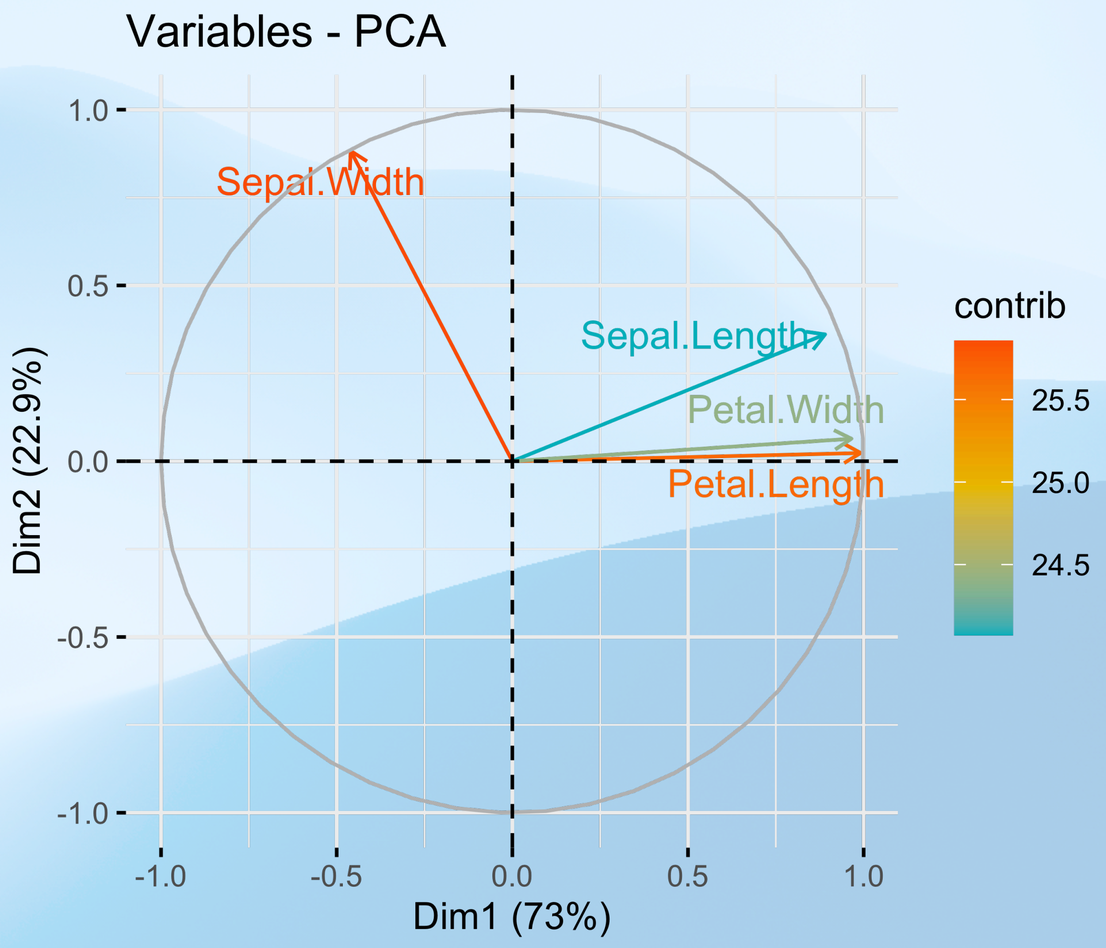
Good and simple guide to PCA in R: https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html



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.

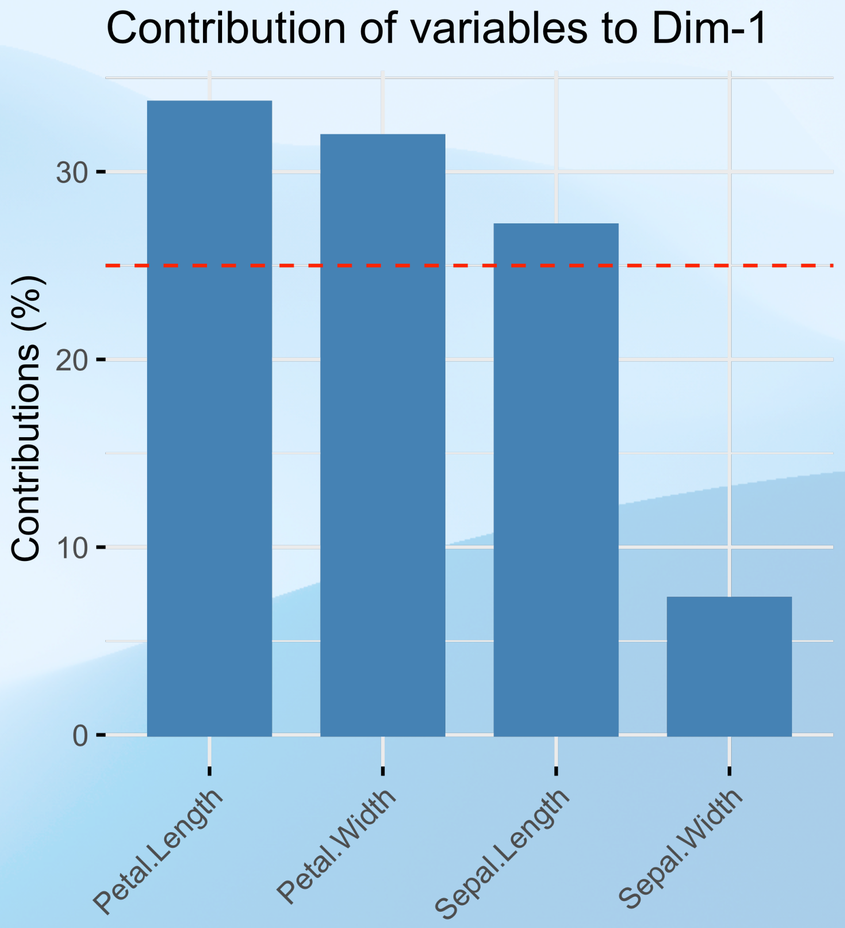
	PC1	PC2
Sepal Length	+	-
Sepal Width	-	-
Petal Length	+	-
Petal Width	+	-



Biplot: PCA plot with loadings

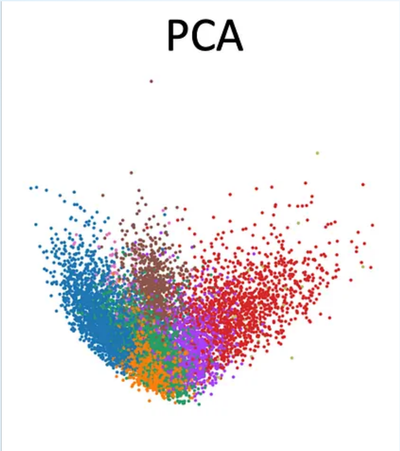
- Loadings: Arrows represent the direction and magnitude of each variable's contribution to the PC's.
- The length of the arrow indicates the strength of the variable's contribution.
- The direction of the arrow shows how the variables correlates with each PC.

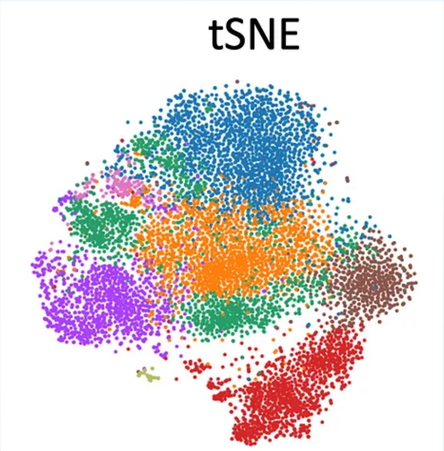
	PC1	PC2
Sepal Length	+	-
Sepal Width	-	-
Petal Length	+	-
Petal Width	+	-

