# Lab 4: Intro to Machine Learning

## Practice session covering topics discussed in Lecture 4

**M. Chiara Mimmi, Ph.D.** | Università degli Studi di Pavia

July 27, 2024

## GOAL OF TODAY'S PRACTICE SESSION

- Revisit PCA algorithm explored via MetaboAnalyst, to learn how we can compute it with R
- Understand some key elements of statistical Power Analysis
- Introduce how ML approaches deal with available data

The examples and datasets in this Lab session follow very closely two sources:

1. The tutorial on "Principal Component Analysis (PCA) in R" by: Statistics Globe
2. The materials in support of the "Core Statistics using R" course by: Martin van Rongen

# Topics discussed in Lecture # 4

## Lecture 4: topics

- Introduction to `MetaboAnalyst` software
  - A useful R-based resources for metabolomics
- Elements of statistical Power Analysis

# R ENVIRONMENT SET UP & DATA

# Needed R Packages

- We will use functions from packages `base`, `utils`, and `stats` (pre-installed and pre-loaded)
- We may also use the packages below (specifying `package::function` for clarity).

```r
# Load them for this R session

# --- General
library(here)     # tools find your project's files, based on working directory
library(dplyr)    # A Grammar of Data Manipulation
library(skimr)    # Compact and Flexible Summaries of Data
library(magrittr) # A Forward-Pipe Operator for R
library(readr)    # A Forward-Pipe Operator for R

# Plotting & data visualization
library(ggplot2)      # Create Elegant Data Visualisations Using the Grammar of Graphics
library(ggfortify)     # Data Visualization Tools for Statistical Analysis Results
library(scatterplot3d) # 3D Scatter Plot

# --- Statistics
library(MASS)       # Support Functions and Datasets for Venables and Ripley's MASS
library(factoextra) # Extract and Visualize the Results of Multivariate Data Analyses
library(FactoMineR) # Multivariate Exploratory Data Analysis and Data Mining
library(rstatix)    # Pipe-Friendly Framework for Basic Statistical Tests

# --- Tidymodels (meta package)
library(rsample)    # General Resampling Infrastructure
library(broom)      # Convert Statistical Objects into Tidy Tibbles
```

# DATASETS for today

In this tutorial, we will use:

- the biopsy data attached to the MASS package.
- a few clean datasets used in the “Core Statistics using R” course by: Martin van Rongen

# Dataset on Breast Cancer Biopsy

**Name**: Biopsy Data on Breast Cancer Patients
**Documentation**: See reference on the data downloaded and conditioned for R here https://cran.r-project.org/web/packages/MASS/MASS.pdf
**Sampling details**: This breast cancer database was obtained from the University of Wisconsin Hospitals, Madison from Dr. William H. Wolberg. He assessed biopsies of breast tumours for 699 patients up to 15 July 1992; each of nine attributes has been scored on a scale of 1 to 10, and the outcome is also known. The dataset contains the original Wisconsin breast cancer data with 699 observations on 11 variables.

# Importing Dataset `biopsy`

- The data can be interactively obtained form the `MASS` R package

```r
# (after loading pckg)
library(MASS) # Support Functions and Datasets for Venables and Ripley's MASS

# I can call
utils::data(biopsy)
```

# `biopsy` variables with description

| Variable | Type | Description |
|---|---|---|
| ID | character | Sample ID |
| V1 | integer 1 - 10 | clump thickness |
| V2 | integer 1 - 10 | uniformity of cell size |
| V3 | integer 1 - 10 | uniformity of cell shape |
| V4 | integer 1 - 10 | marginal adhesion |
| V5 | integer 1 - 10 | single epithelial cell size |
| V6 | integer 1 - 10 | bare nuclei (16 values are missing) |
| V7 | integer 1 - 10 | bland chromatin |
| V8 | integer 1 - 10 | normal nucleoli |
| V9 | integer 1 - 10 | mitoses |
| class | factor | benign or malignant |

# `biopsy` variables exploration 1/2

The `biopsy` data contains **699 observations of 11 variables**.

