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

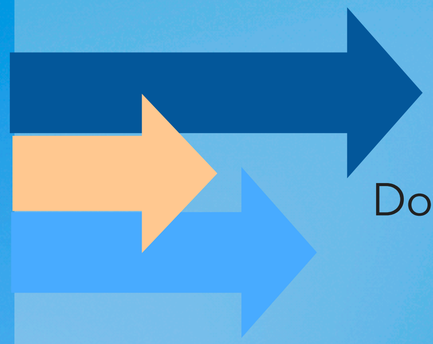
Henrike Zschach

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/

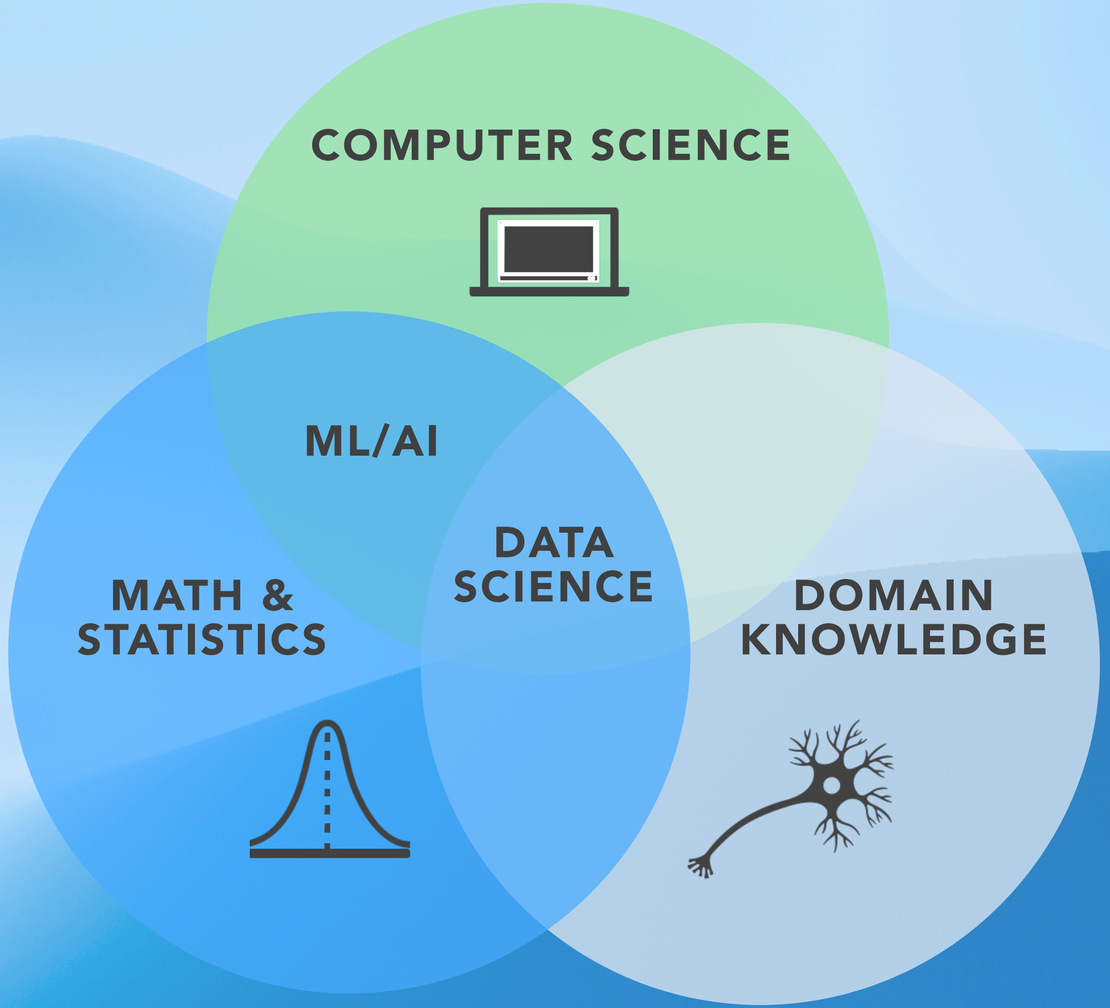
Go to course website: https://center-for-health-data-science.github.io/R4DataScience/

Program

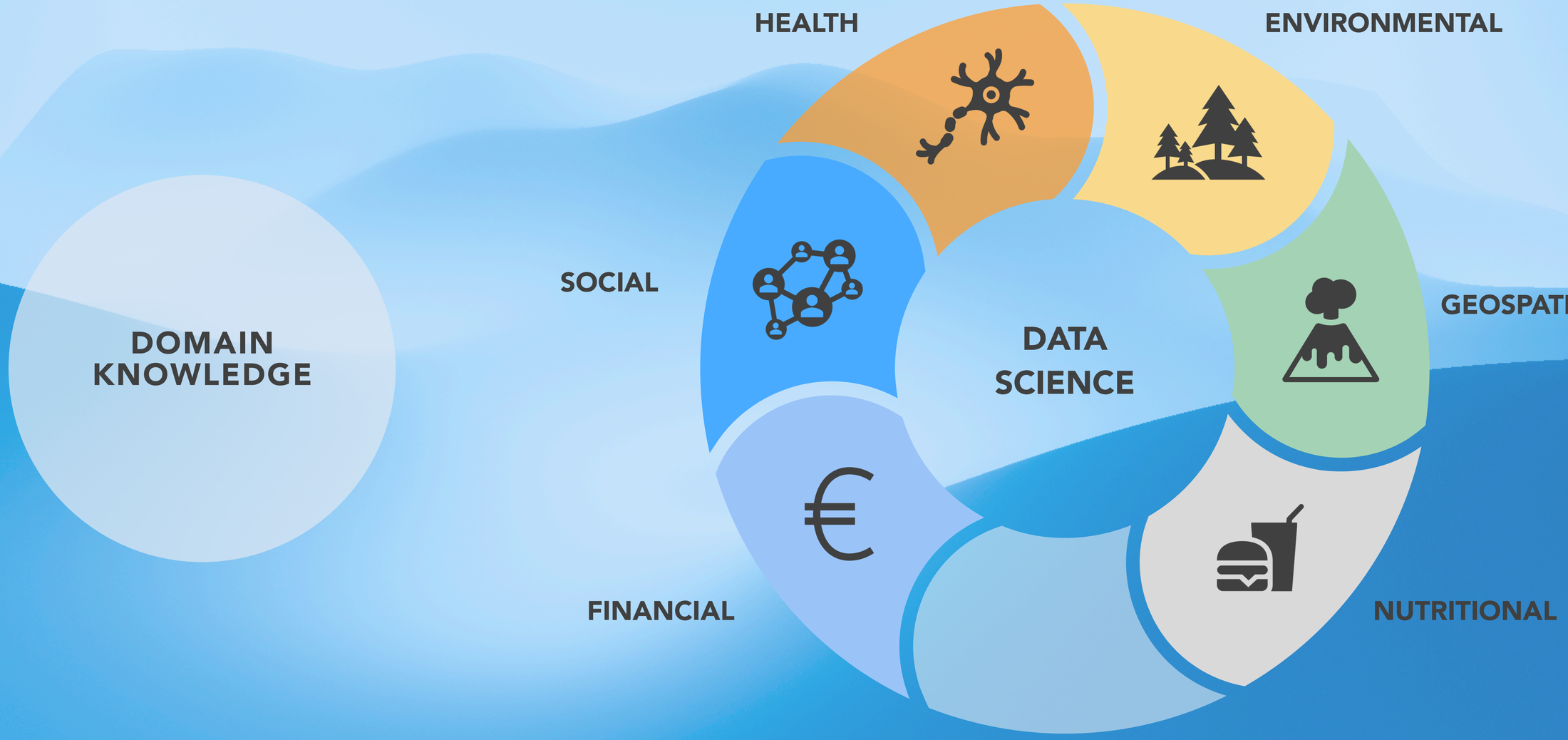
Day 1		Day 2		Day 3	
08:30-09:00	**Installation Issues + Coffee**	08:45-09:00	**Coffee**	08:45-09:00	**Coffee**
09:00-09:30	**Introduction to Course**	09:00-09:45	**P3: Exploratory Data Analysis**	09:00-09:45	**P5A: Models in R**
09:30-10:00	**P0: Intro to R and Quarto**	09:45-11:00	**E3: Exploratory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

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

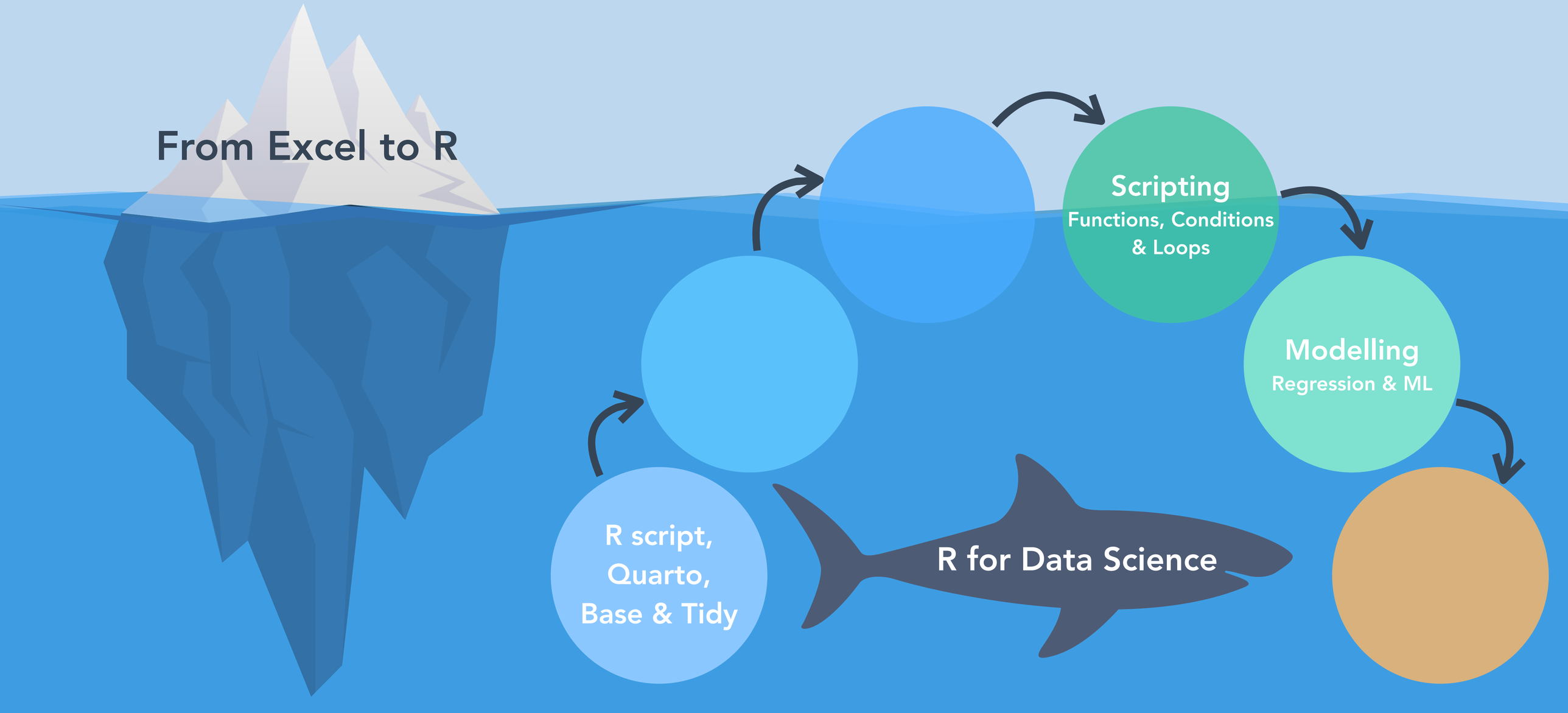
Biological (omics) Databases
Medical Image Data

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
  - No formatting
  - 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
  - **Bold**, *italic*, underlining
- 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

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

Load Packages

```r
{r}
library(tidyverse)
library(readxl)
```

Load Data

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

Inspect Data

Check dimensions of data

```r
{r}
dim(diabetes)
```

Check structure of data

```r
{r}
str(diabetes)
```

Check for NA's in each column

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

Exploratory Data Analysis

Plot distribution of BMI

```r
{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

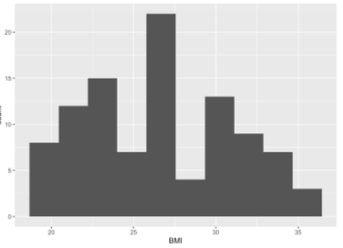
```r
colSums(is.na(diabetes))
```