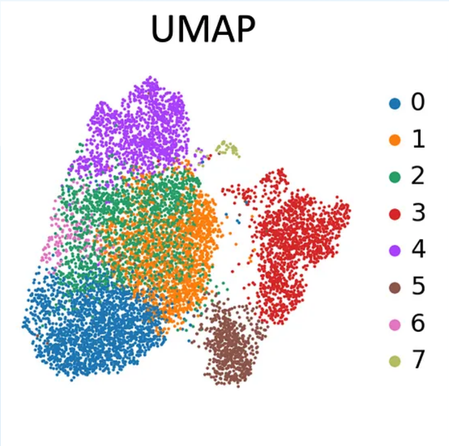


Correlation

- 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







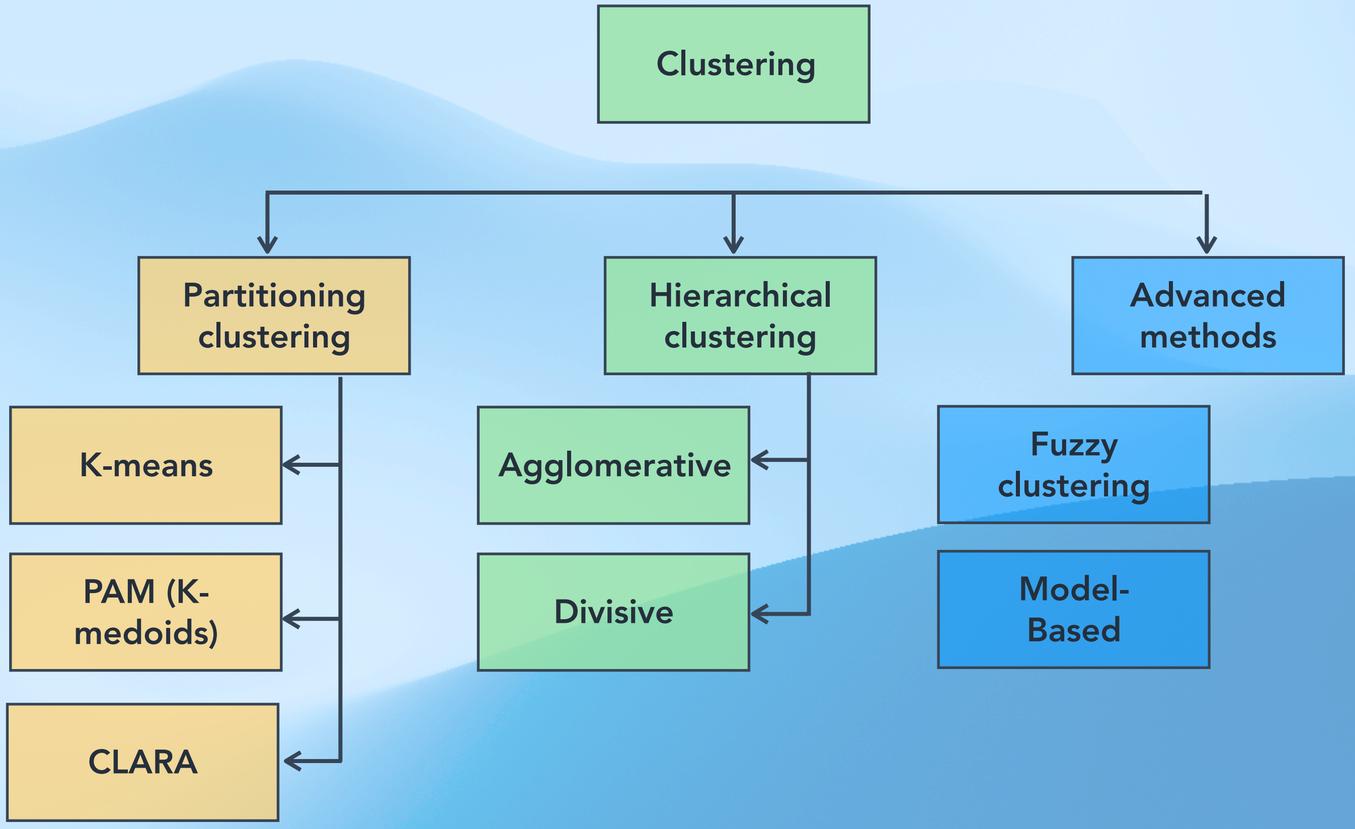
Exercise 3A
Exploratory Data Analysis

Clustering

- distance measures,

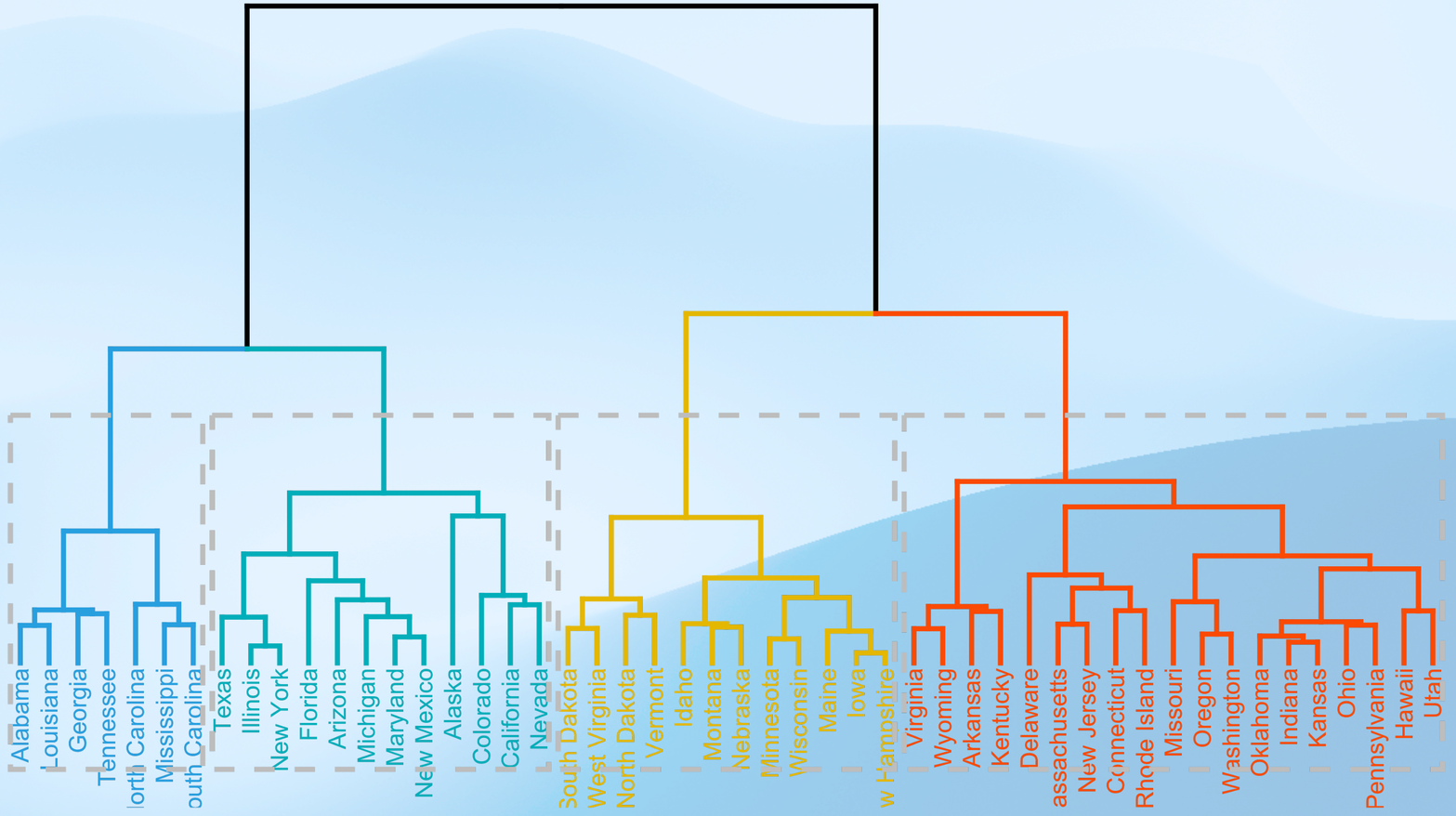
- Partitioning methods
- Hierarchical clustering
- Fuzzy clustering
- Density-based clustering
- Model-based clustering

- cluster validation methods,
-



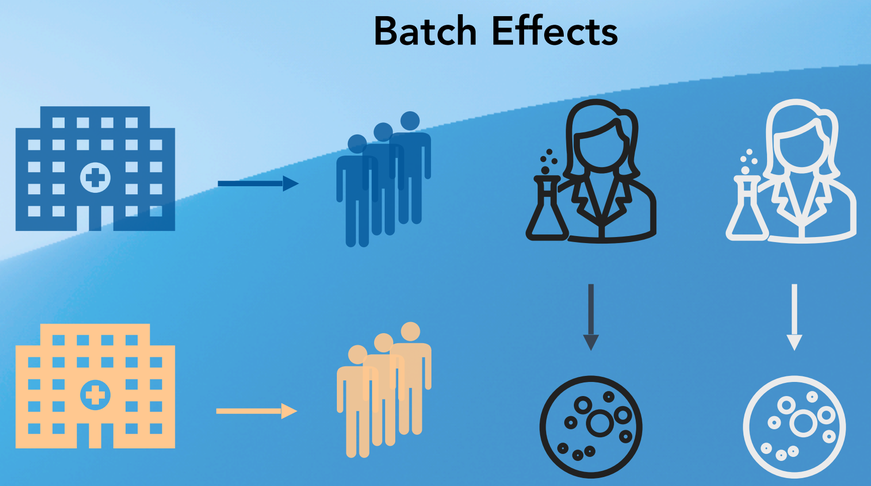
Clustering

- cluster validation methods,
-



Batch Effects

- Batch effects arise from: data collection, handling, storage, experimental procedure, etc.
- Not natural/biological variation - should be ‘removed’!



Exercise 3B
Exploratory Data Analysis

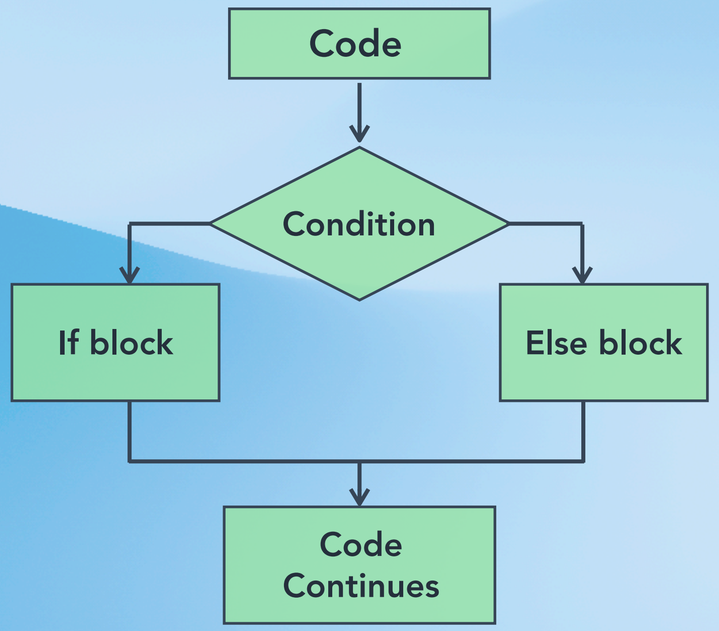
See you tomorrow!