The dataset also contains a character variable: `ID`, and a factor variable: `class`, with two levels ("benign" and "malignant").

```
1 # check variable types
2 str(biopsy)
```

```
'data.frame':   699 obs. of  11 variables:
 $ ID   : chr  "1000025" "1002945" "1015425" "1016277" ...
 $ V1   : int  5 5 3 6 4 8 1 2 2 4 ...
 $ V2   : int  1 4 1 8 1 10 1 1 1 2 ...
 $ V3   : int  1 4 1 8 1 10 1 2 1 1 ...
 $ V4   : int  1 5 1 1 3 8 1 1 1 1 ...
 $ V5   : int  2 7 2 3 2 7 2 2 2 2 ...
 $ V6   : int  1 10 2 4 1 10 10 1 1 1 ...
 $ V7   : int  3 3 3 3 3 9 3 3 1 2 ...
 $ V8   : int  1 2 1 7 1 7 1 1 1 1 ...
 $ V9   : int  1 1 1 1 1 1 1 1 5 1 ...
 $ class: Factor w/ 2 levels "benign","malignant": 1 1 1 1 1 2 1 1 1 1 ...
```

# biopsy variables exploration 2/2

There is also one incomplete variable V6

- remember the package skimr for exploring a dataframe?

```
# check vars with missing values
biopsy %>%
  skimr::skim(starts_with("V")) %>%
  dplyr::select(skim_variable,
                n_missing)
```

```
# A tibble: 9 × 2
  skim_variable n_missing
  <chr>             <int>
1 V1                    0
2 V2                    0
3 V3                    0
4 V4                    0
5 V5                    0
6 V6                   16
7 V7                    0
8 V8                    0
9 V9                    0
```

# `biopsy` dataset manipulation

We will:

- exclude the non-numerical variables (`ID` and `class`) before conducting the PCA.
- exclude the individuals with missing values using the `na.omit()` or `filter(complete.cases()` functions.
- We can do both in 2 equivalent ways:

## with `base` R (more compact)

```
1 # new dataset
2 data_biopsy <- na.omit(biopsy[,-c(1,1
```

## with `dplyr` (more explicit)

```
1 # new dataset
2 data_biopsy <- biopsy %>%
3   # drop incomplete & non-integer col
4   dplyr::select(-ID, -class) %>%
5   # drop incomplete observations (row
6   dplyr::filter(complete.cases(.))
```

# biopsy dataset manipulation

We obtained a new dataset with 9 variables and 683 observations (instead of the original 699).

```
# check reduced dataset
str(data_biopsy)
```

```
'data.frame':   683 obs. of  9 variables:
 $ V1: int  5 5 3 6 4 8 1 2 2 4 ...
 $ V2: int  1 4 1 8 1 10 1 1 1 2 ...
 $ V3: int  1 4 1 8 1 10 1 2 1 1 ...
 $ V4: int  1 5 1 1 3 8 1 1 1 1 ...
 $ V5: int  2 7 2 3 2 7 2 2 2 2 ...
 $ V6: int  1 10 2 4 1 10 10 1 1 1 ...
 $ V7: int  3 3 3 3 3 9 3 3 1 2 ...
 $ V8: int  1 2 1 7 1 7 1 1 1 1 ...
 $ V9: int  1 1 1 1 1 1 1 1 5 1 ...
```

# ML: EXAMPLE of UNSUPERVISED ALGORITHM

# PCA in R

Reducing high-dimensional data to a lower number of variables

# Calculate Principal Components

The first step of PCA is to calculate the principal components. To accomplish this, we use the `prcomp()` function from the `stats` package.