```
              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)
```



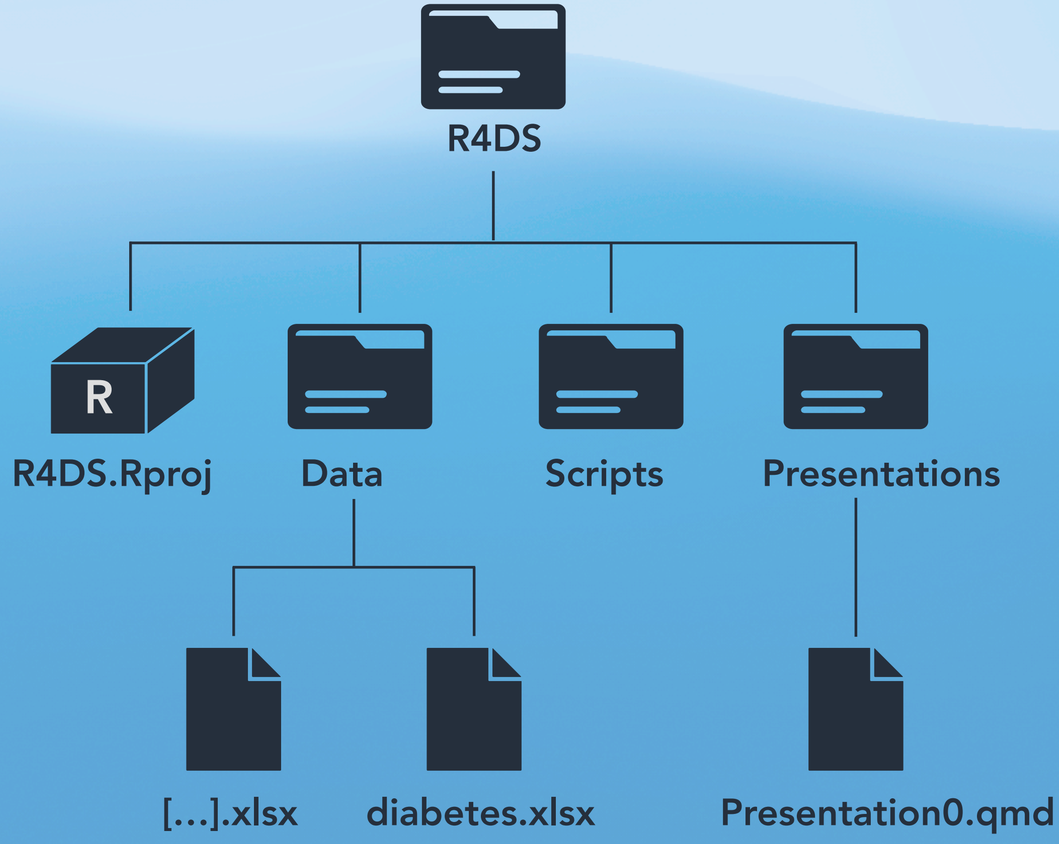
R project

R4DS.Rproj

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

.Rproj.user

- Hidden directory to store of temporary files.
  - Workspace state: open tabs, cursor positions
  - Plot-viewer history
  - Session specific information
  - (If you have well documented code you will not need this)



Exercise 0
Intro to R and Quarto

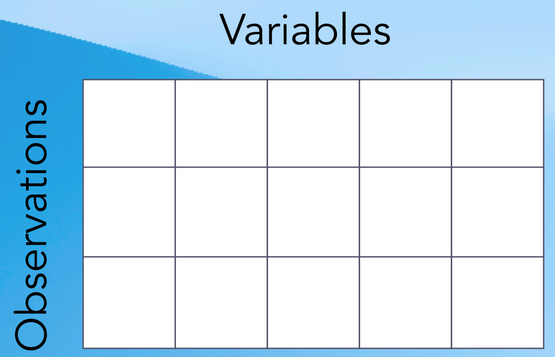
Part 1
Data Cleaning

Data Cleaning

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

Merge several datasets into one.

R Data Types & Structures



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
Age

Data Frame
Age
Height
BMI

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	`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 Cleaning

Part 2
Summary Statistics and Data Wrangling

Why EDA

Exploratory Data Analysis (EDA) is an essential step in any data analysis or modeling workflow.

- Understand **distributions**
- Spot **outliers**
- Reveal **patterns**
- Explore **relationships** between variables

EDA is about getting an overview of your data before fitting it to any models.

There are three broad ways to investigate variables:

Univariate

Examining one variable at a time.

Bivariate

Exploring relationships between two variables.

Multivariate

Looking at relationships among multiple variables.

Why EDA

Univariate Analysis

Histograms – Distributions of numeric variables

Boxplots – Detect outliers and skewness

B**ar charts** – Explore the frequency of categorical variables

Summary stats – Mean, median, standard deviation (SD), min, max, interquartile range (IQR)

Bivariate Analysis

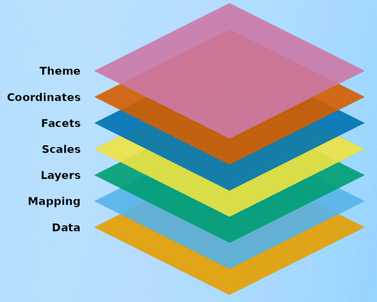
Scatter plots – Explore relationships between two numeric variables

Correlation coefficients – Assess linear strength (e.g., Pearson's r)

Contingency tables – Relationships between categorical variables

Line graphs – Track trends over time

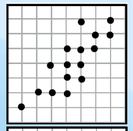
ggplot2 Recap

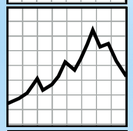


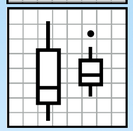