Part 4
Scripting in R

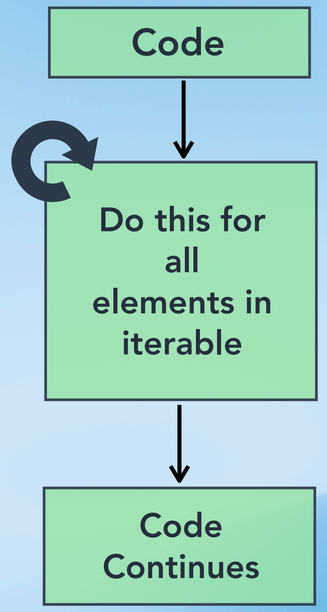
Reuseable, Short and Functional Code

Scripting in R

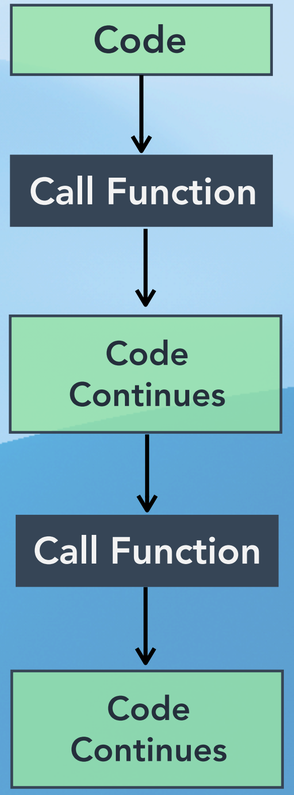
Conditionals



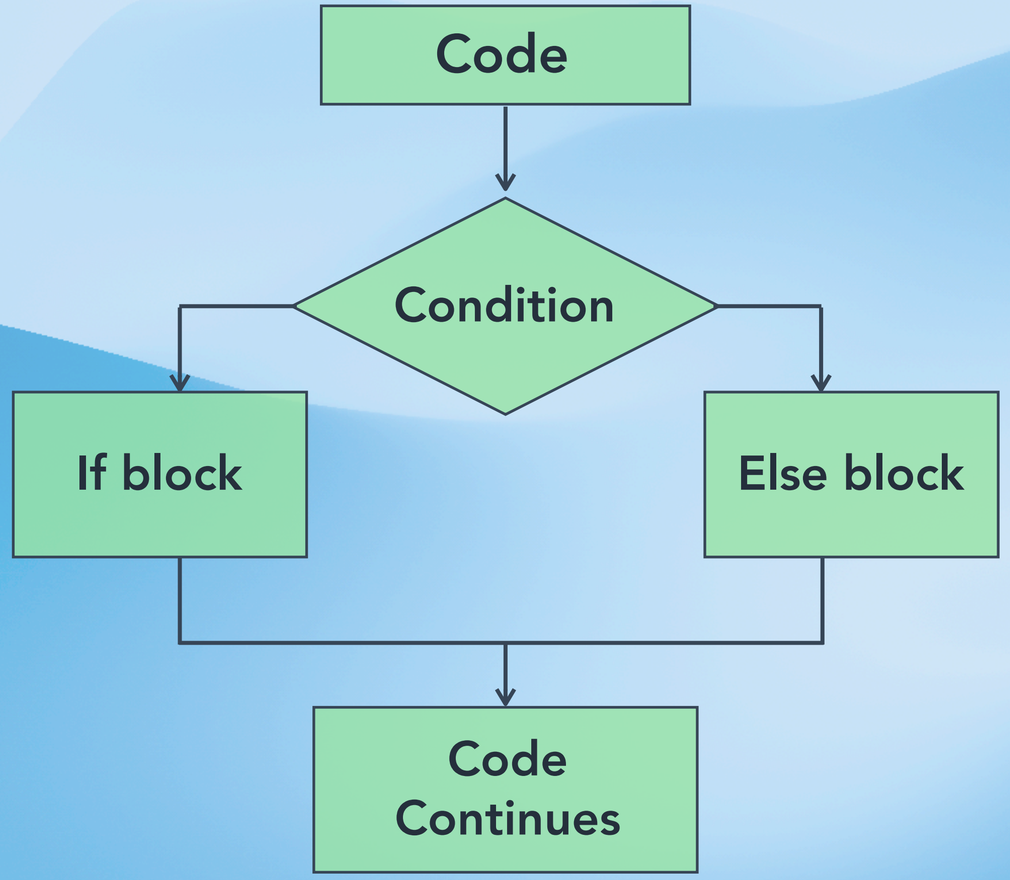
Loop



Functions



Conditionals (if-else statements)



Symbol	Meaning
> , <	Great, 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

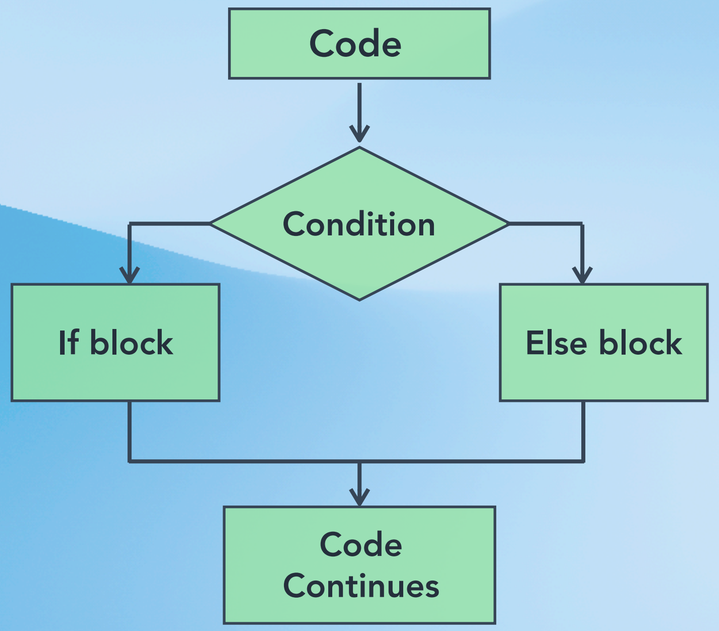
```
var1 > var2
var1 < var2
var1 == var2
var1 != var2 # Not equal to
var1 - var2 == var3
is.numeric(var1), is.character(var1), …
```

Combine conditions

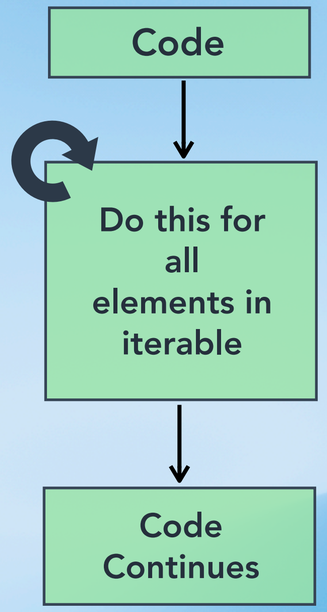
```
var1 > var2 & var1 < var3
var1 == var2 | var1 == var3
(var1 > var2 & var1 < var3) | (var1 == var2)
```

Scripting in R

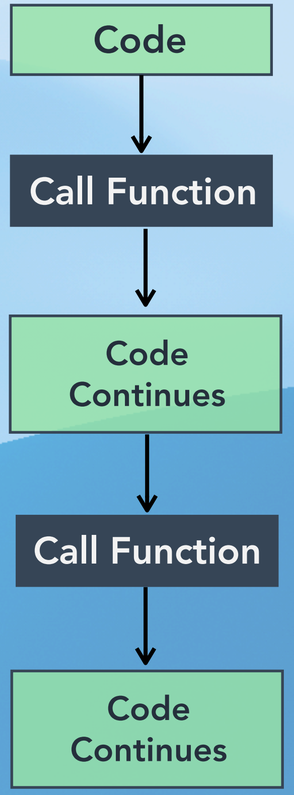
Conditionals



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

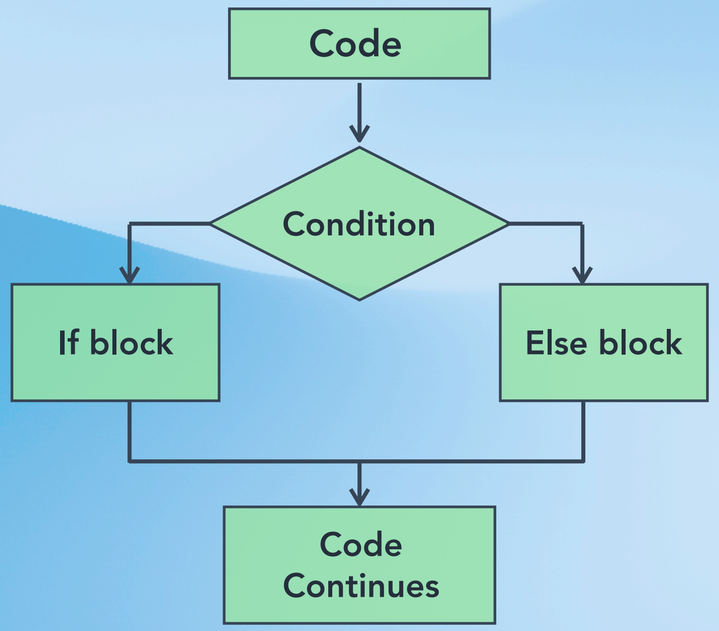
- Looping in R *can* be slow if data is large!
- Alternative is the family of **apply functions:** apply a function to every; value, list element or row/column.
- These functions are written in C code - fast!