- With argument `"scale = TRUE"` each variable in the biopsy data is scaled to have a mean of `0` and a standard deviation of `1` before calculating the principal components (just like option `Autoscaling` in MetaboAnalyst)

```
# calculate principal component
biopsy_pca <- prcomp(data_biopsy,
                     # standardize variables
                     scale = TRUE)
```

# Analyze Principal Components

Let's check out the elements of our obtained `biopsy_pca` object

- (All accessible via the `$` operator)

```
1 names(biopsy_pca)
```

```
[1] "sdev"     "rotation" "center"   "scale"    "x"
```

**"sdev"** = the standard deviation of the principal components

**"sdev"^2** = the variance of the principal components (**eigenvalues** of the covariance/correlation matrix)

**"rotation"** = the matrix of variable **loadings** (i.e., a matrix whose columns contain the **eigenvectors**).

**"center"** and **"scale"** = the means and standard deviations of the original variables before the transformation;

**"x"** = the principal component scores (after PCA the observations are expressed in principal component scores)

# Analyze Principal Components (cont.)

We can see the summary of the analysis using the `summary()` function

1. The first row gives the **Standard deviation** of each component, which can also be retrieved via `biopsy_pca$sdev`.
2. The second row shows the **Proportion of Variance**, i.e. the percentage of explained variance.

```
summary(biopsy_pca)
```

```
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     2.4289 0.88088 0.73434 0.67796 0.61667 0.54943 0.54259
Proportion of Variance 0.6555 0.08622 0.05992 0.05107 0.04225 0.03354 0.03271
Cumulative Proportion  0.6555 0.74172 0.80163 0.85270 0.89496 0.92850 0.96121
                           PC8     PC9
Standard deviation     0.51062 0.29729
Proportion of Variance 0.02897 0.00982
Cumulative Proportion  0.99018 1.00000
```

## Proportion of Variance for components

2. The row with **Proportion of Variance** can be either accessed from summary or calculated as follows:

```
# a) Extracting Proportion of Variance from summary
summary(biopsy_pca)$importance[2,]
```

```
    PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
0.65550 0.08622 0.05992 0.05107 0.04225 0.03354 0.03271 0.02897 0.00982
```

```
# b) (same thing)
round(biopsy_pca$sdev^2 / sum(biopsy_pca$sdev^2), digits = 5)
```

```
[1] 0.65550 0.08622 0.05992 0.05107 0.04225 0.03354 0.03271 0.02897 0.00982
```

> The output suggests the **1st principal component** explains around 65% of the total variance, the **2nd principal component** explains about 9% of the variance, and this goes on with diminishing proportion for each component.

## Cumulative Proportion of variance for components

3. The last row from the `summary(biopsy_pca)`, shows the **Cumulative Proportion** of variance, which calculates the cumulative sum of the Proportion of Variance.

```
# Extracting Cumulative Proportion from summary
summary(biopsy_pca)$importance[3,]
```

```
    PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
0.65550 0.74172 0.80163 0.85270 0.89496 0.92850 0.96121 0.99018 1.00000
```

> Once you computed the PCA in R you must decide the number of components to retain based on the obtained results.

# VISUALIZING OUTPUTS

# Scree plot

There are several ways to decide on the number of components to retain.