```
df %>%
  ggplot(aes(x = var1, y = var2, color = var3))  +
  geom_XXX() +
  scale_color_manual(values = c("A", "B")) +
  labs(
    title = "title",
    x = "x axis title",
    y = "y axis title",
    color = "legend title"
  ) +
  theme_XXX()
```

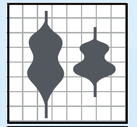
Aesthetics (aes)

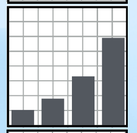
color, fill, shape, group, linetype


geom_point()

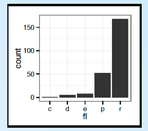

geom_line()

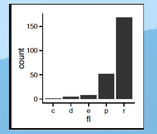

geom_boxplot()

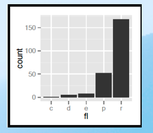

geom_violin()

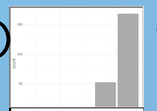

geom_bar()

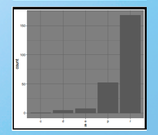

geom_hist()


theme_grey()
# default


theme_bw()


theme_classic()

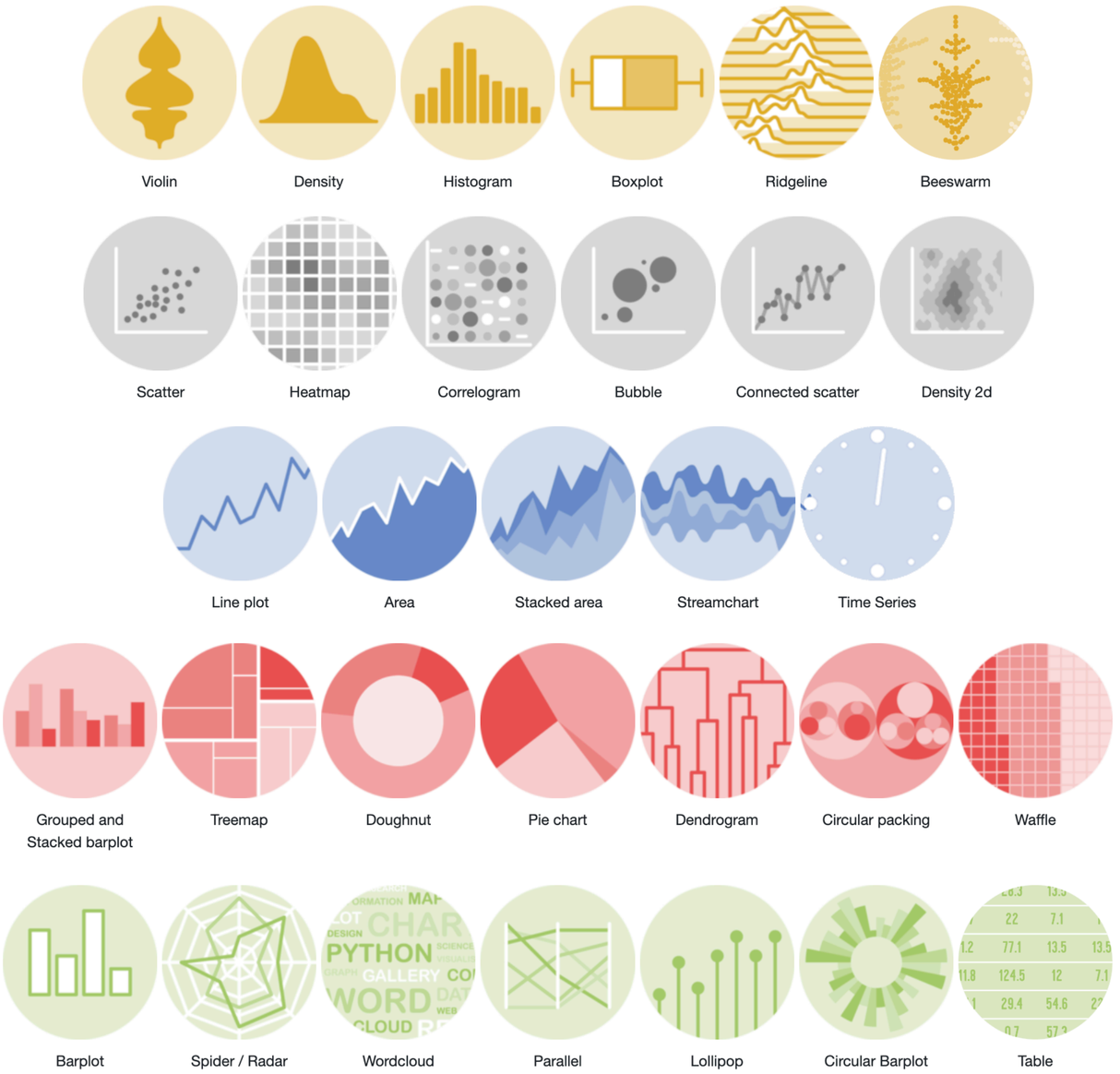

theme_light()


theme_dark()


theme_minimal()

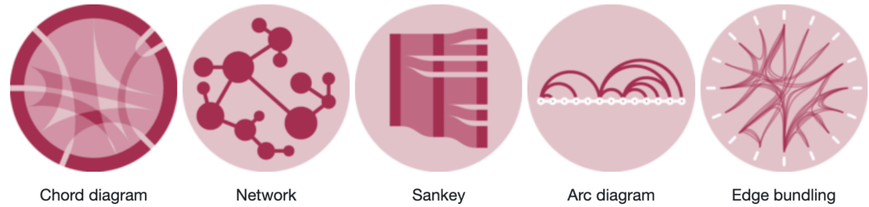
Cool New Plot?

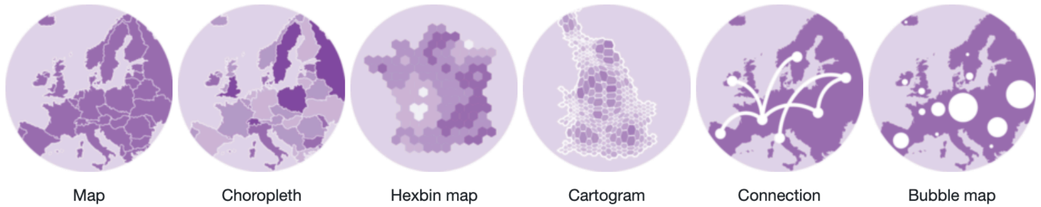
https://r-graph-gallery.com/

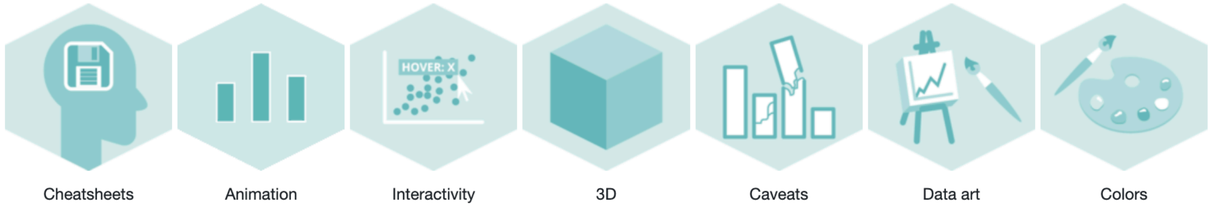


Cool New Plot?

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







Wide and long format

A key idea in ggplot - each piece of information must have its own column:

- One column for the x-axis
- One for the y-axis
- One for color, shape, size, etc.

ID	Stage	Grade	Gene 1	Gene 2
Sample 1	I	Grade 1	30	27
Sample 2	IV	Grade 3	14	42
Sample 3	III	Grade 1	25	23
..	..	..	..	..

ID	Variable	Value
Sample 1	Stage	I
Sample 1	Grade	Grade 1
Sample 1	Gene 1	30
Sample 1	Gene 2	27
Sample 2	Stage	IV
Sample 2	Grade	Grade 3
Sample 2	Gene 1	14
Sample 2	Site D	42
Sample 3	Stage	III
Sample 3	Grade	Grade 1
Sample 3	Gene 1	25
Sample 3	Gene 2	23
…	…	..

This is easy to browse (like spreadsheets), but **inconvenient** - you'd have to make a separate plot for each gene!

Outliers

Data points that differ markedly from the rest.

An outlier may be a natural (extreme) variation … or it may arise from measurement/entry/anomalies errors in:

- Data collection
- Lab experiment
- Database/Registry Entry

An outlier is **a sample** or **a single datapoint**

Skew summary statistics

Distort visualizations

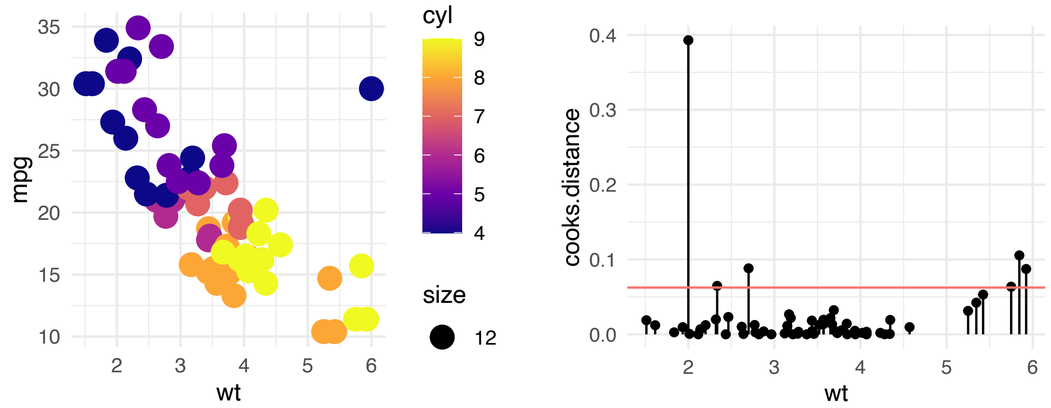
Affect model performance

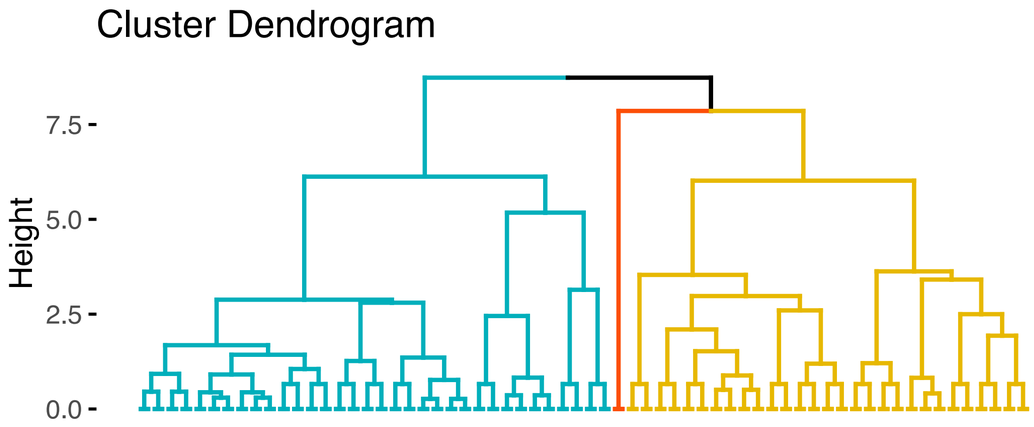
Outliers - Visual Inspection & Stats Test

- **Visualization:**
  - Distributional plots
  - Clustering Dendrogram
  - 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





Outliers

If it doesn’t make sense, you can:

- Cap them at a reasonable threshold
- Transform them (e.g., log transformation).
- Remove

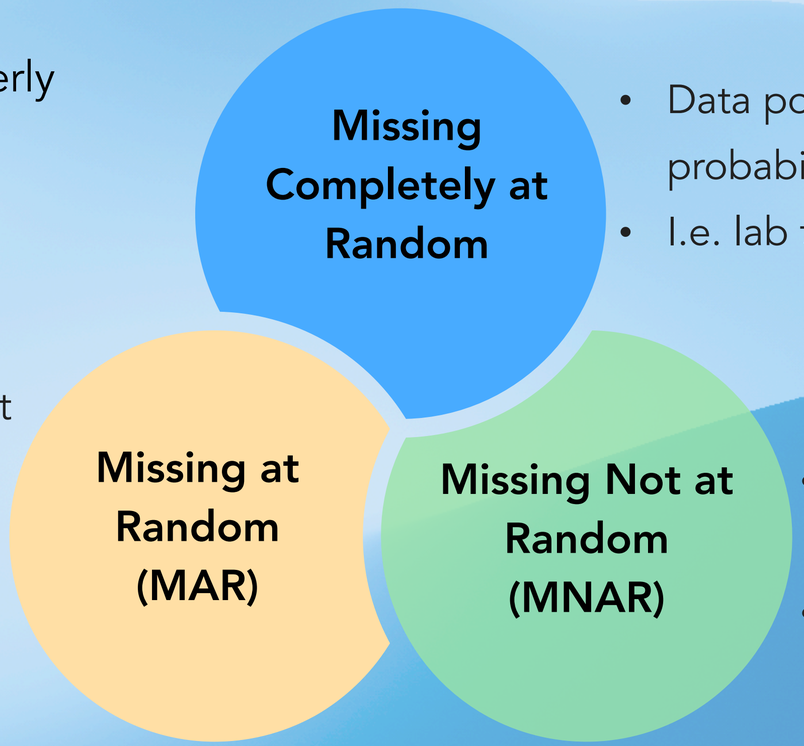
Be cautious when removing data:

- Outlier removal should always be guided by domain knowledge and clear justification.
- Removing too much — or for the wrong reasons — can distort your analysis.

Missing Values

Missing data is common in most real-world datasets and can significantly affect the quality of our analysis.

It's essential to identify, understand, and properly handle missing values to avoid biased or misleading conclusions.



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

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