Apply Functions

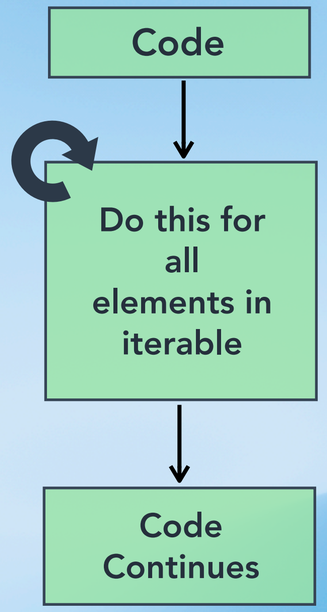
```
# Input is a matrix, specify rows or cols. Output is a list:
apply(df, 1, FUN = mean)

# 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

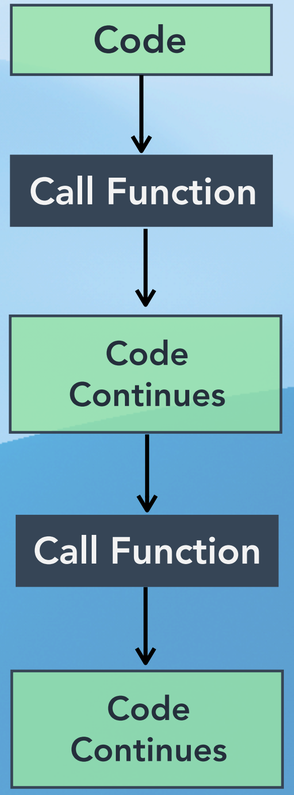
Conditionals



Loop



Functions



Functions

Pros:

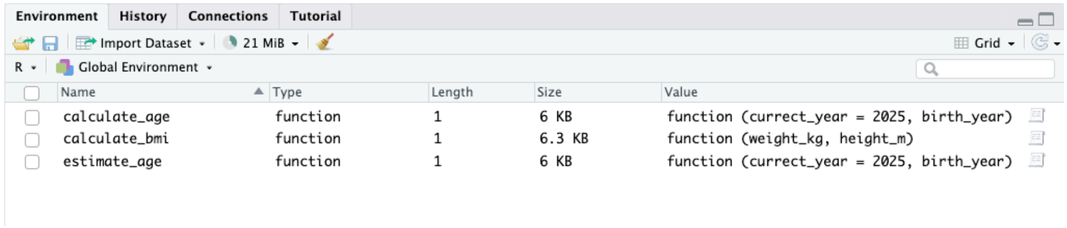
- Reusable: Write once, use many times.
- Readable: Code is clean and easy to understand.
- Modular: Breaks down complex tasks into steps.

Code Example:

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

Globale variables

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



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

functions.R | main_script.qmd

Source on Save

```r
# Packages should be loaded in the main script.

# Function to calculate BMI
# Required packages: none
calculate_bmi <- function(weight_kg, height_m){

  bmi <- weight_kg/height_m^2

  return(bmi)

}

# Function to estimate age
# Required packages: none
estimate_age <- function(currect_year = 2025, birth_year){

  age <- currect_year - birth_year

  return(age)

}
```

functions.R | main_script.qmd

Render on Save | Render

Source | Visual

```r
---
title: "Main Script"
format: html
---

## Load packages

## Source script
```{r}
source('./functions.R')
```

Calculate BMI.
```{r}
calculate_bmi(weight_kg = 70, height_m = 1.80)
```

```
[1] 21.60494
```

Estimate age in 2025 when born in 1997.
```{r}
estimate_age(birth_year = 1997)
```

```
[1] 28
```

Estimate age in 2005 when born in 1997.
```{r}
estimate_age(currect_year = 2005, birth_year = 1997)
```

```
[1] 8
```
```

Functions and Error Handling

functions.R | main_script.qmd

Source on Save

```r
# Function to calculate BMI, with control points and error checking.
# Required packages: none
calculate_bmi_2 <- function(weight_kg, height_m) {
  # Check if weight and height are numeric
  if (!is.numeric(weight_kg) | !is.numeric(height_m)) {
    stop("Both weight_kg and height_m must be numeric values.")
  }

  # Check if weight and height are positive
  if (weight_kg <= 0) {
    stop("Weight must be a positive value.")
  }
  if (height_m <= 0) {
    stop("Height must be a positive value.")
  }

  # Calculate BMI
  bmi <- weight_kg / height_m^2

  # Check if BMI is within a reasonable range
  if (bmi < 10 | bmi > 60) {
    warning("The calculated BMI is outside the normal range. Please check your input values.")
  }

  return(bmi)

}
```

functions.R | main_script.qmd

Render on Save | Render

Source | Visual

The BMI function with out error handling returns a meaningless BMI value if given a negative weight.

```{r}
calculate_bmi(weight_kg = -50, height_m = 1.80)
```

```
[1] -15.4321
```

The BMI function with error handling returns an error if given a negative weight.

```{r}
calculate_bmi_2(weight_kg = -50, height_m = 1.80)
```

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

The BMI function with error handling returns an warning if a BMI outside the normal range is calculated.