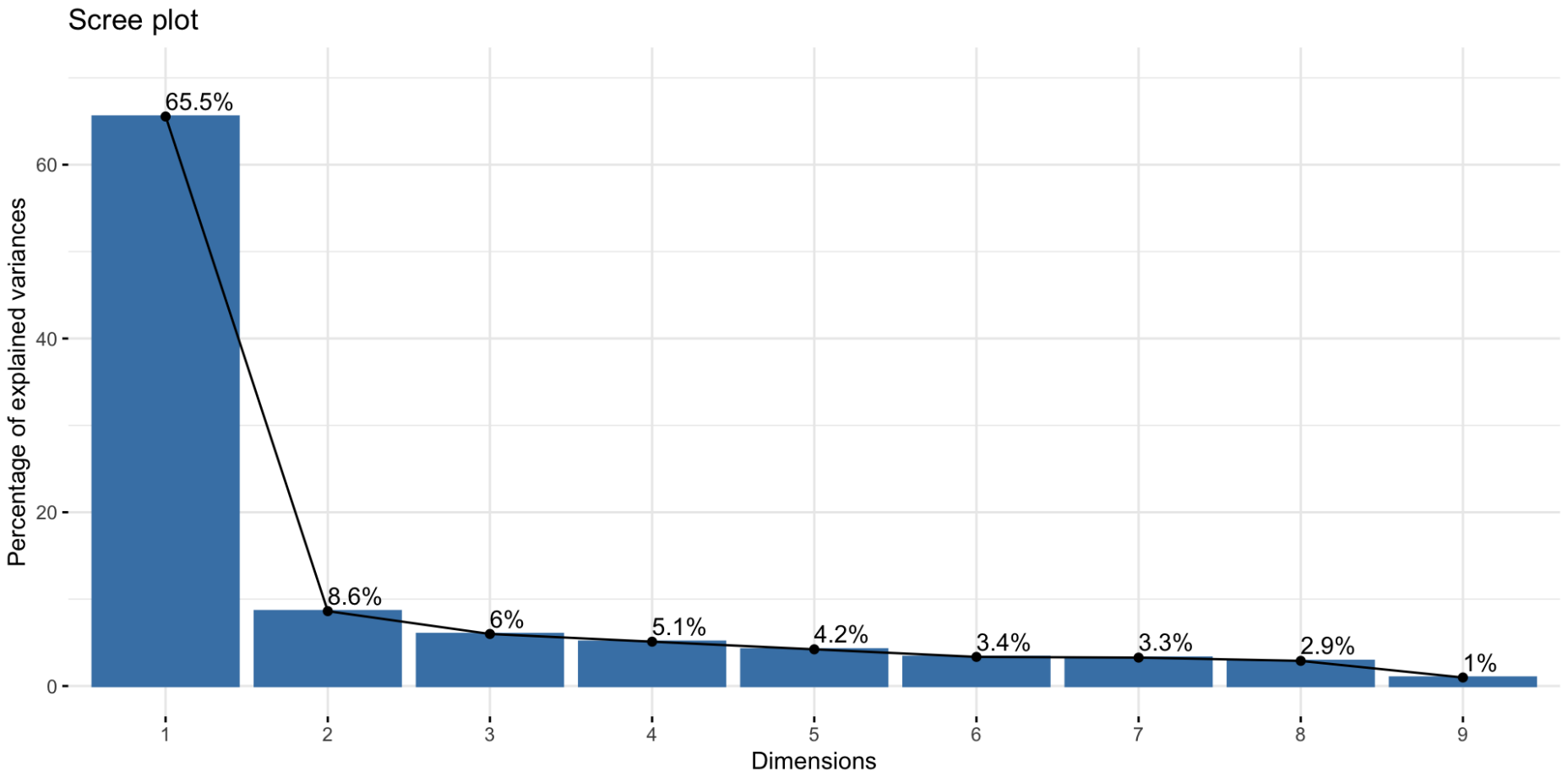
- One helpful option is visualizing the percentage of explained variance per principal component via a **scree plot**.
  - Plotting with the `fviz_eig()` function from the `factoextra` package

```
# Scree plot shows the variance of each principal component
factoextra::fviz_eig(biopsy_pca,
                     addlabels = TRUE,
                     ylim = c(0, 70))
```

> Visualization is essential in the interpretation of PCA results. Based on the number of retained principal components, which is usually the first few, the observations expressed in component scores can be plotted in several ways.

# Scree plot

The obtained **scree plot** simply visualizes the output of `summary(biopsy_pca)`.