- Depends on the missing value itself
- I.e. patient with depression skip mental health surveys

Missing Values

- **Remove** rows with missing values — good when missingness is MCAR (completely random) and affects only a small portion of the data.

- **Replace** missing values
  - *Numerical columns: Replace with mean, median, or mode.*
  - *Categorical columns: Use the most frequent category or introduce a new "Unknown" category.*

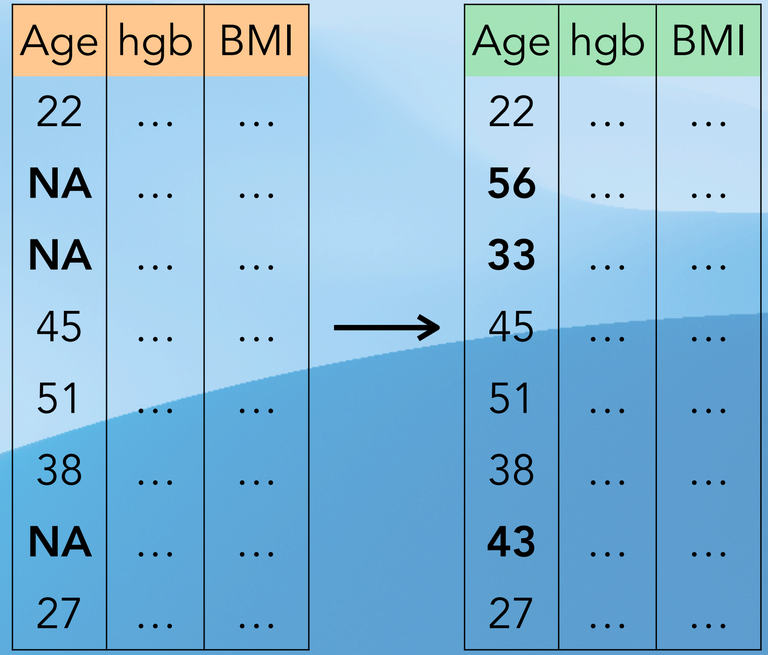
- ***Impute** missing values (Regression, Maximum Likelihood, ML)*

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)



Exercise 2
Summary Statistics and Data Wrangling

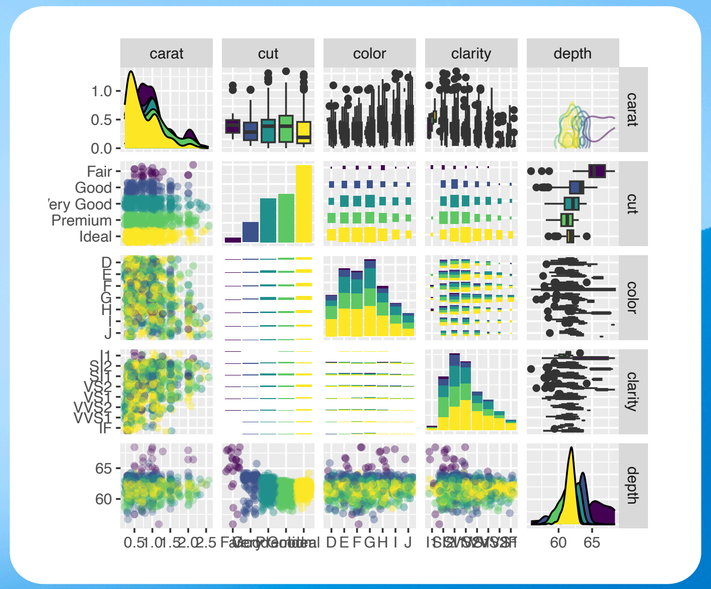
Part 3
Exploratory Data Analysis: Multivariate Analysis

Why Multivariate EDA

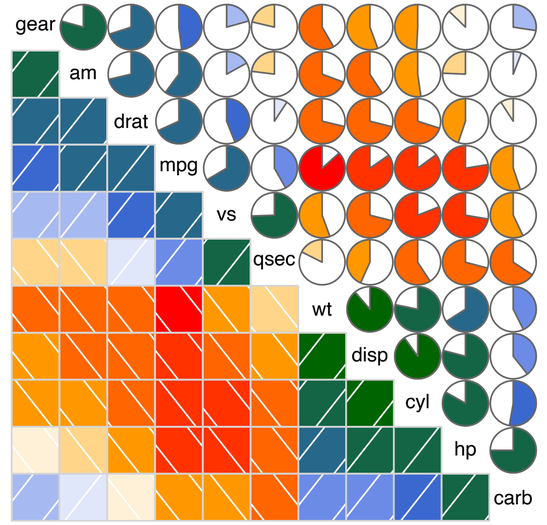
- **The Challenge of Complexity (Reality of High-Dimensional Data)**

- As the number of variables increases, especially if you have dozens of gene expression values, categorical groups, and clinical variables - simple visualizations become too noisy, overwhelming and hard to interpret to interpret.

Scatter plots crowded



Correlation plots dense and overwhelming

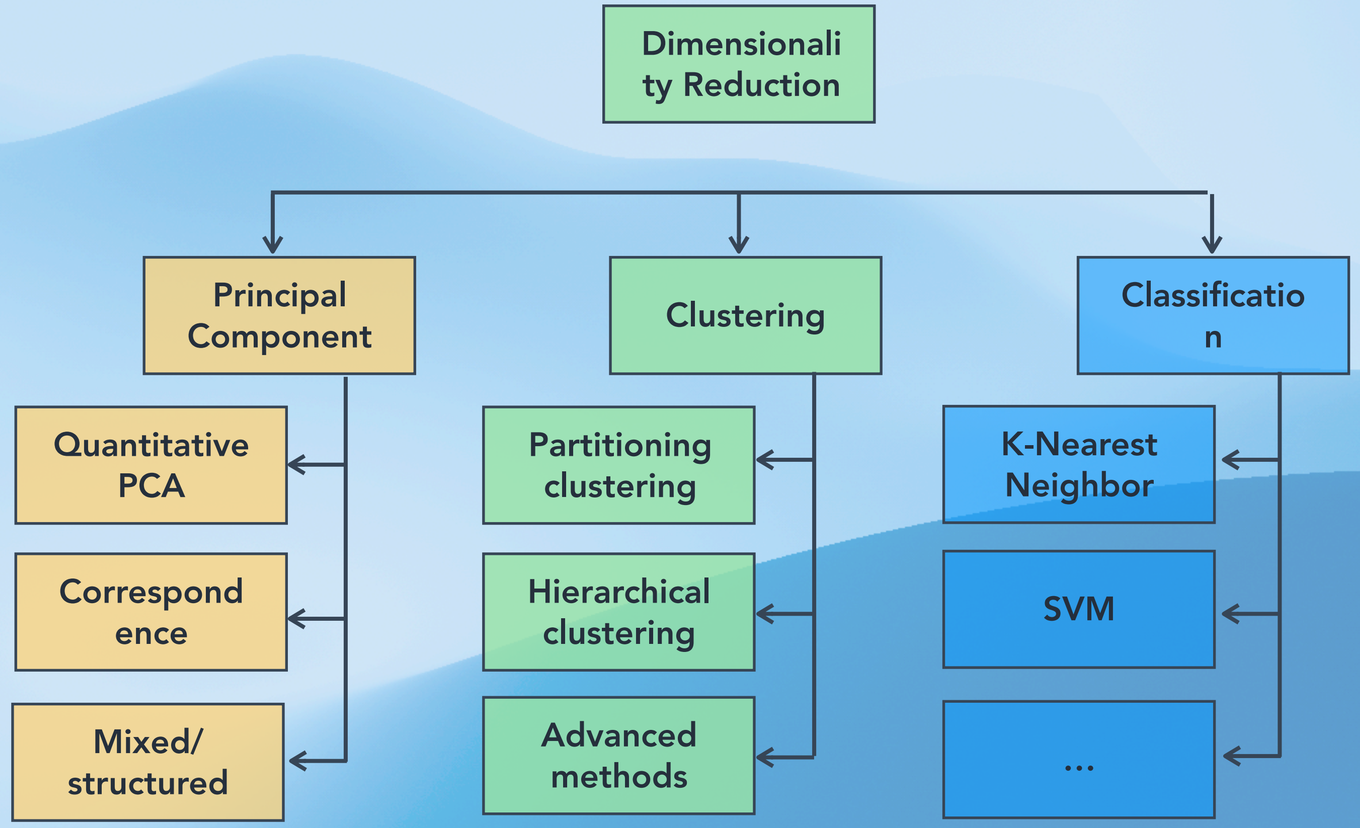


Dimensionality Reduction

We need to simplify:

Multivariate analysis can help us uncover trends, clusters, and interactions.

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



Data Transformation

Real-world data is often messy:

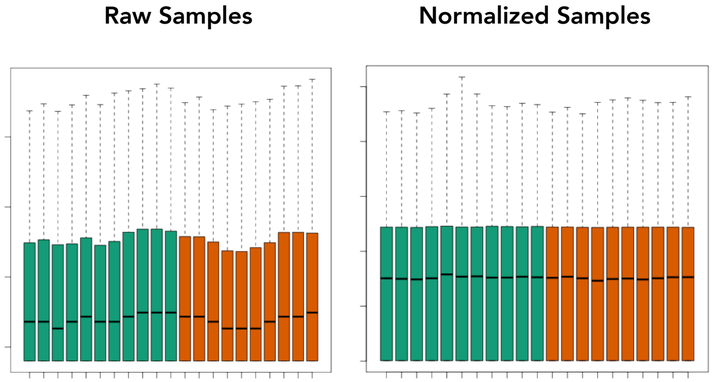
- Skewed
- Non-linear
- Heteroscedastic (i.e., variance changes with the mean).

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

- Scale
- Transform
- Stabilize variance

To meet model assumptions (e.g., normality, homoscedasticity) and perform accurate analysis and modeling.

Normalization



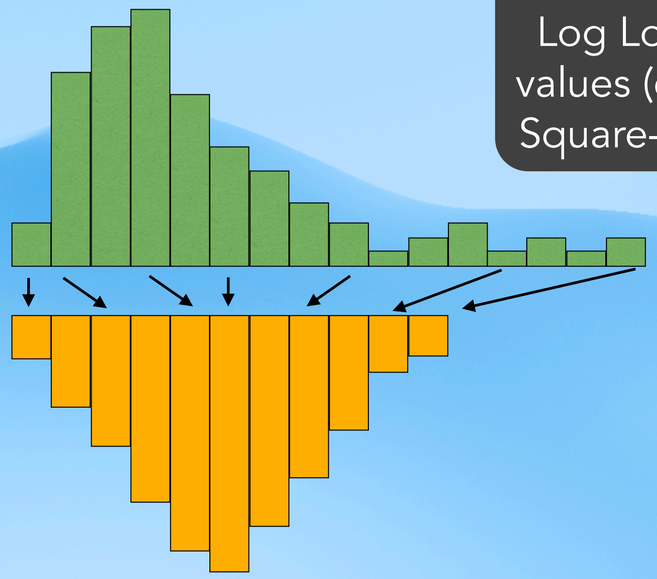
- Can use for not normally distributed data.
- Variables do not get zero centered.
- Normalization within a range (max, min).
- Affected by outliers.

Data Transformation

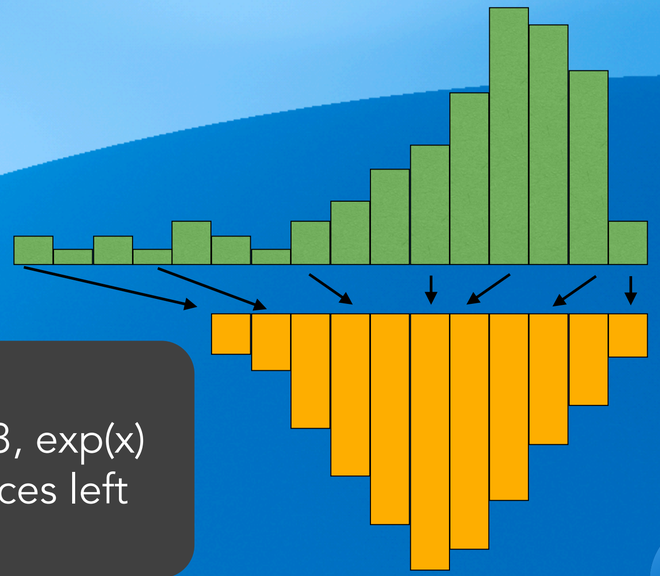
In many datasets, especially in biological or economic data, variables are skewed rather than normally distributed.

If predictors are highly skewed, models are:

- Sensitive to extreme outliers
- Influential points can distort results



Right-skewed
Log Log2 Log10 - Compresses large values (count expression, income data)
Square-root - sqrt(x) - Gentler than log



Left-skewed
Squaring or cubing x^2, x^3, exp(x)
Expands large values, reduces left skew.

Data Transformation

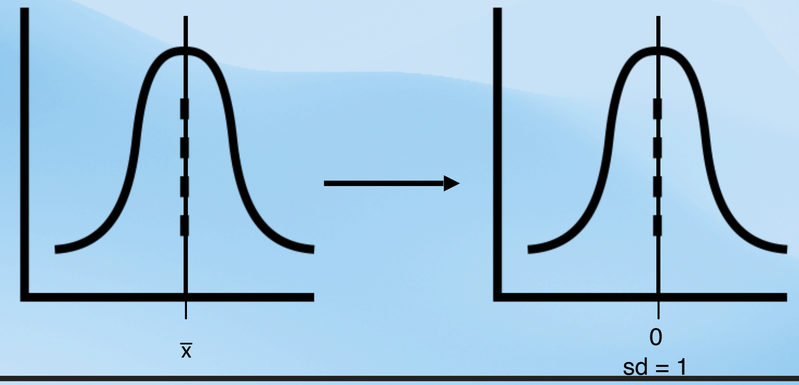
Standardization is essential when:

- Variables have different units or ranges
- If you plan to apply PCA, clustering, or distance-based models

Not constrained to a range.

Not affected by outliers.

Standardization



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

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

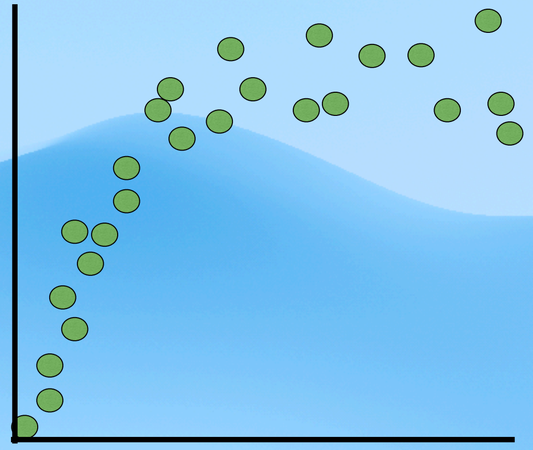
Variance Stabilization

Many statistical and machine learning methods assume **homoscedasticity**, meaning that the variance is constant across all levels of the data.