```{r}
calculate_bmi_2(weight_kg = 25, height_m = 1.80)
```

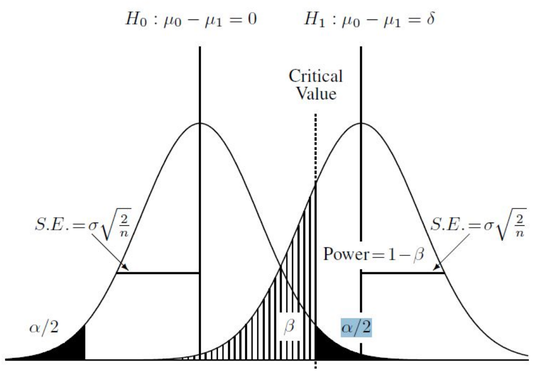
```
Warning in calculate_bmi_2(weight_kg = 25, height_m = 1.8) :
  The calculated BMI is outside the normal range. Please check your input values.
[1] 7.716049
```

Exercise 4
Scripting in R

Part 5
Modelling in R

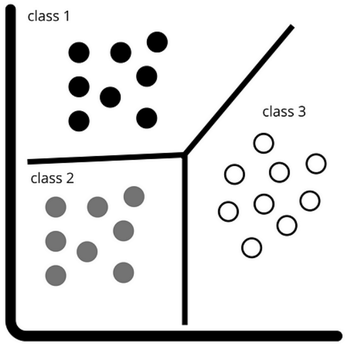
Types of models

Hypothesis Testing



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

Prediction / Classification

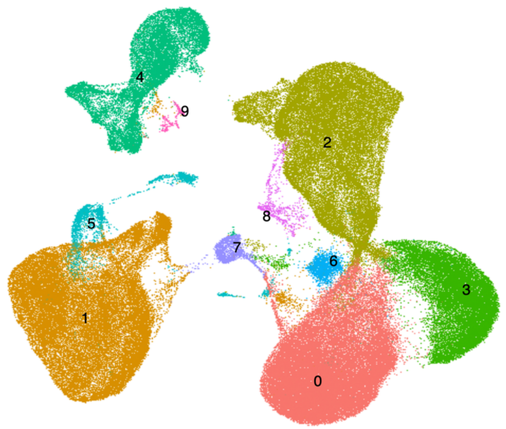


Classification Model

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

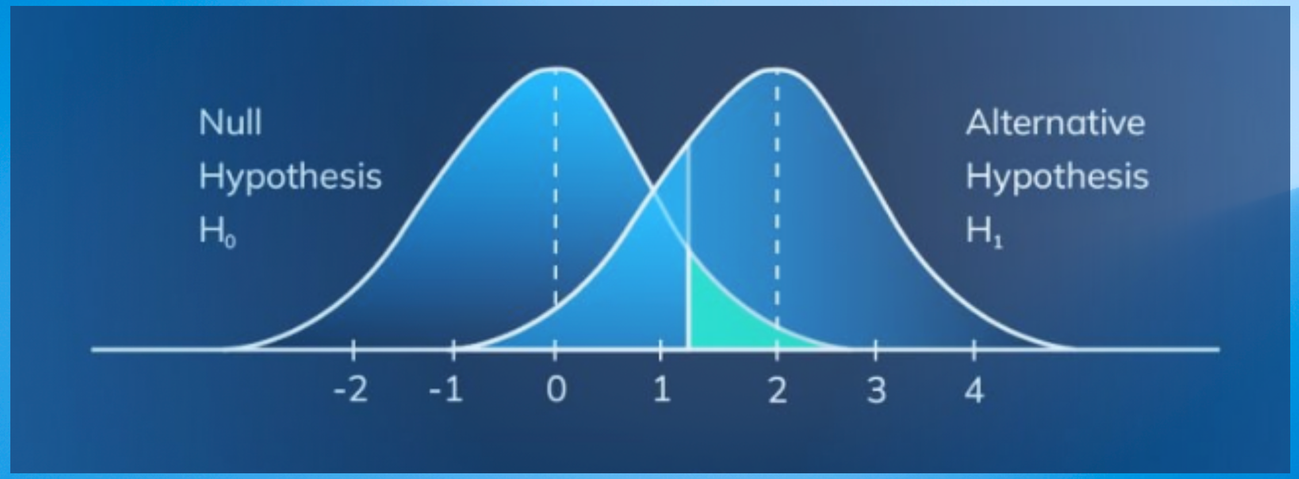
Questions that boil down to:

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:

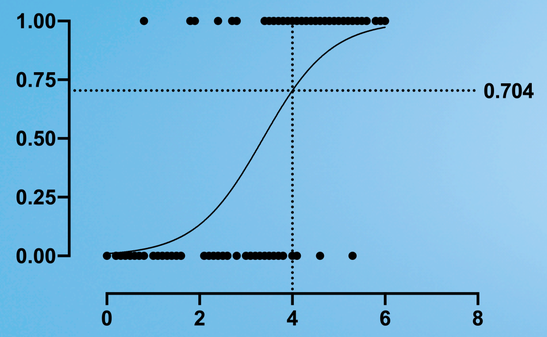
- Student's t-test
- F-test
- ANOVA / MANOVA
- Fisher's exact test



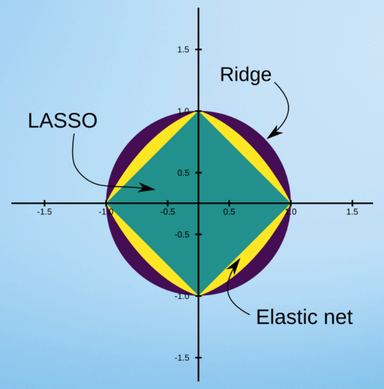
Predictive Modelling & Classification

- 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)

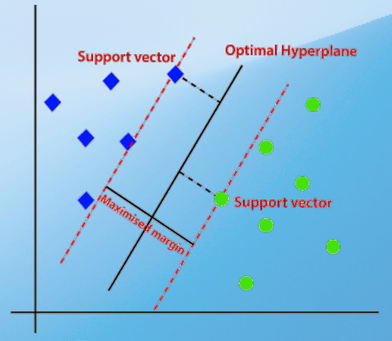
Logistic Regression



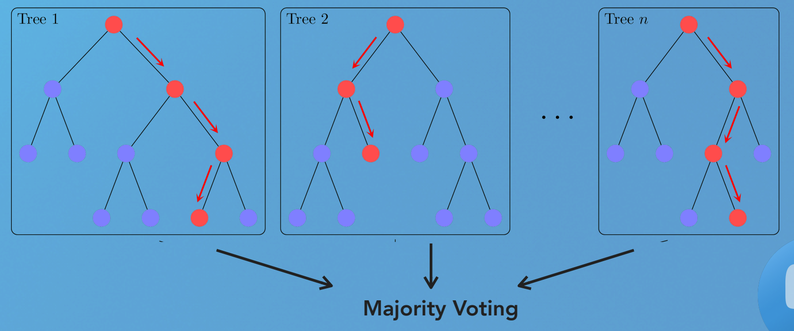
Penalized Regression



Support Vector Machine

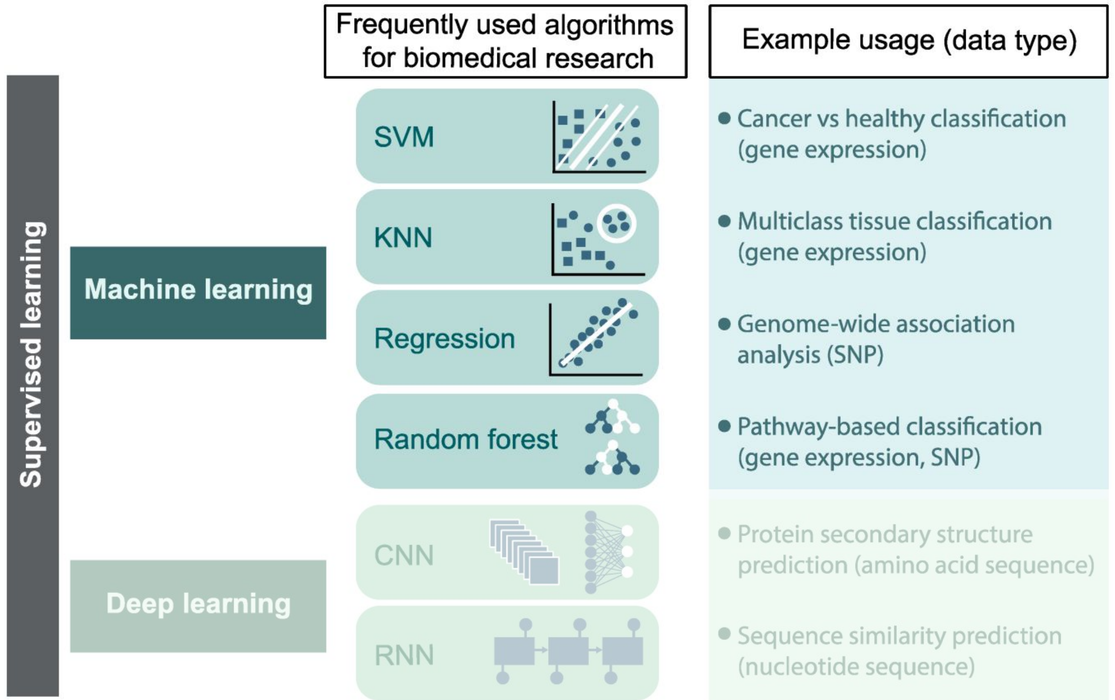


Random Forest



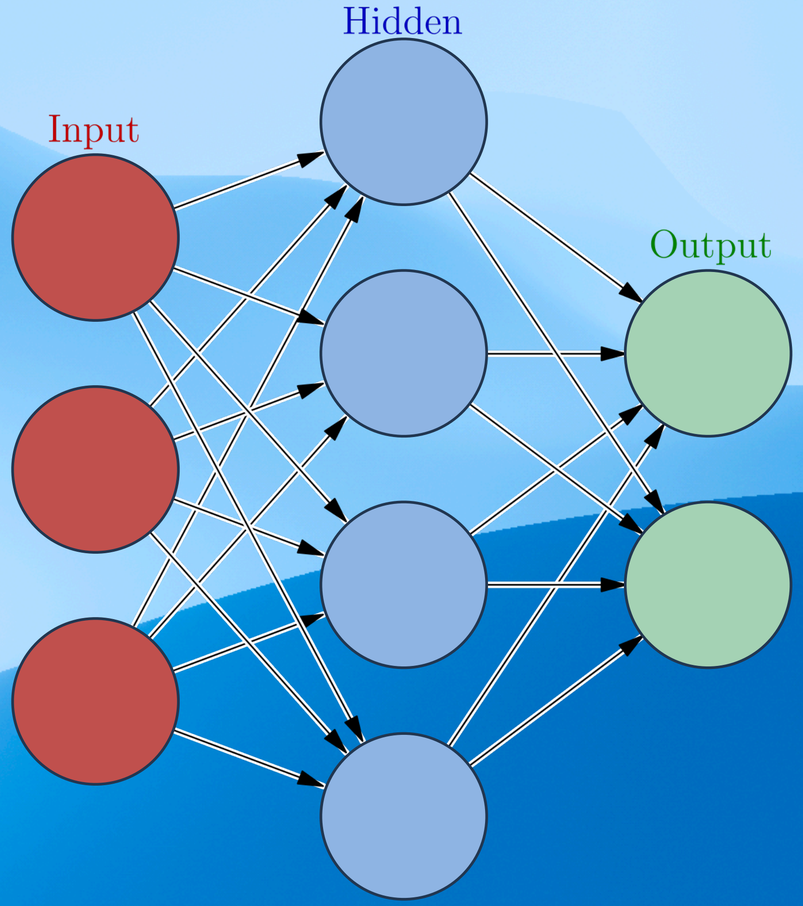
Supervised Learning

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



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.
- NN can be difficult to understand in detail but not in broad strokes, let's try…