# Principal Component `Scores` plot

After a PCA, the observations are expressed as **principal component scores**.
It is also important to visualize the observations along the new axes (principal components) to interpret the relations in the dataset:

1. We need to retrieve the principal component scores by calling `biopsy_pca$x`, and store them in a data set called `PC_scores`.
2. Next we draw a `scatterplot` of the observations – expressed in terms of principal components

```
1 PC_scores <- as.data.frame(biopsy_pca$x)
2 head(PC_scores)
```

# Principal Component Scores plot

```
         PC1         PC2         PC3         PC4         PC5         PC6
1  1.469095 -0.10419679  0.56527102 -0.03193593  0.15088743 -0.05997679
2 -1.440990 -0.56972390 -0.23642767 -0.47779958 -1.64188188  0.48268150
3  1.591311 -0.07606412 -0.04882192 -0.09232038  0.05969539  0.27916615
4 -1.478728 -0.52806481  0.60260642  1.40979365  0.56032669 -0.06298211
5  1.343877 -0.09065261 -0.02997533 -0.33803588  0.10874960 -0.43105416
6 -5.010654 -1.53379305 -0.46067165  0.29517264 -0.39155544 -0.11527442
         PC7        PC8          PC9
1 -0.3491471  0.4200360  0.005687222
2  1.1150819  0.3792992 -0.023409926
3 -0.2325697  0.2096465 -0.013361828
4  0.2109599 -1.6059184 -0.182642900
5 -0.2596714  0.4463277  0.038791241
6 -0.3842529 -0.1489917  0.042953075
```

# Principal Component `Scores` plot (adding label variable)

3. In addition, if the data includes a factor variable, like in this case, it may be interesting to show the grouping on the plot as well.

- In such cases, the label variable `class` can be added to the PC set as follows.

```
biopsy_no_na <- na.omit(biopsy)
# adding grouping variable
PC_scores$Label <- biopsy_no_na$class
```

The visualization of the observation points (point cloud) could be in 2D or 3D.

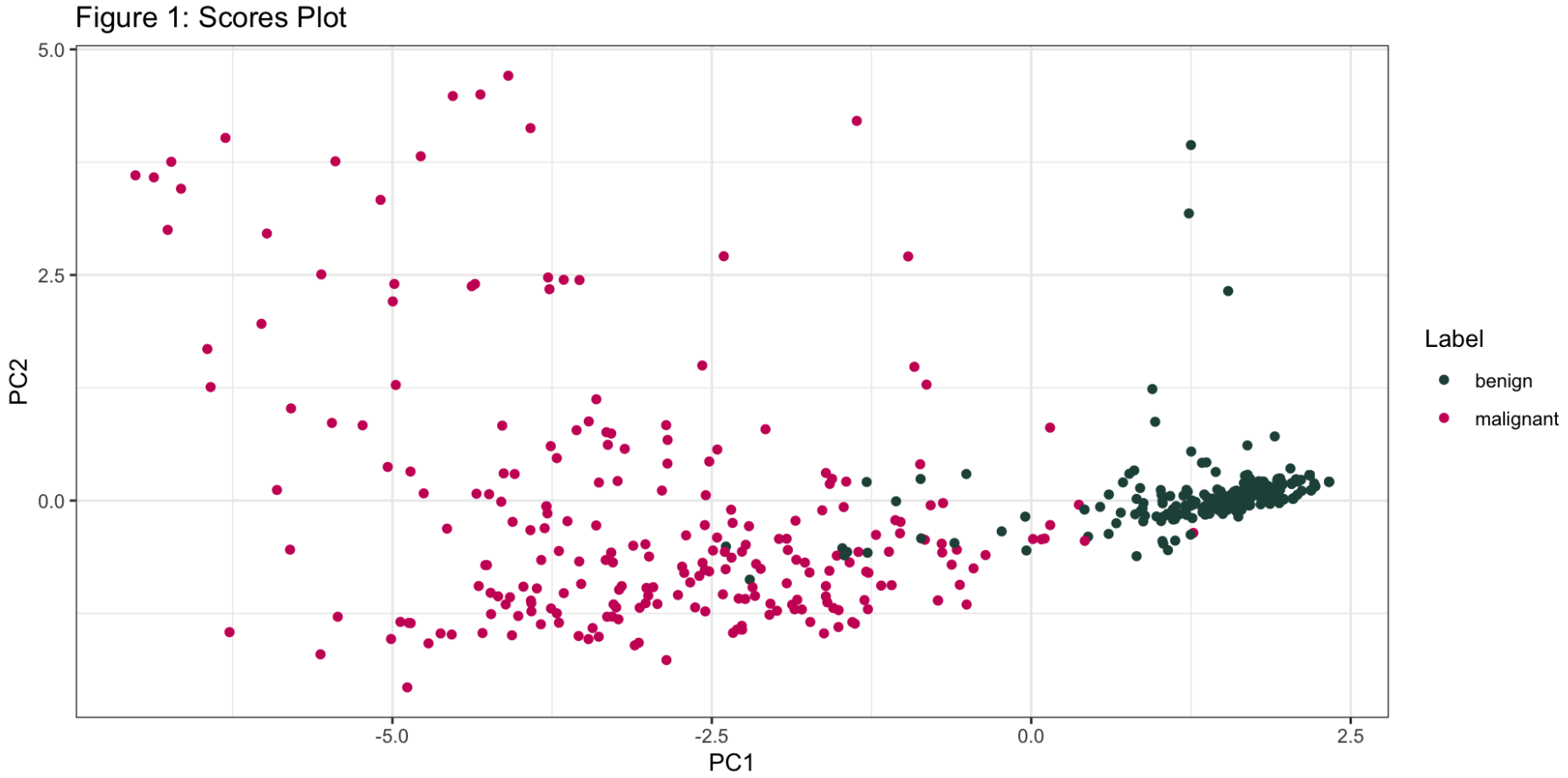
# Principal Component Scores plot (2D)