However, this is rarely the case in real-world datasets. In biological and count-based data (like gene expression), we often encounter **heteroscedasticity.**

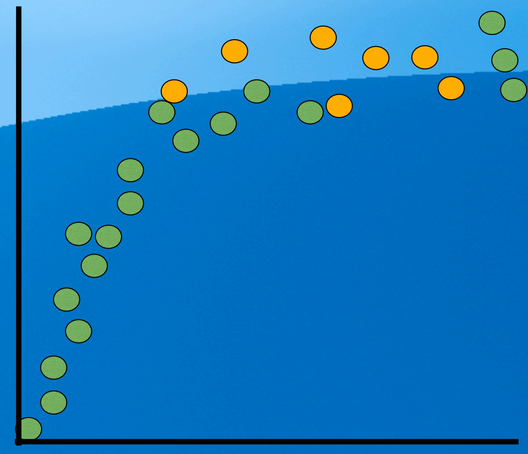
This can distort analyses and lead to misleading conclusions. For example:

- PCA and clustering can be dominated by high-variance features, obscuring biological signals.



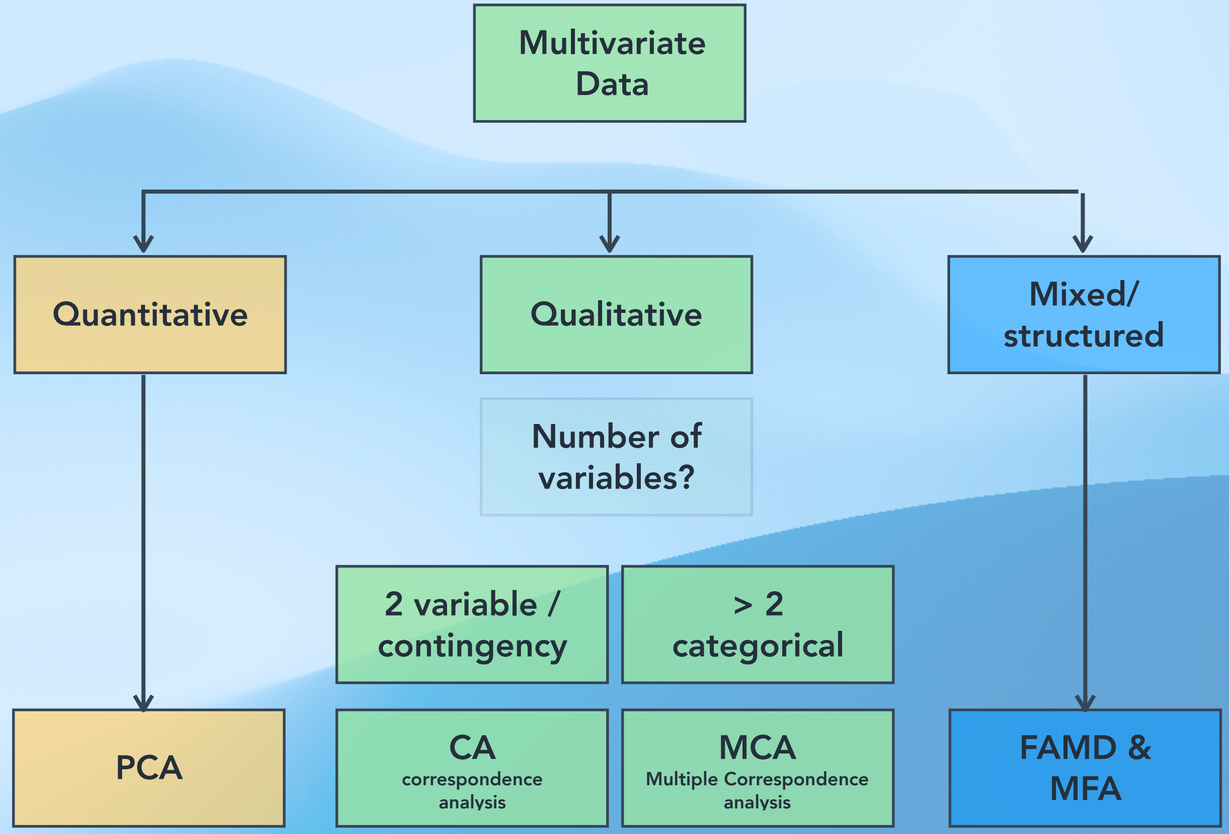
Heteroscedasticity

Homoscedasticity



Principal Component Methods

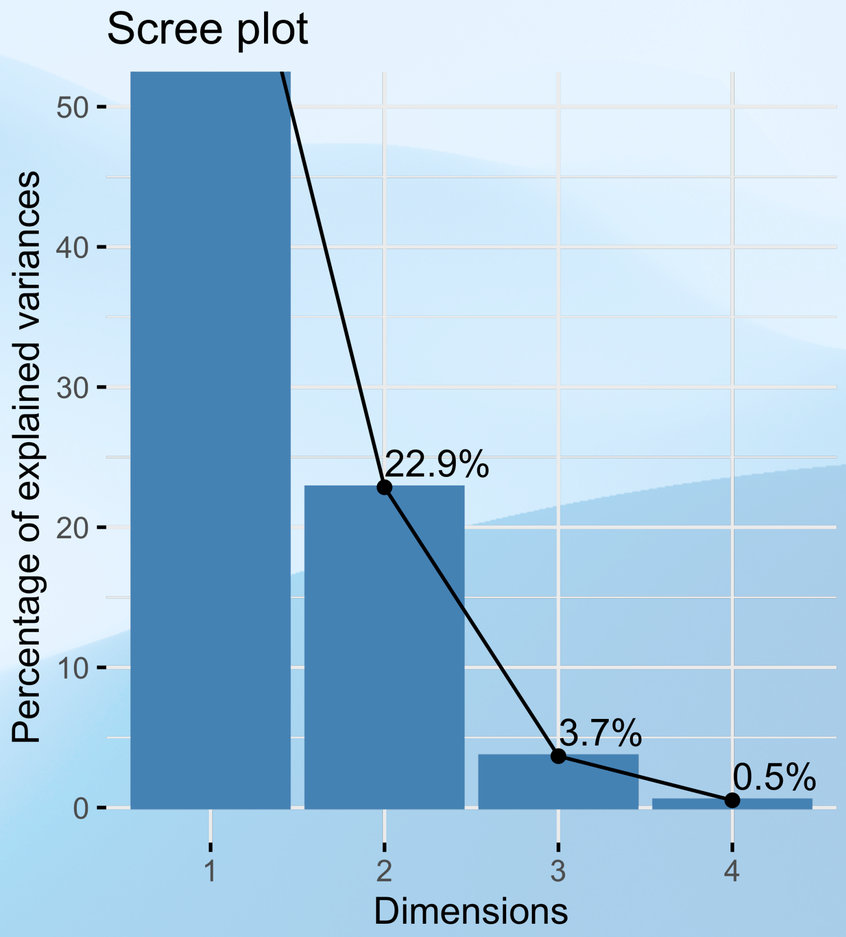
- Helps summarize, simplify, and visualize large datasets.
- Classical methods
  - PCA, CA and MCA
  - (continuous variables, contingency table and qualitative variables, respectively).
- Advanced methods -
  - FAMD, MFA and HMFA -
  - a mix of variables (qualitatives & quantitatives)



Principal Component Analysis (PCA)

- Linear dimensionality reduction:
- High → Low dimensional space
- Transforms the data into a smaller set of new variables (PC1, PC2..), that capture most of the original variability in the dataset.

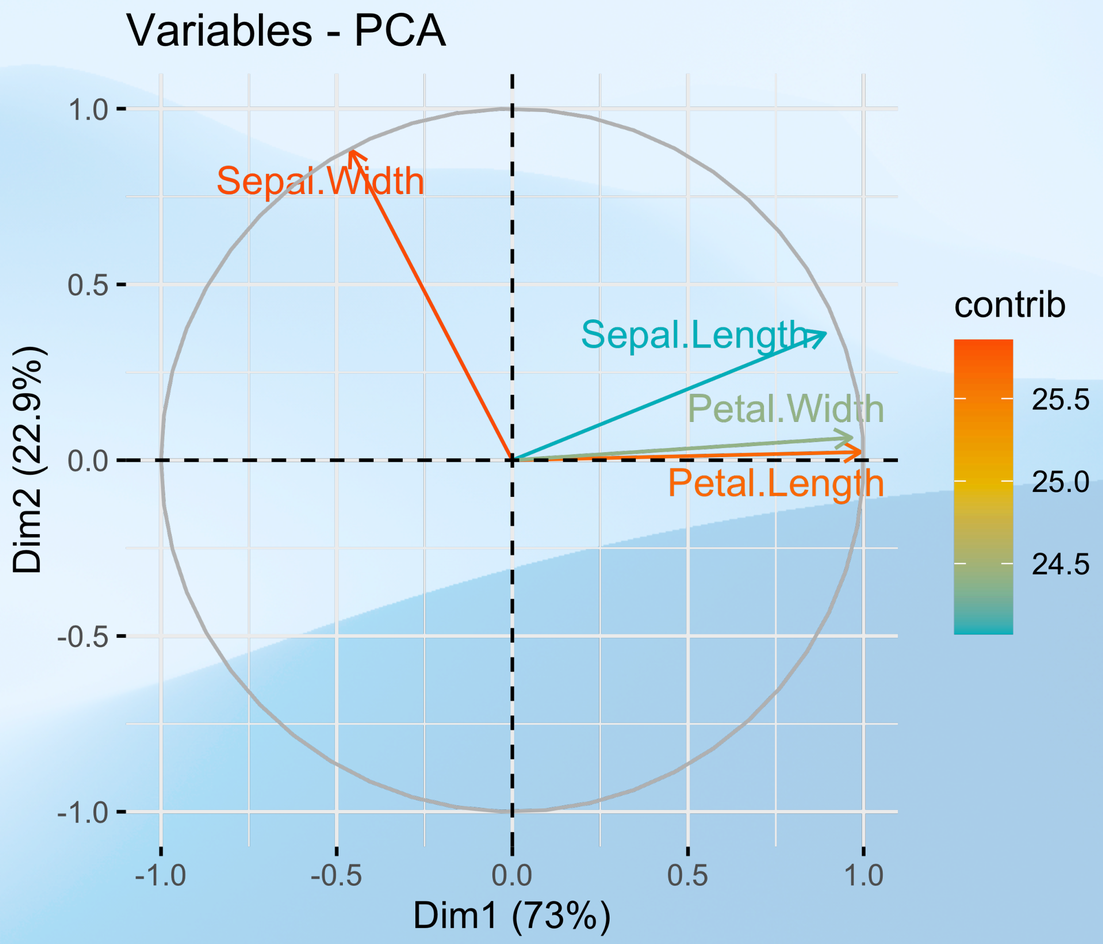
	Alc.	Alkalini	Flabon	Color	…
Indiv1	1.51	-1.17	1.03	0.25	…
Indiv2	0.25	-2.48	0.73	-0.29	…
Indiv3	1.50	-0.96	0.97	0.57	…
…	…	…	…	…	…



PCA plot with loadings

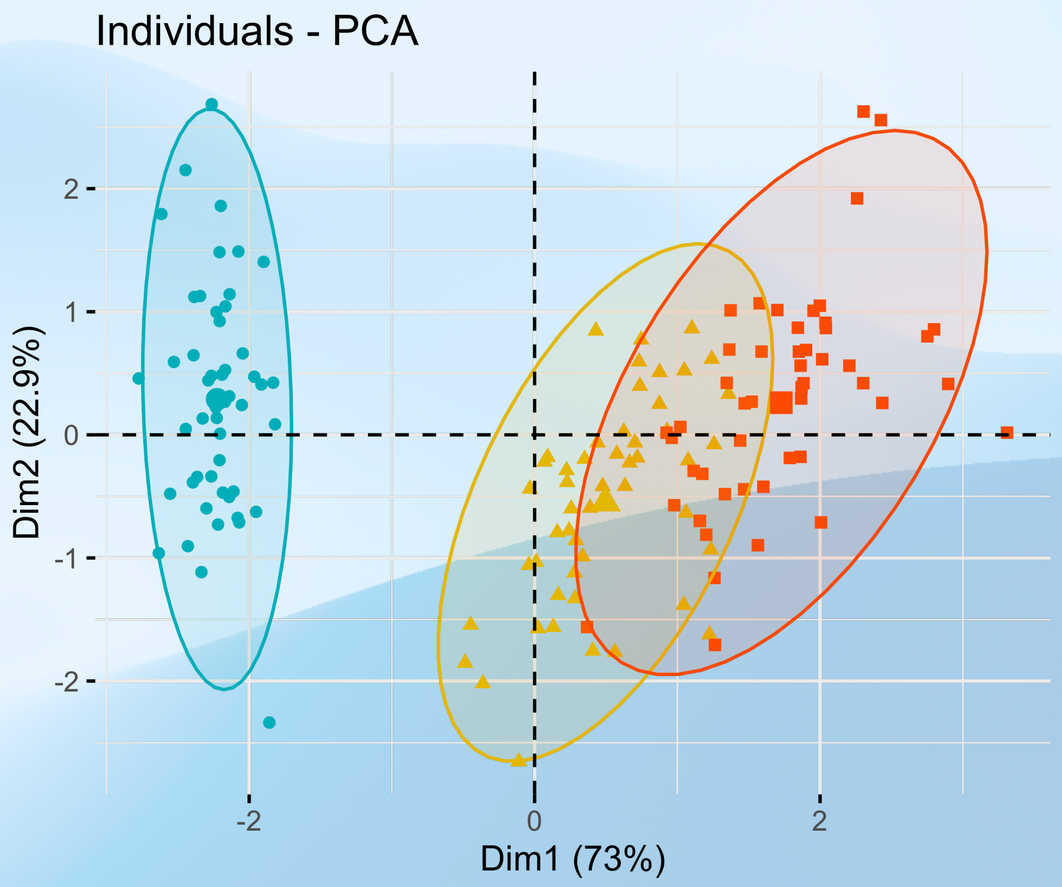
- Loadings: Arrows represent the direction and magnitude of each variable's contribution to the PC's.
- The length of the arrow indicates the strength of the variable's contribution.
- The direction of the arrow shows how the variables correlates with each PC.

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



PCA plot with loadings

- This, helps us to identify the most important features for building models.



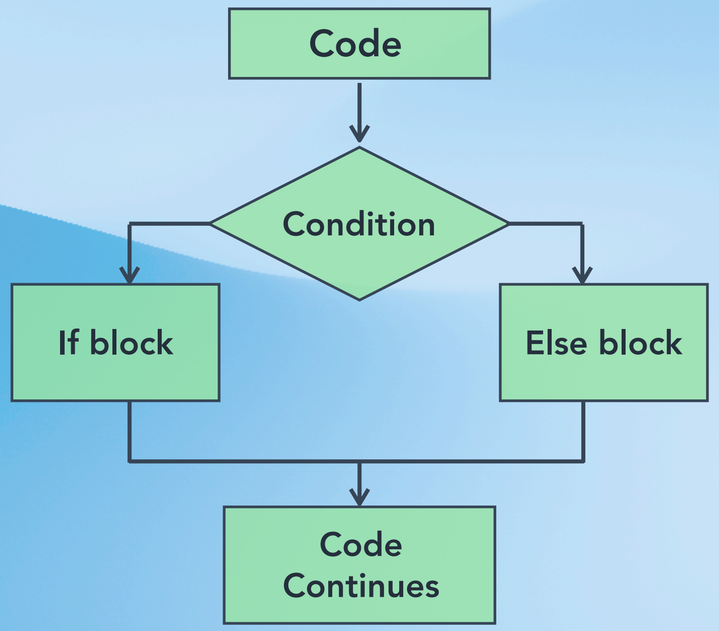
Good and simple guide to PCA in R: https://cran.r-project.org/web/packages/ggfortify/vignettes/plot_pca.html

Exercise 3
Exploratory Data Analysis

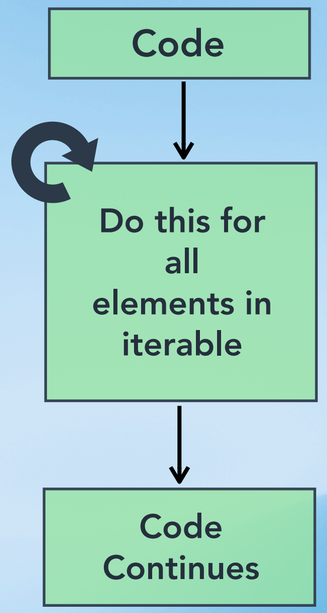
Part 4A
Conditions and For-loops

Scripting in R

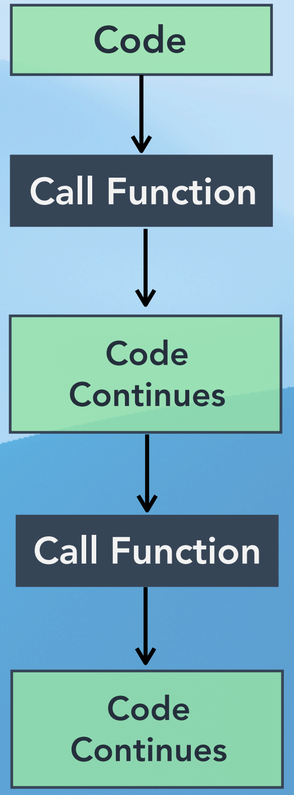
Conditionals



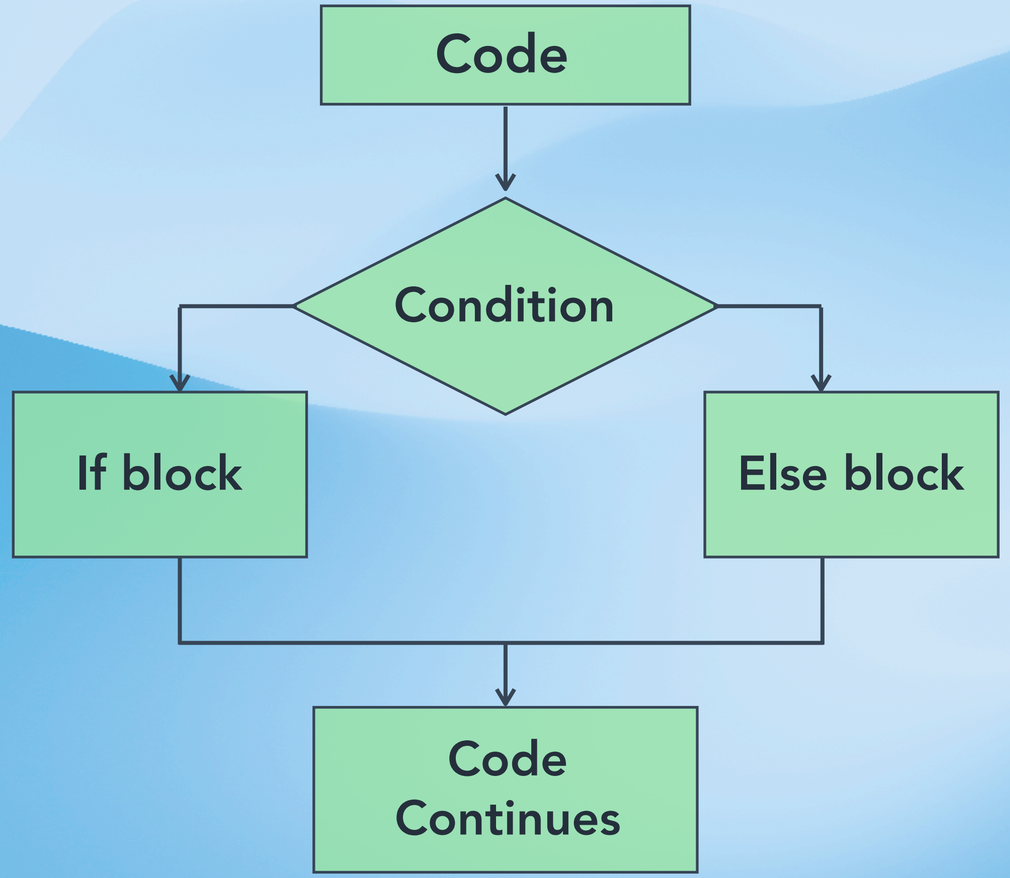
Loop



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

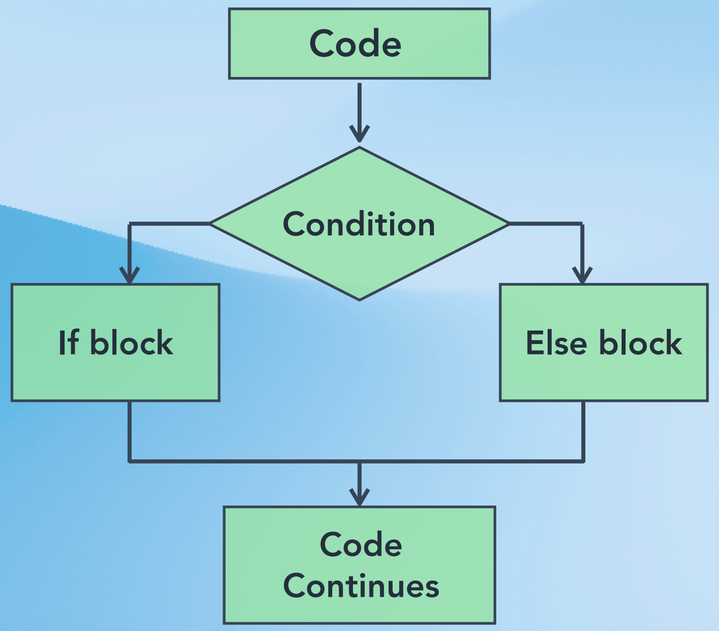
```
var1 > var2
var1 < var2
var1 == var2
var1 != var2 # Not equal to
var1 - var2 == var3
is.numeric(var1), is.character(var1), …
```

Combine conditions

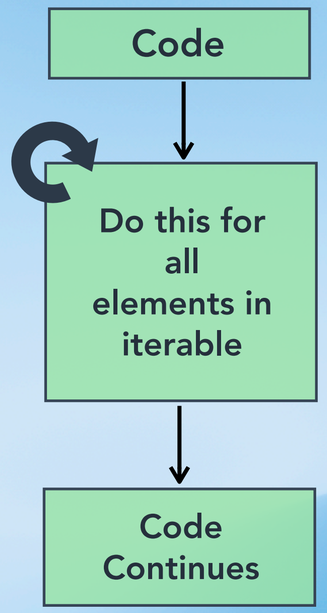
```
var1 > var2 & var1 < var3
var1 == var2 | var1 == var3
(var1 > var2 & var1 < var3) | (var1 == var2)
```

Scripting in R

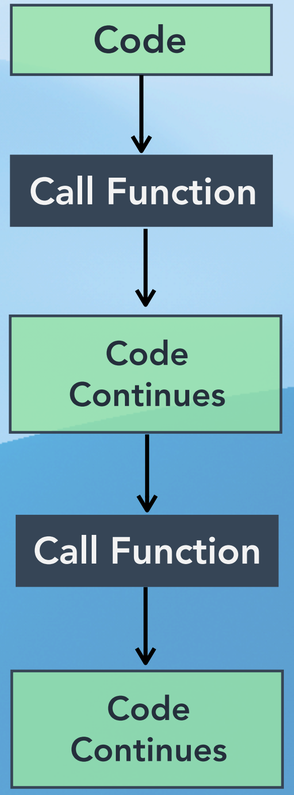
Conditionals



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

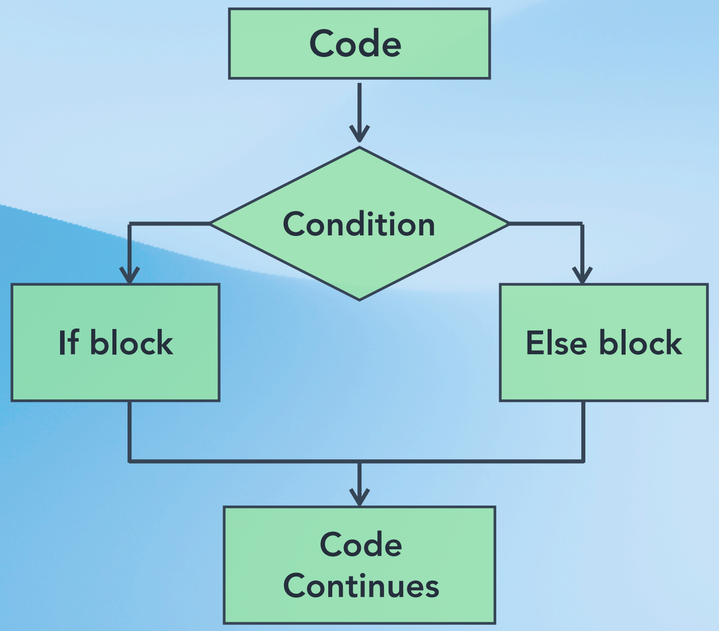
- Looping in R *can* be slow if data is large!
- Alternative is the family of **apply functions:** apply a function to every; value, list element or row/column.
- These functions are written in  code - fast!

Apply Functions

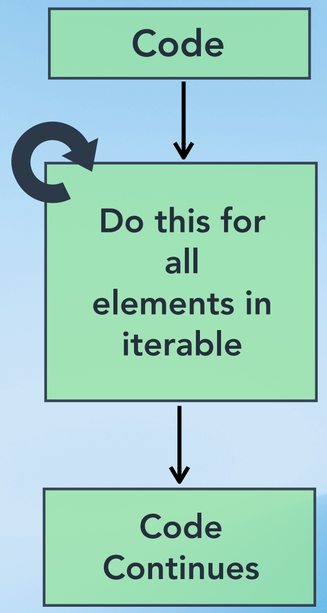
```
# Input is a matrix, specify rows or cols. Output is a list:
apply(df, 1, FUN = mean)