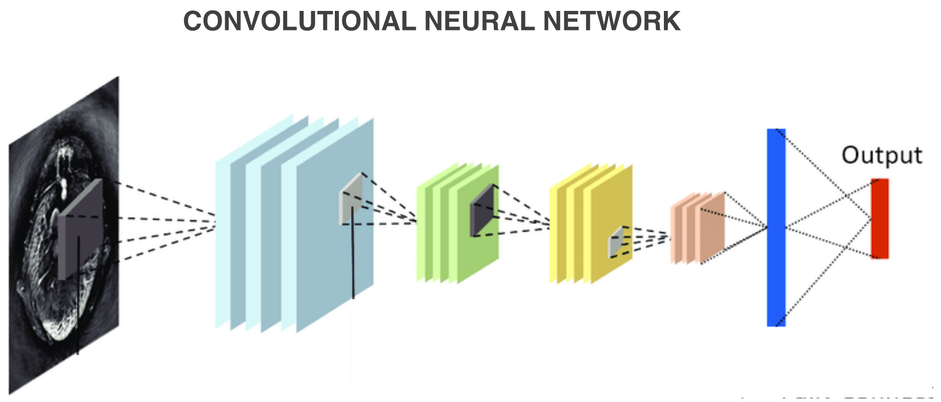


Medical Image Analysis

- 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:

ChatGPT

- 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

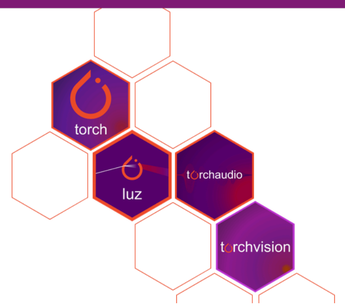
H2O
Version 3.26.0.8

Deep Neural Network - Supervised Image Classification in H20 R

Deep Neural Networks (or Deep Dearning) is based on a multi-layer, feed-forward artificial neural network that is trained with stochastic gradient descent using back-propagation. The network can contain many hidden layers consisting of neurons with activation functions. Advanced features such as adaptive learning rate, rate annealing, momentum training, dropout, L1 or L2 regularization, checkpointing, and grid search enable high predictive accuracy. Each computer node trains a copy of the global model parameters on its local data with multi-threading (asynchronously) and contributes periodically to the global model via model averaging across the network.

In this tutorial will show how to implement a Deep Neural Network for pixel based supervised classification of Sentinel-2 multispectral images using the H20 package in R.

torch for R ECOSYSTEM GET STARTED USE TORCH GET TECHNICAL RESOURCES DOCS



TORCH FOR R

An open source machine learning framework based on PyTorch. torch provides fast array computation with strong GPU acceleration and a neural networks library built on a tape-based autograd system. The 'torch for R' ecosystem is a collection of extensions for torch.