The Scores Plot can be visualized via the ggplot2 package. The grouping is indicated by the color argument (Label); the geom_point() is used to plot the point cloud.

```
ggplot(PC_scores,
       aes(x = PC1,
           y = PC2,
           color = Label)) +
  geom_point() +
  scale_color_manual(values=c("#245048", "#CC0066")) +
  ggtitle("Figure 1: Scores Plot") +
  theme_bw()
```

Figure 1 shows the observations projected into the new data space made up of principal components

# Principal Component Scores plot (2D)



# Principal Component Scores (2D Ellipse Plot)

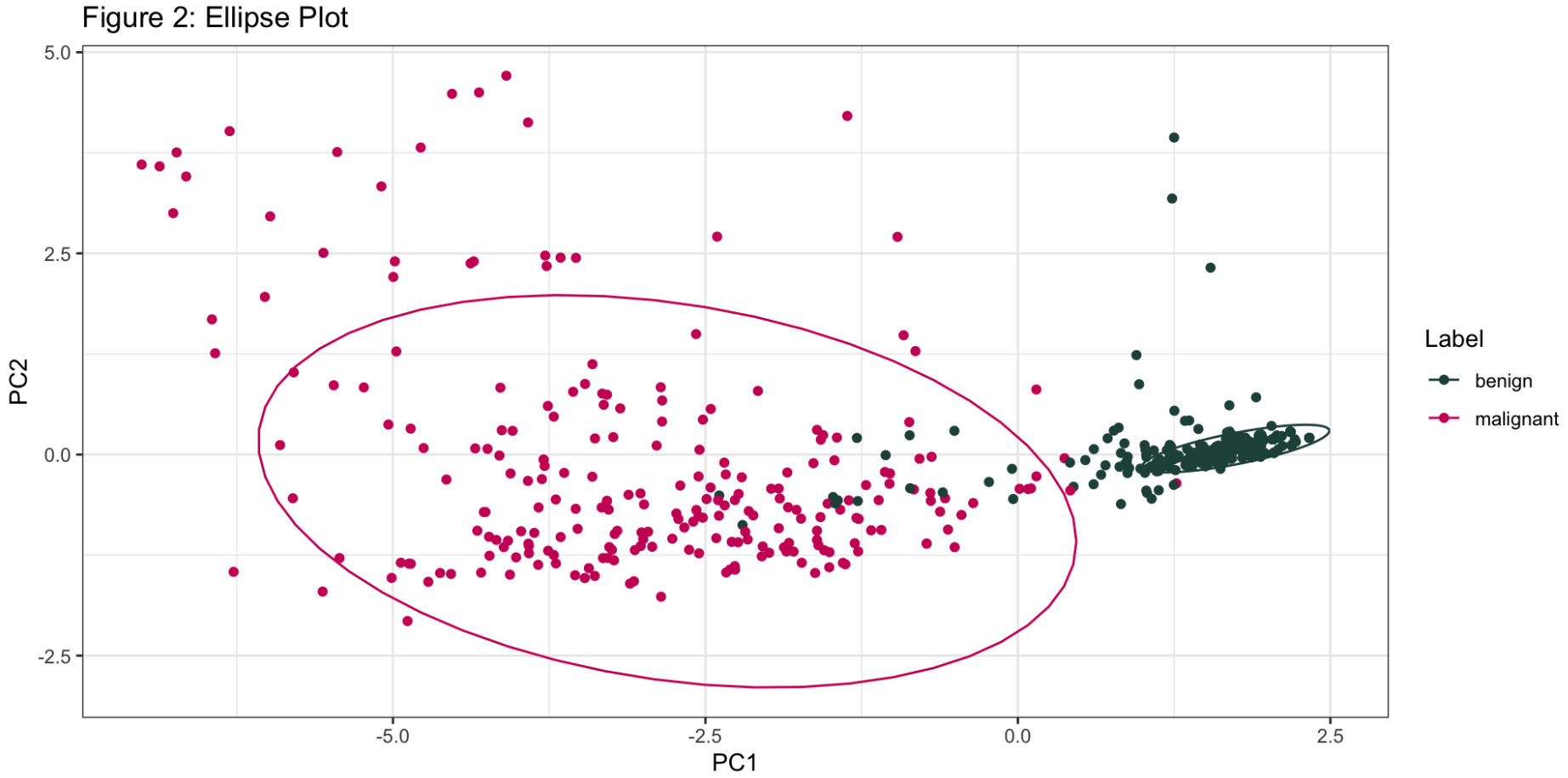
Confidence ellipses can also be added to a grouped scatter plot visualized after a PCA. We use the `ggplot2` package.

- grouping is indicated by argument the `color = Label`;
- `geom_point()` is used for the point cloud;
- the `stat_ellipse()` function is called to add the ellipses per biopsy group.

```
ggplot(PC_scores,
       aes(x = PC1,
           y = PC2,
           color = Label)) +
  geom_point() +
  scale_color_manual(values=c("#245048", "#CC0066")) +
  stat_ellipse() +
  ggtitle("Figure 2: Ellipse Plot") +
  theme_bw()
```

Figure 2 shows the observations projected into the new data space made up of principal components, with 95% confidence regions displayed.

# Principal Component Scores (2D Ellipse Plot)



# Principal Component Scores plot (3D)

A 3D scatterplot of observations shows the first three principal components' scores.