# 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

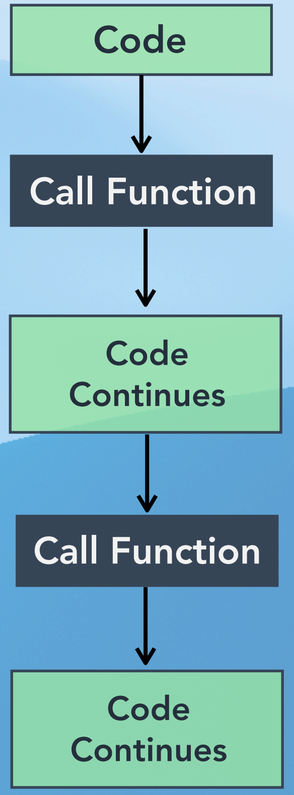
Conditionals



Loop



Functions



Exercise 4A
Conditions and For-loops

Part 4B
Functions

Functions

Pros:

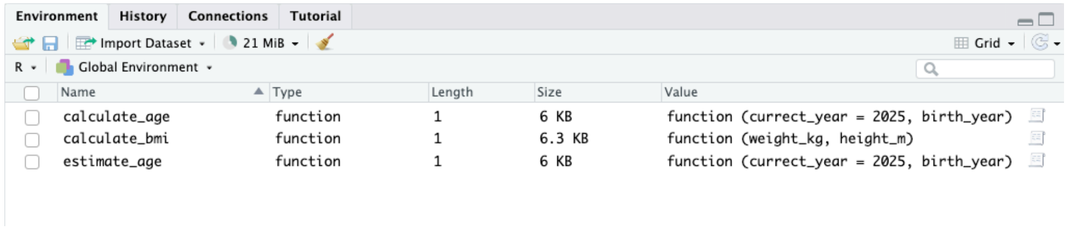
- Reusable: Write once, use many times.
- Readable: Code is clean and easy to understand.
- Modular: Breaks down complex tasks into steps.

Code Example:

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

Globale variables

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



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

````
---
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_segment.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)
```

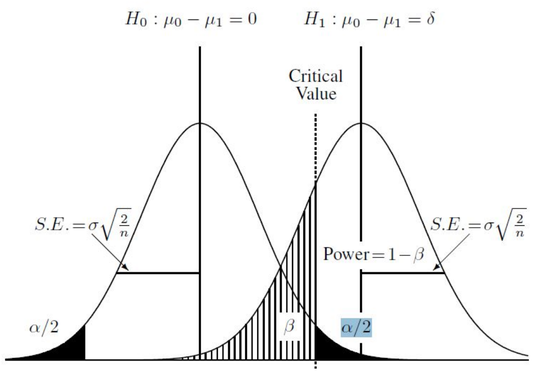
```
Warning in calculate_bmi_2(weight_kg = 25, height_m = 1.8) :
  The calculated BMI is outside the normal range. Please check your input values.
[1] 7.716049
```

Exercise 4B
Functions

Part 5A
Models in R

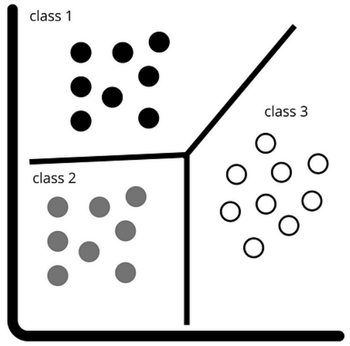
Types of models

Hypothesis Testing



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

Prediction / Classification

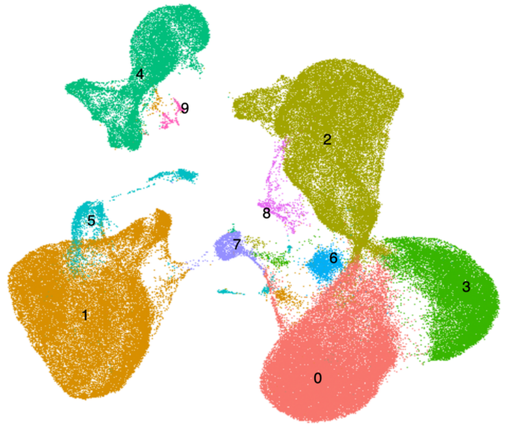


Classification Model

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 vs normal samples?
- Is there a difference in median height between men and women?

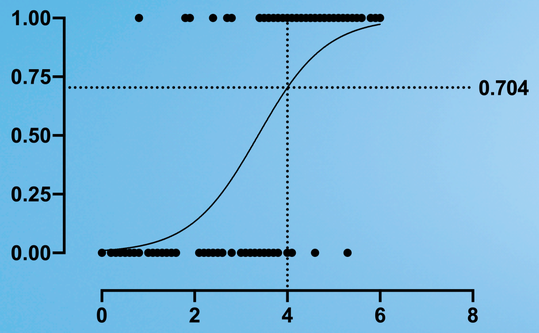
Difference tests:

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

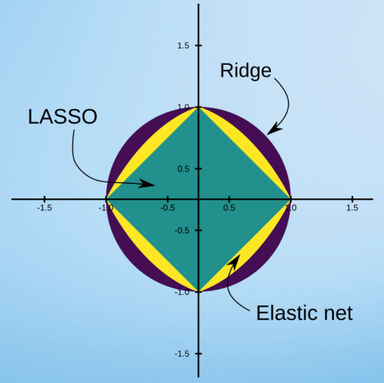
Predictive Modelling & Classification

- 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)

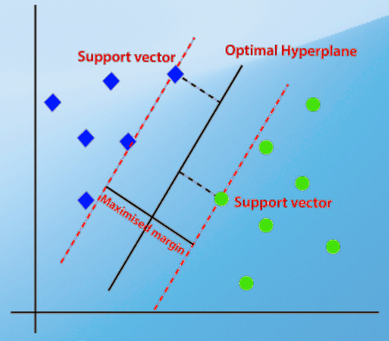
Logistic Regression



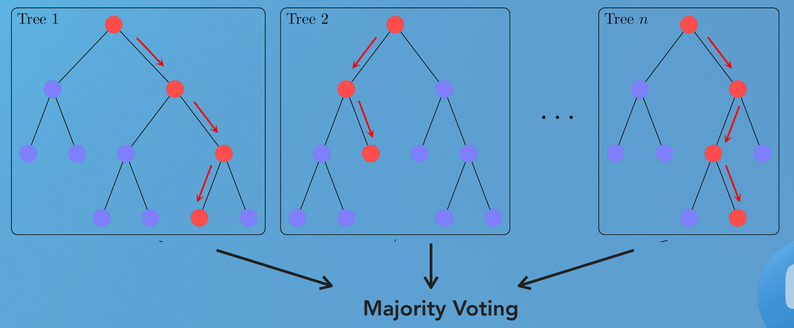
Penalized Regression



Support Vector Machine

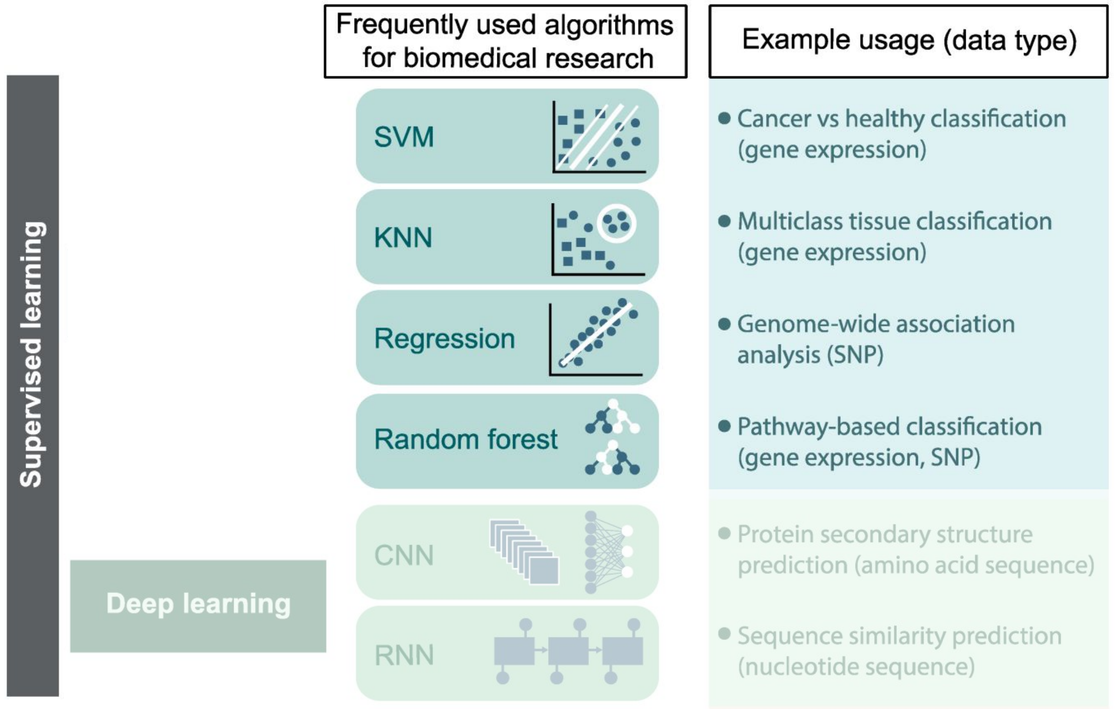


Random Forest



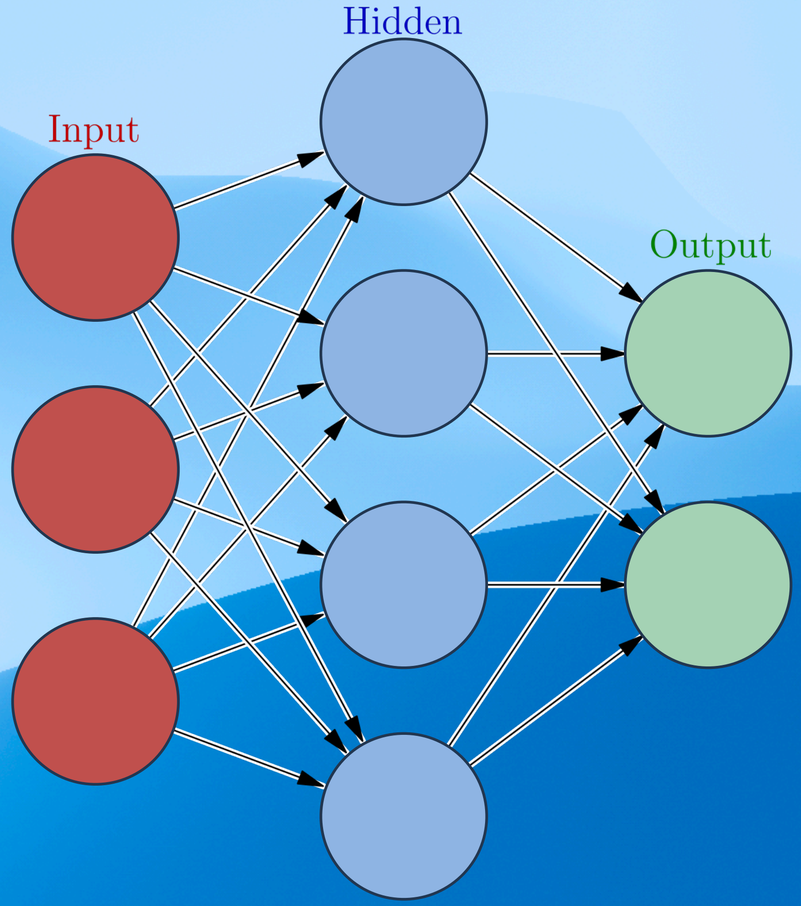
Supervised Learning / Modeling

- **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



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.