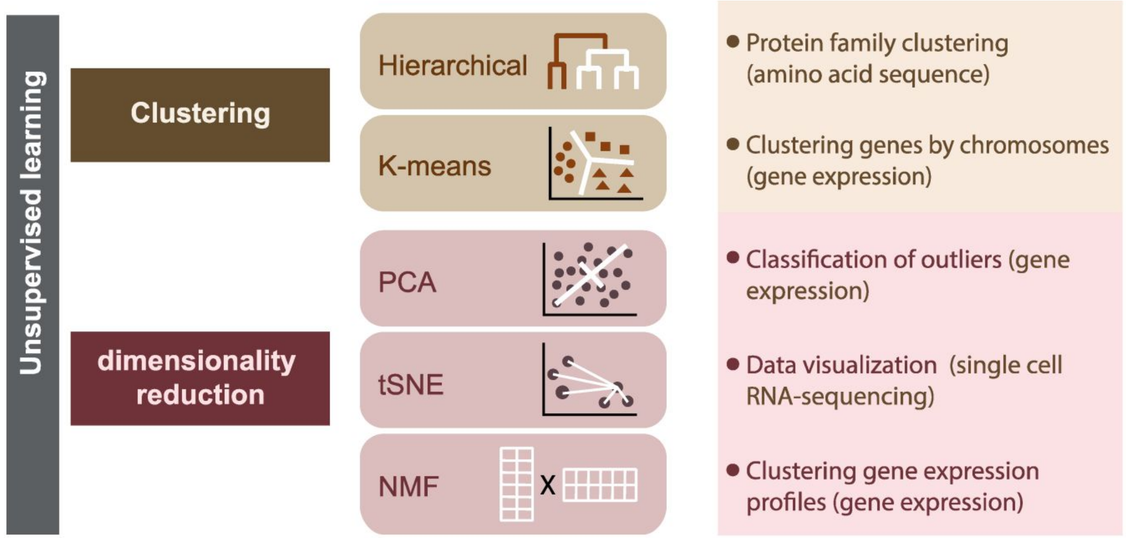
Deep Learning in Python

PyTorch

TensorFlow + Keras

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"

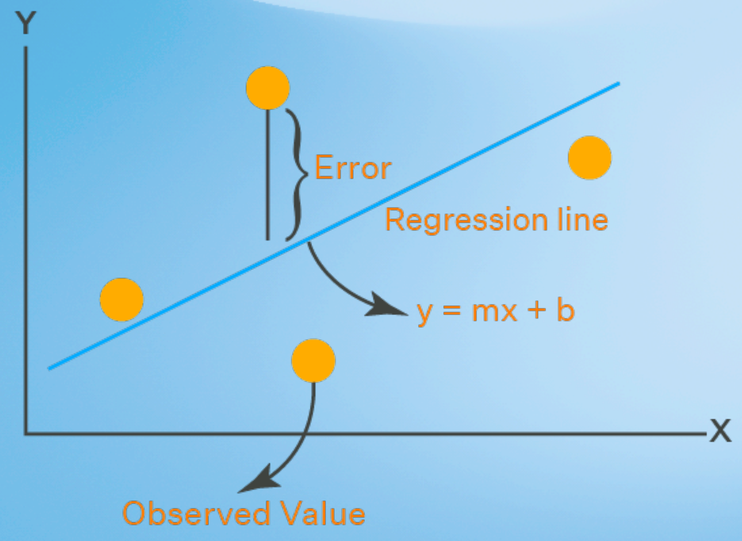
- Community-driven, open-source R packages for bioinformatics
- RNA-seq, GWAS, epigenetic, proteomics, annotation, ect.
- BiocManager in the install wrapper for Bioconductor:
  `biocmanagaer::install("package")`

Part 5
Modelling Evaluation

Model Evaluation Metrics

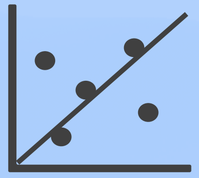
How well does the model fit the data?

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



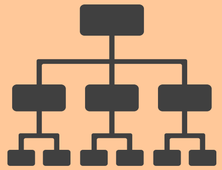
REGRESSION

- R-squared
- MSE
- RMSE
- RMSLE



CLASSIFICATION

- Accuracy
- Precision
- Recall
- ROC/AUC



CLUSTERING

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

Train & Test Dataset

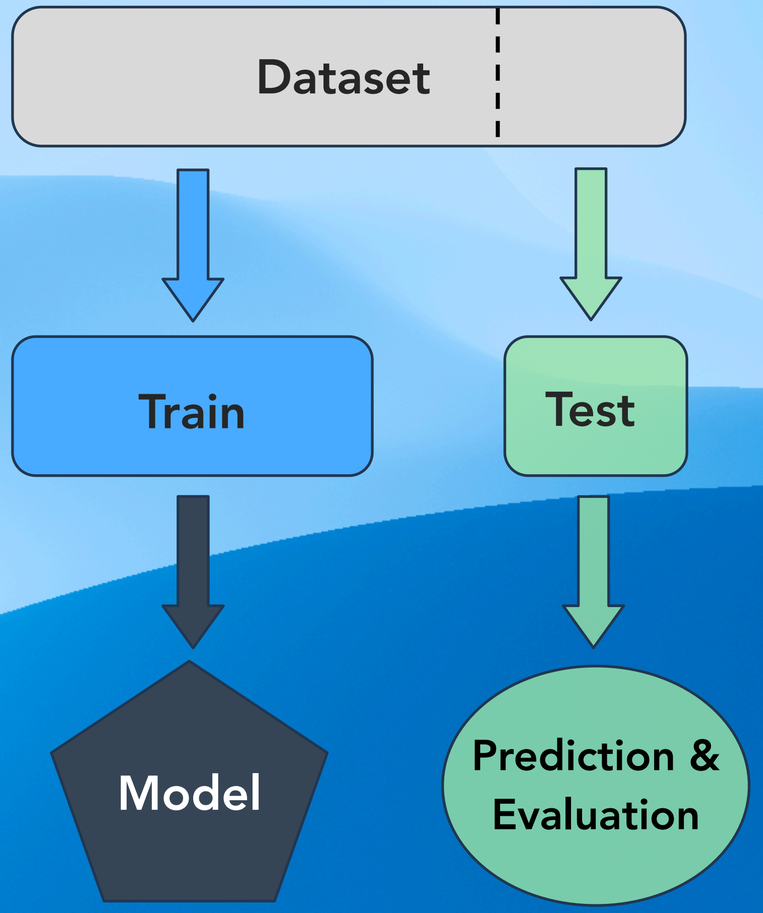
- We split our dataset into a **training set** and a **test set**

- **Training set** is used to train (estimate the parameters of) our model
- **Test set** is used to evaluate the utility of our model

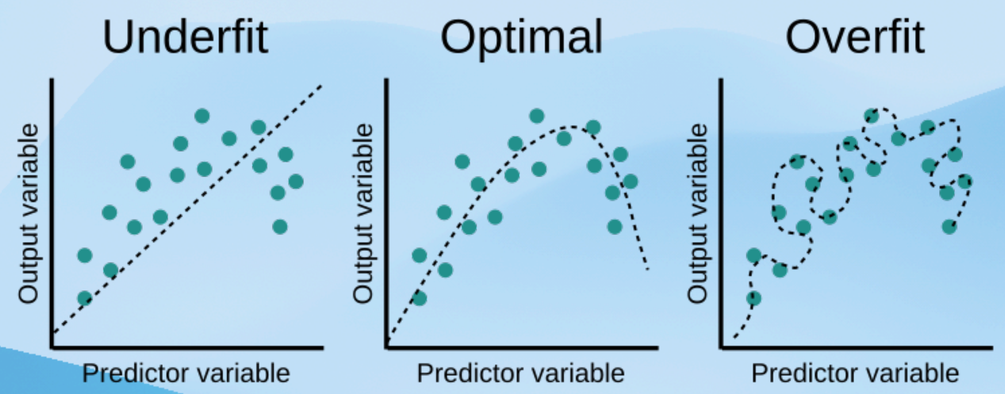
- **N.B:** Sets must be balanced and contain all observed classes!

R-syntax:

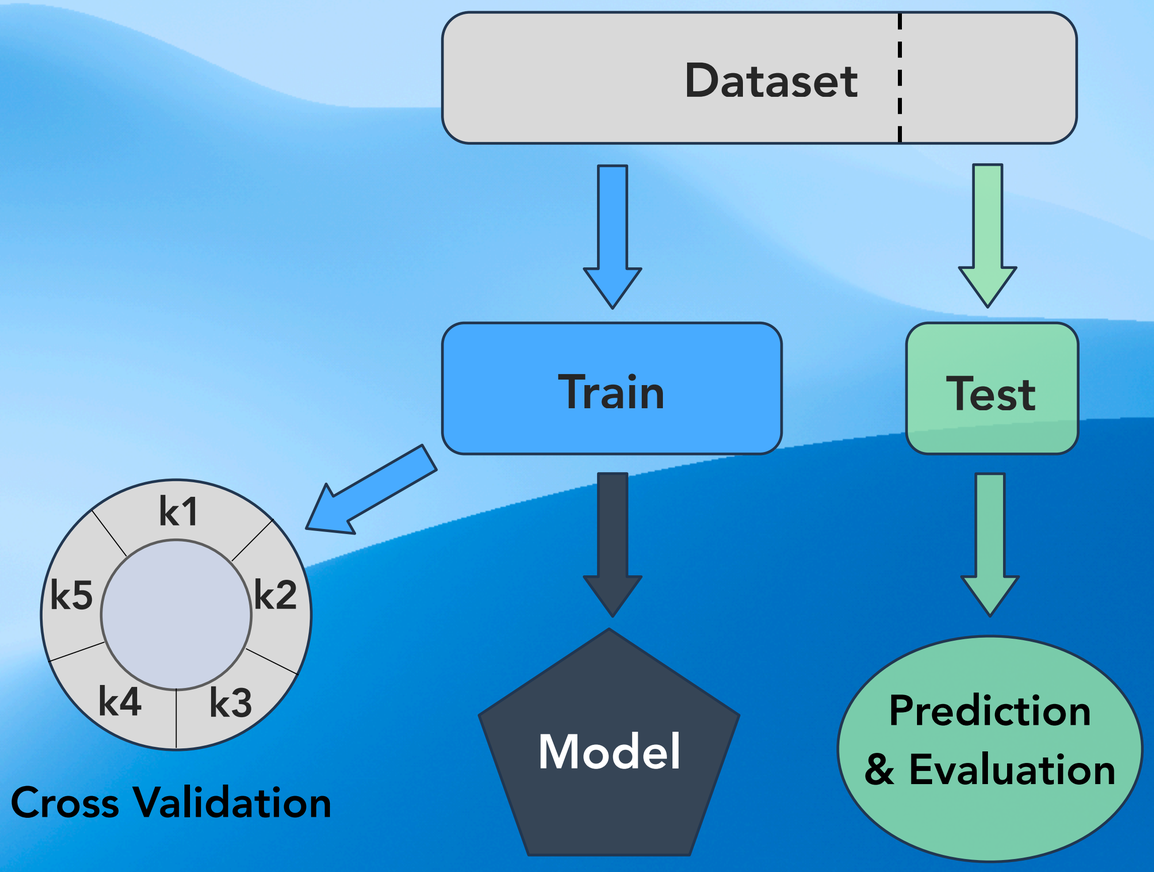
```
data <- read_csv("data/data.csv")

library(caret)
split <- createDataPartition(data$mpg, p = 0.8, list = FALSE)

train_data <- data[split, ]
test_data <- data[-split, ]
```



Model Overfitting

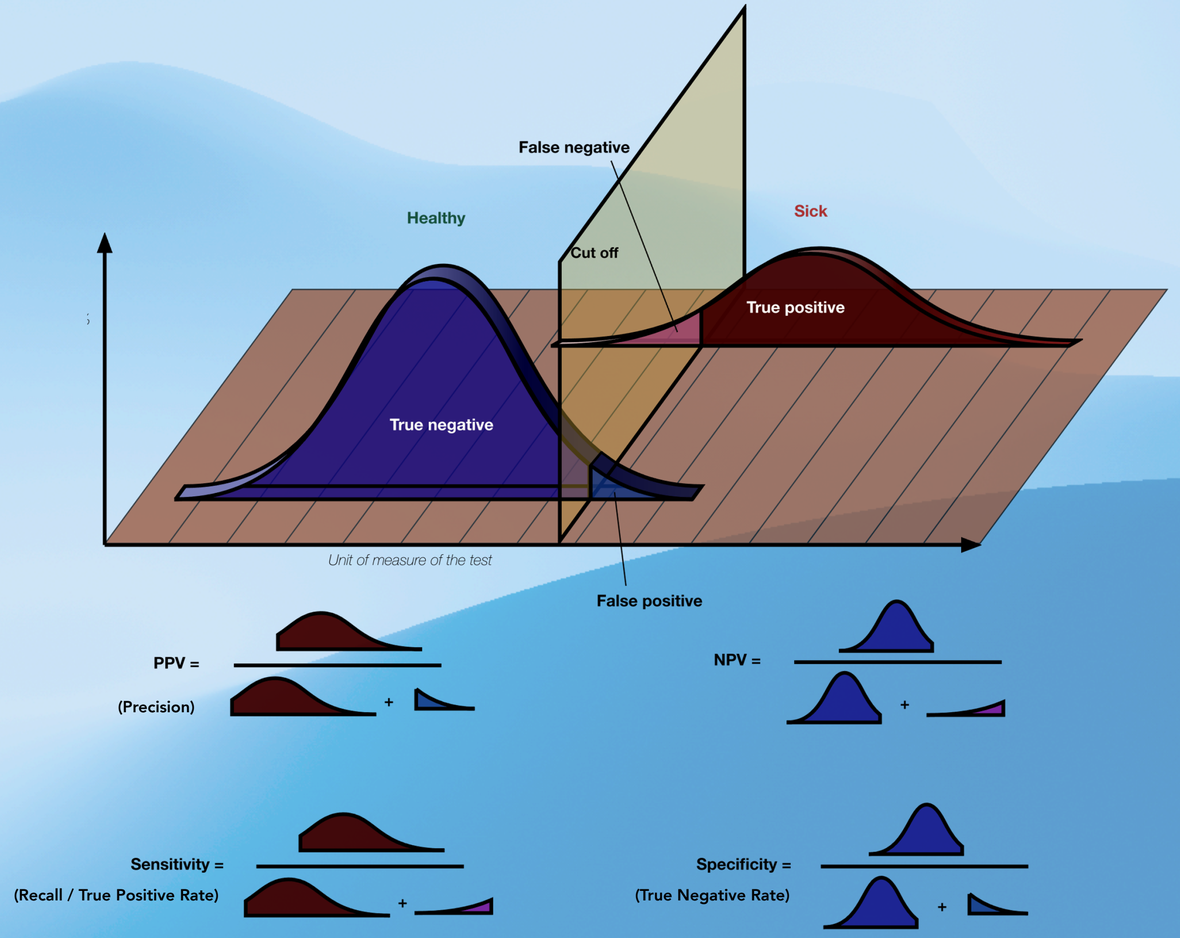


- When a model follows the data too closely we get an effect known as overfitting.
  - Splitting the data into training and evaluation we partly avoid this
  - By performing **cross validation** while training we may avoid it further



Classifier Performance

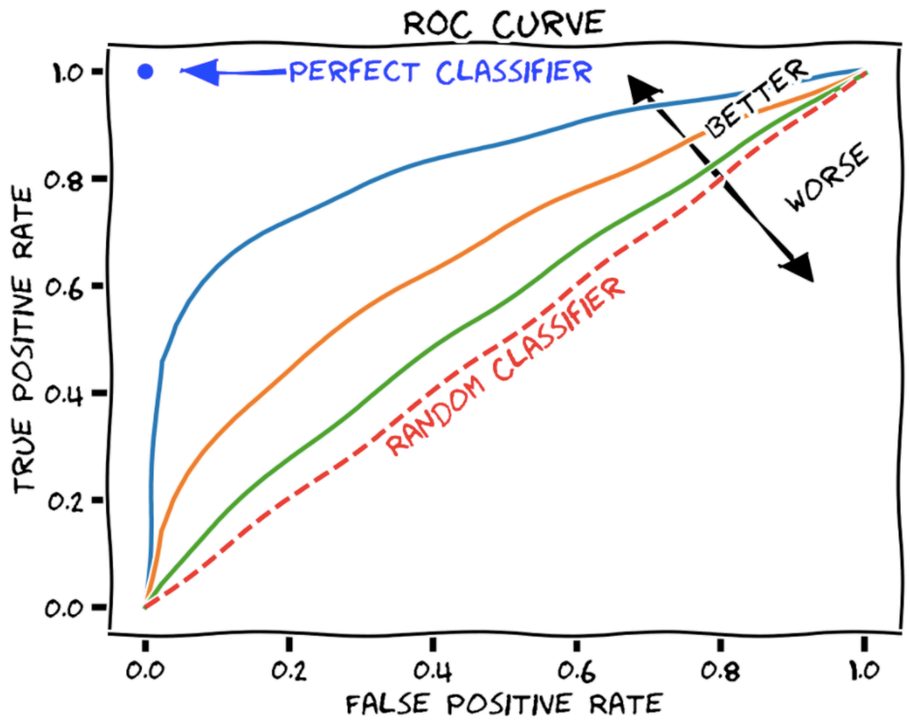
- **Recall** and **Precision** are often used as performance measures of classifiers.
- These metrics depend on the chosen cutoff.
- One can easily optimize on the number of false positives or false negatives, but not both at the same time.



ROC Curve - Area Under the Curve (AUC)

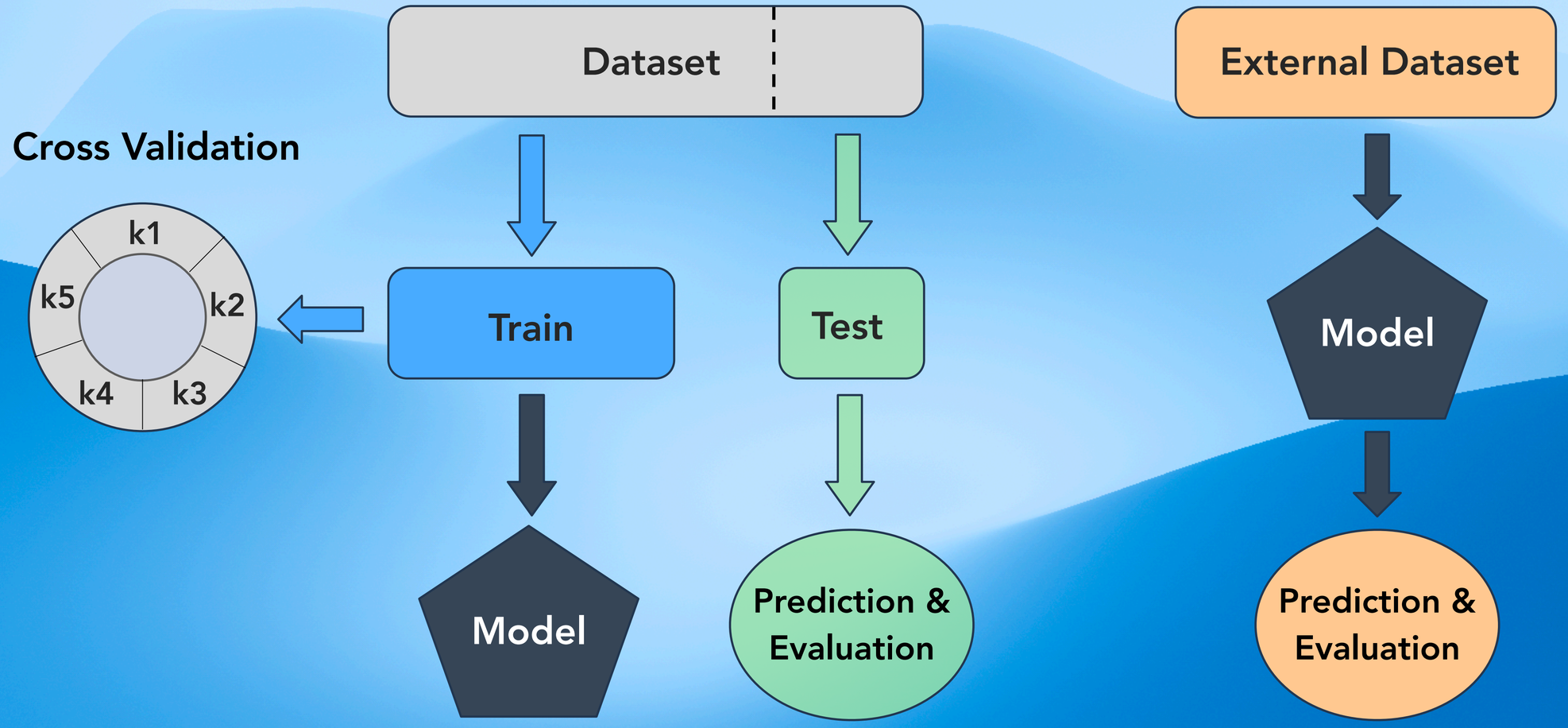
We can use the AUC to report on the goodness of the classification.

How many of the true cases have we discovered?



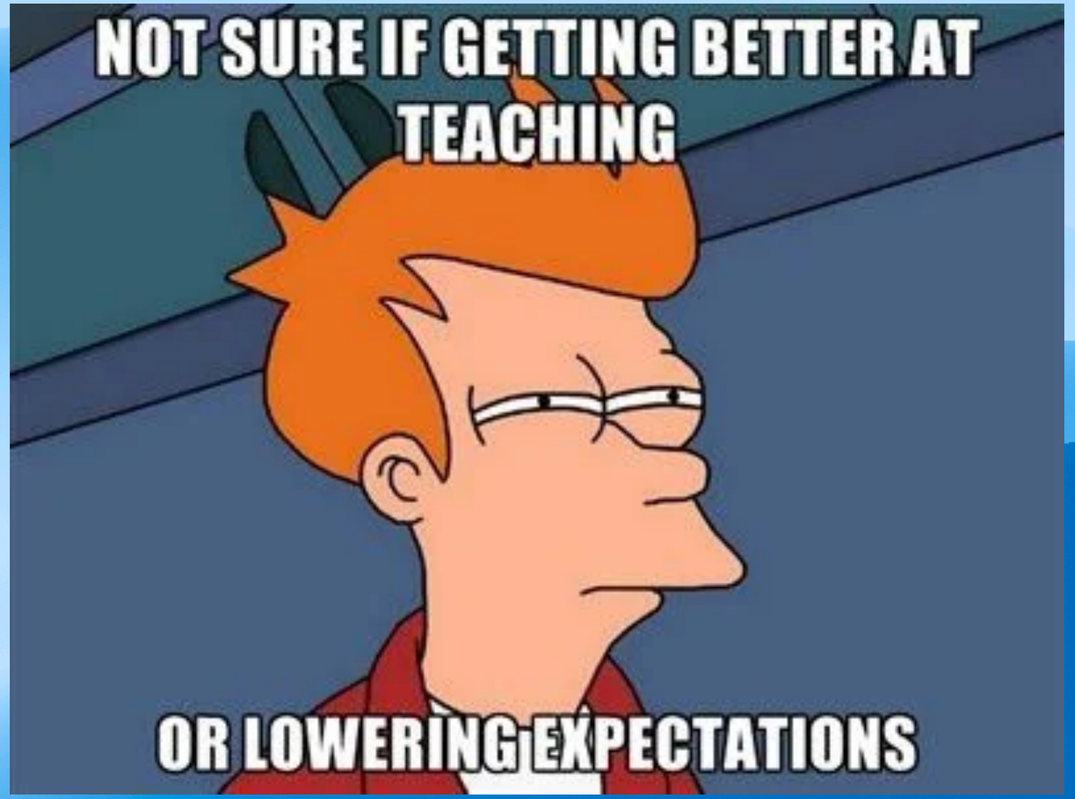
- Receiver Operator Curve (ROC)
- Plotted by using every possible cutoff.
- Each True Positive Rate has an associated False Positive rate.

Train-Test & Validation Dataset



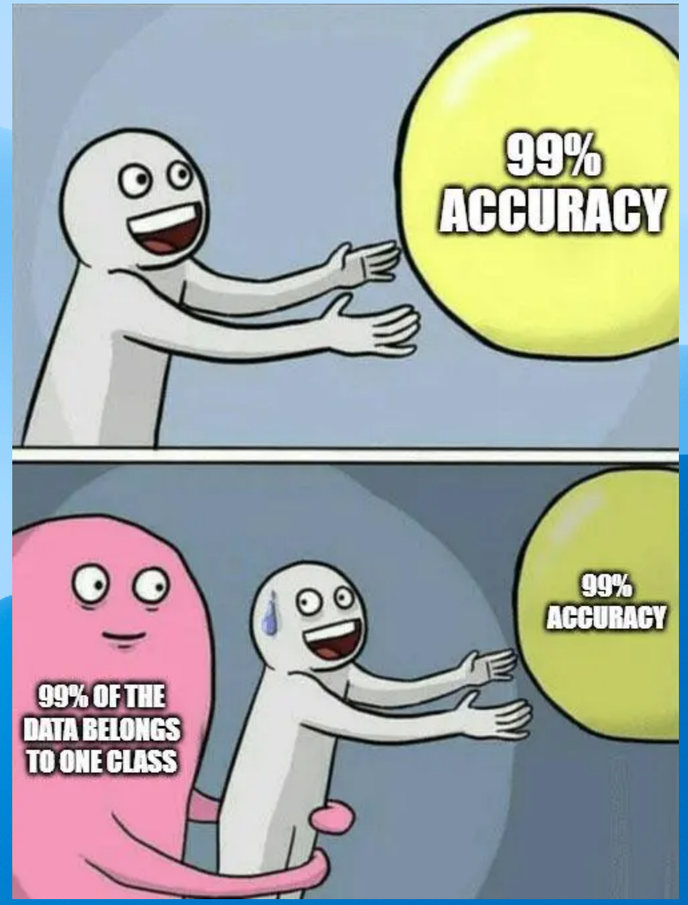
Course Evaluation

- We would love your feedback on this course:



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!