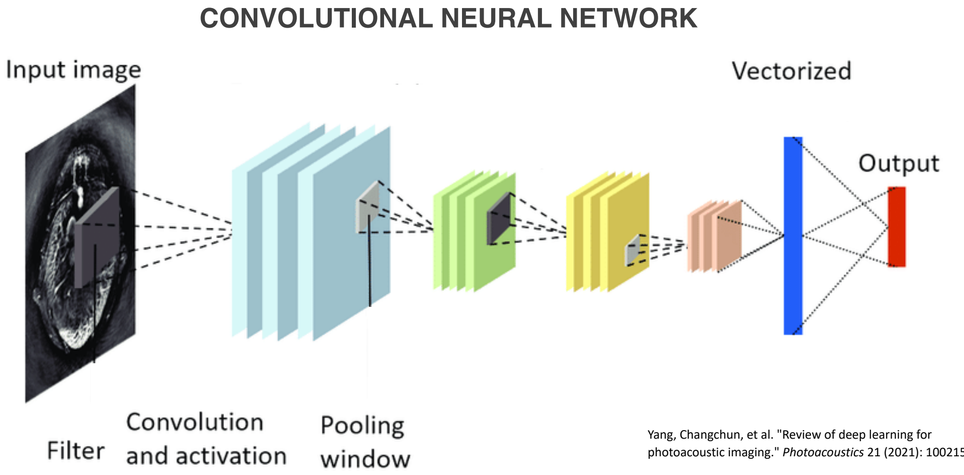


Medical Image Analysis

- 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:

ChatGPT

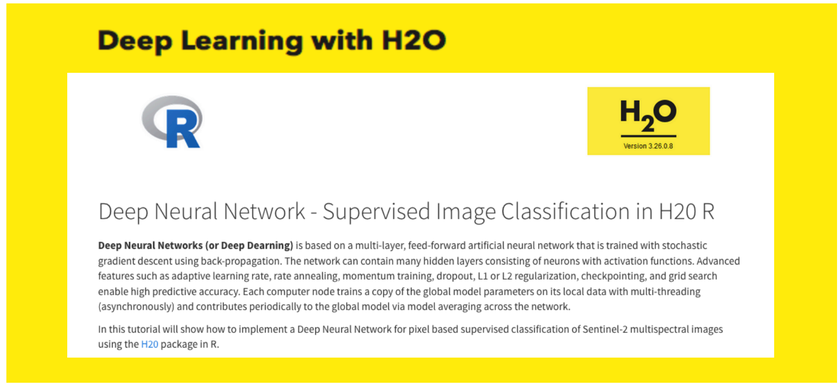
- Many Data Scientists do NOT implement their own NNs, but they use a trained one.

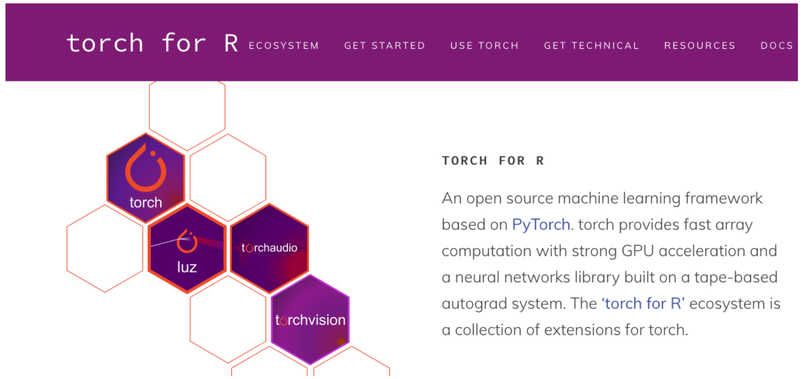


Yang, Changchun, et al. "Review of deep learning for photoacoustic imaging." *Photoacoustics* 21 (2021): 100215.

Neural Networks in R? Perhaps not…

Deep Learning in R

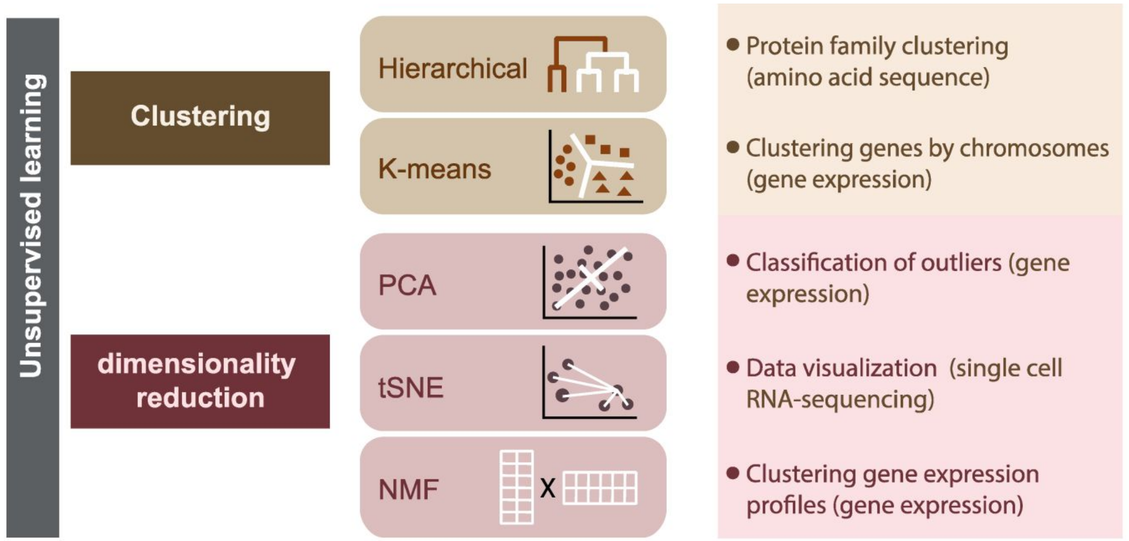




Deep Learning in Python

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

- T**he 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”)`

Exercise 5A
Regression Models

Part 5B
Model Evaluation in R

Train & Test Dataset

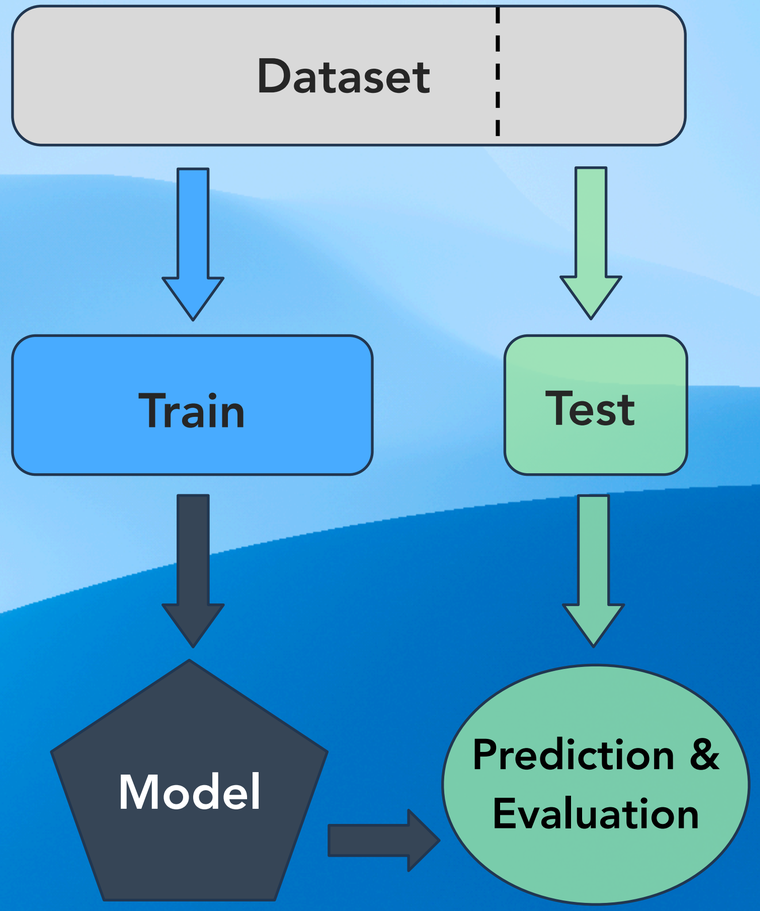
- We split our dataset into a **training set** and a **test set**

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

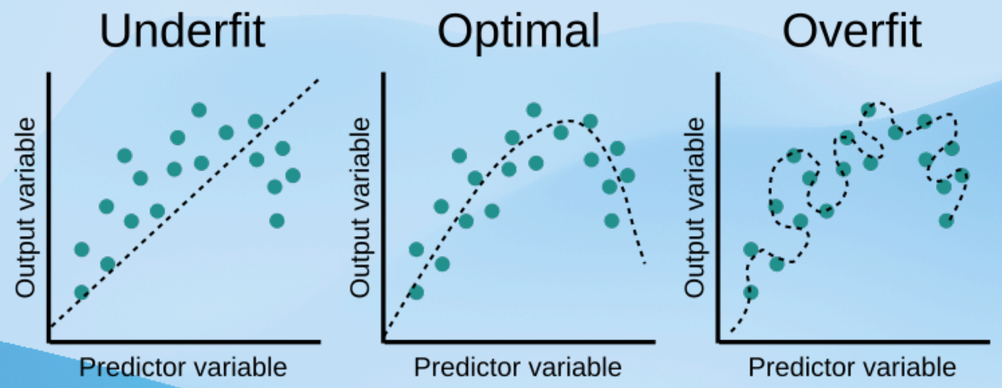
- **N.B:** Sets must be balanced and contain all observed classes!

R-syntax:

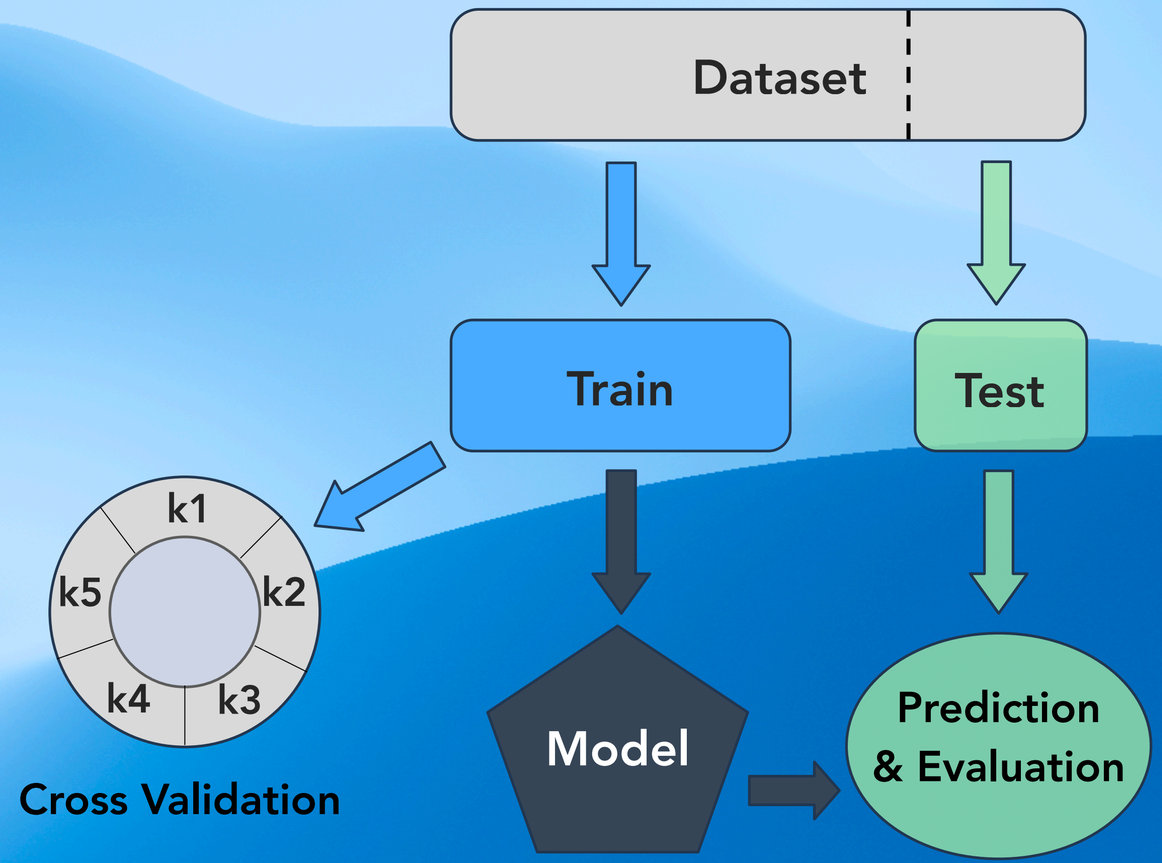
```
data <- read_csv("data/data.csv")

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

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



Model Overfitting



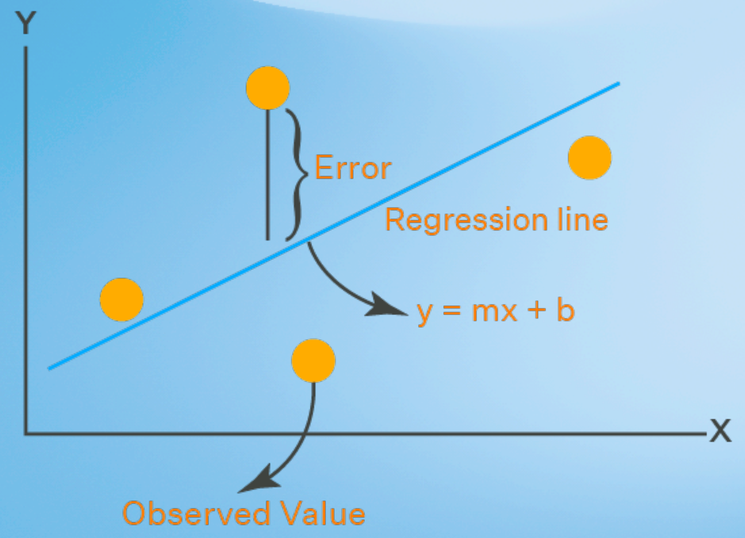
- When a model follows the data too closely we get an effect known as overfitting.
  - Splitting the data into training and evaluation we partly avoid this
  - By performing **cross validation** while training we may avoid it further



Model Evaluation Metrics

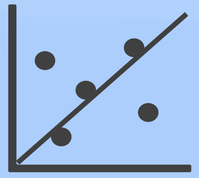
How well does the model fit the data?

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



REGRESSION

- R-squared
- MSE
- RMSE
- RMSLE



CLASSIFICATION

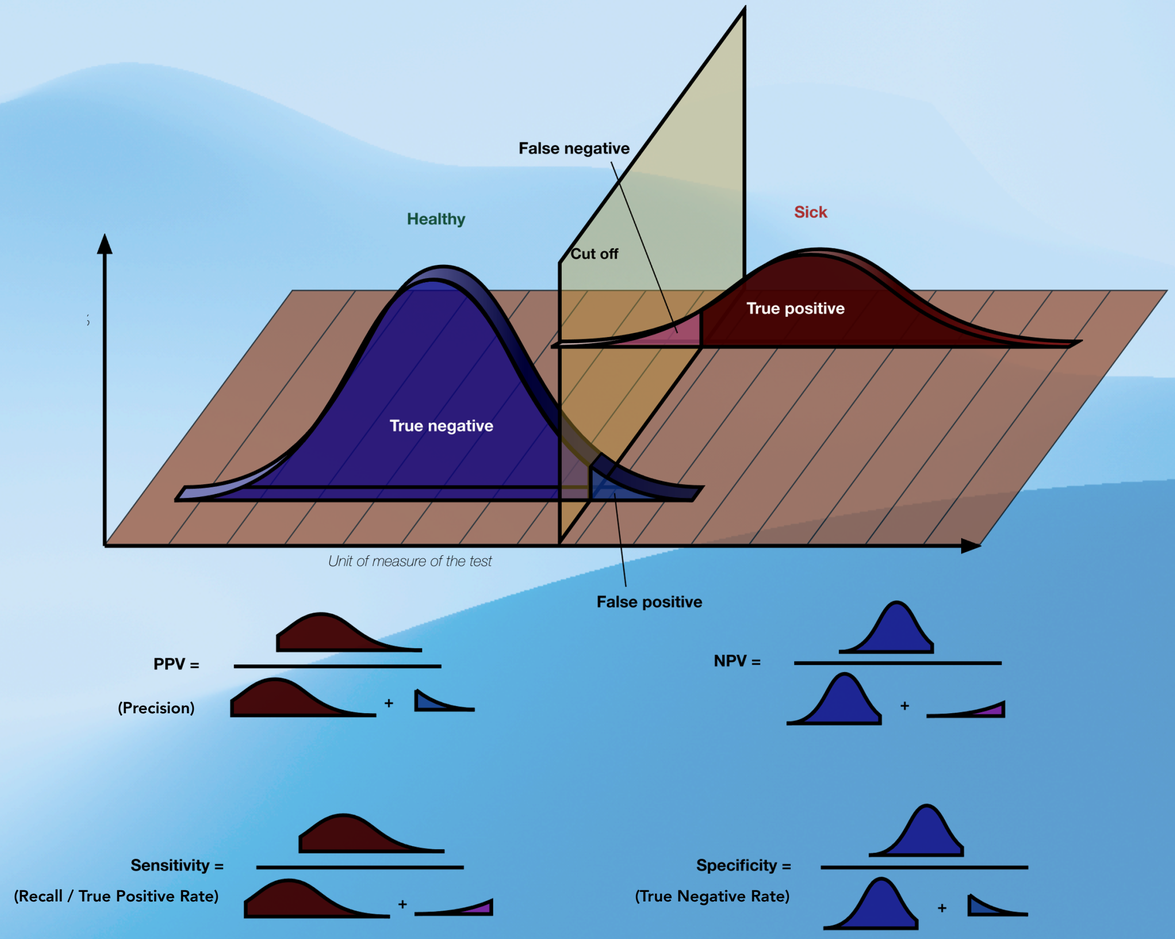
- Accuracy
- Precision
- Recall
- ROC/AUC

CLUSTERING

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

Classifier Performance

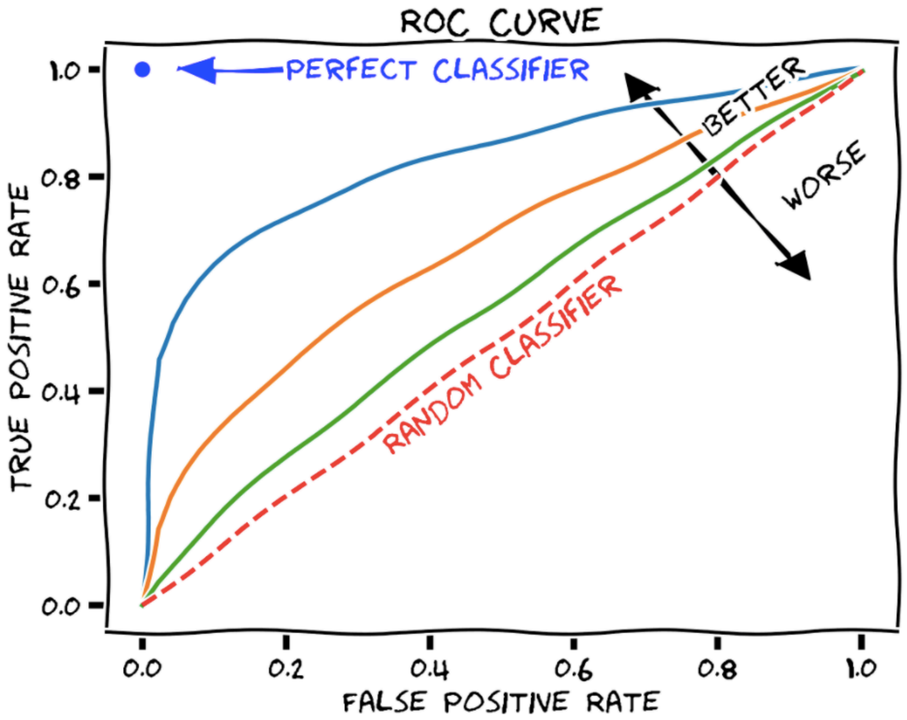
- **Recall** and **Precision** are often used as performance measures of classifiers.
- These metrics depend on the chosen cutoff.
- One can easily optimize on the number of false positives or false negatives, but not both at the same time.



ROC Curve - Area Under the Curve (AUC)

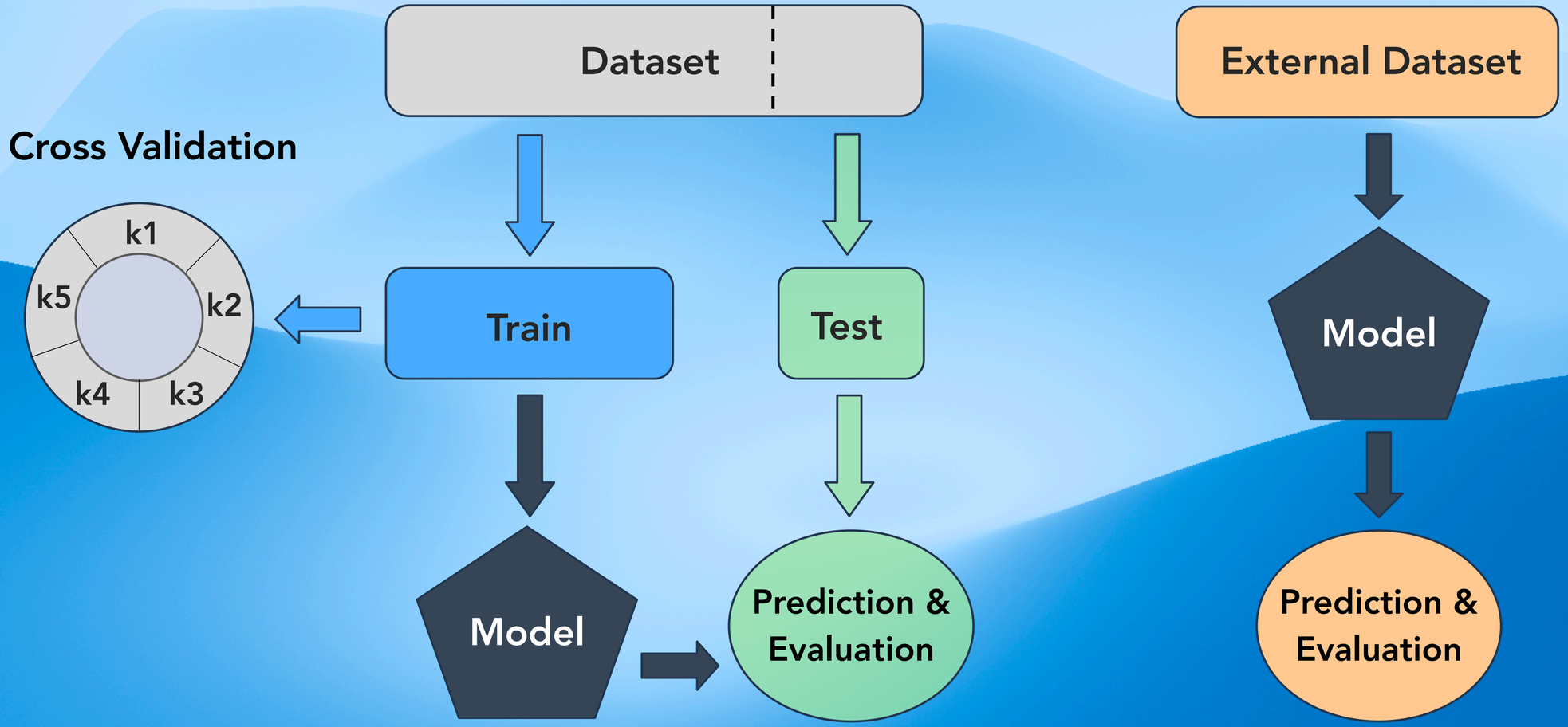
We can use the AUC to report on the goodness of the classification.

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

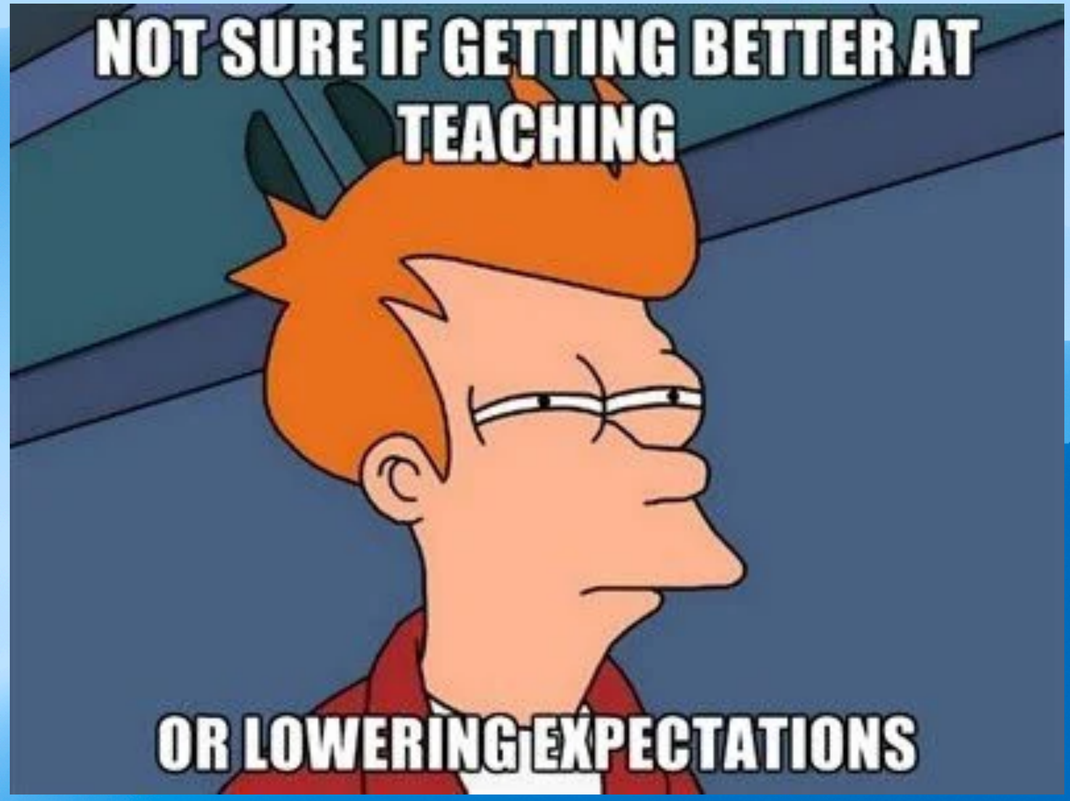


Exercise 5B
Models &
Model Evaluation

Course Evaluation

- We would love your feedback on this course:

https://forms.office.com/Pages/ResponsePage.aspx?id=kX-So6HNlkaviYyfHO_6kaKwKoywcVlMq5OaNsGbwftUNVA5NFBVSTE5V0hGUDFTQjlQQUxOME1DNCQlQCN0PWcu



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!