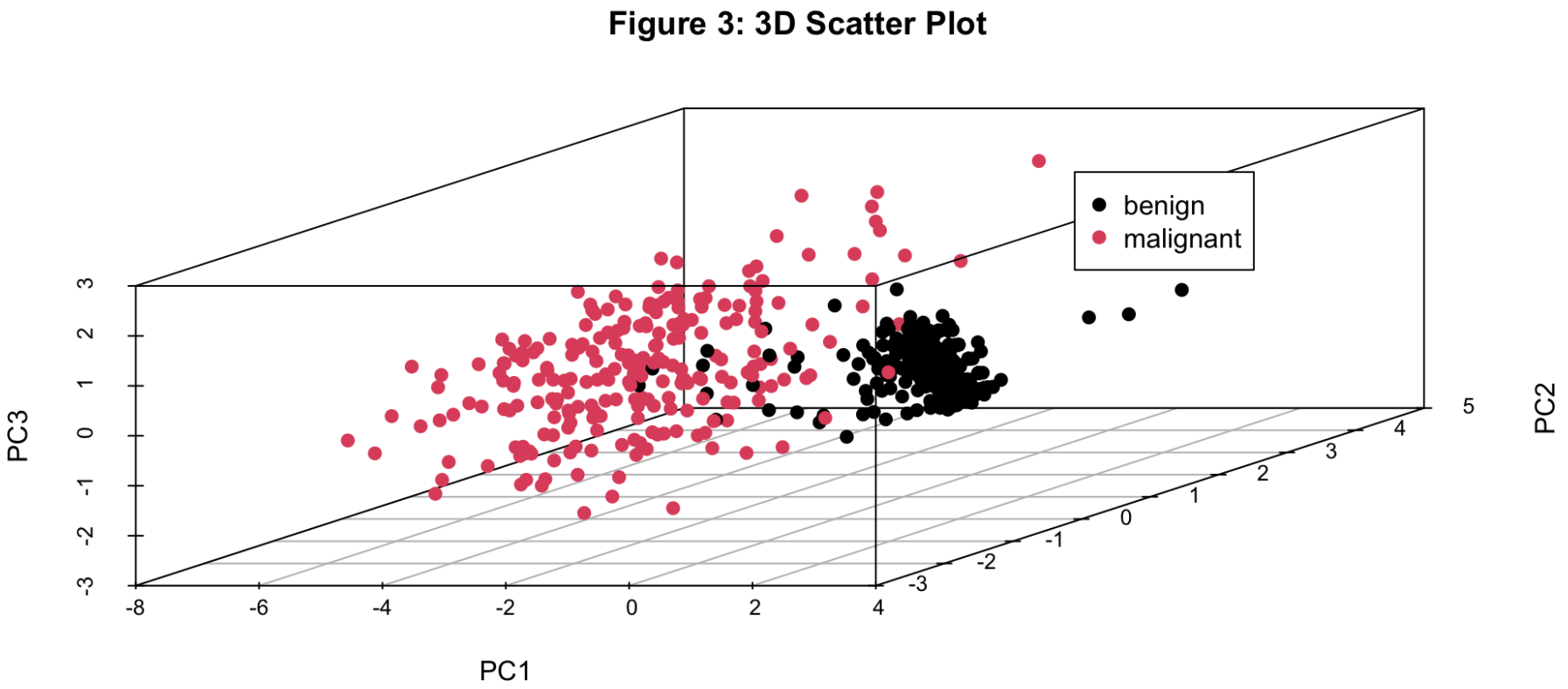
- We use the `scatterplot3d()` function of the `scatterplot3d` package, with the color argument assigned to the Label variable
- To add a legend, we use the `legend()` function and specify its coordinates via the `xyz.convert()` function.

```
# 3D scatterplot ...
plot_3d <- with(PC_scores,
                scatterplot3d::scatterplot3d(PC_scores$PC1,
                                             PC_scores$PC2,
                                             PC_scores$PC3,
                                             color = as.numeric(Label),
                                             pch = 19,
                                             main ="Figure 3: 3D Scatter Plot",
                                             xlab="PC1",
                                             ylab="PC2",
                                             zlab="PC3"))

# ... + legend
legend(plot_3d$xyz.convert(0.5, 0.7, 0.5),
       pch = 19,
       yjust=-0.6,
       xjust=-0.9,
       legend = levels(PC_scores$Label),
       col = seq_along(levels(PC_scores$Label)))
```

Figure 3 shows the observations projected into the new 3D data space made up of principal components.

# Principal Component Scores plot (3D)



## Biplot: principal components v. original variables

Next, we use another special type of scatterplot (a **biplot**) to understand the relationship between the principal components and the original variables.

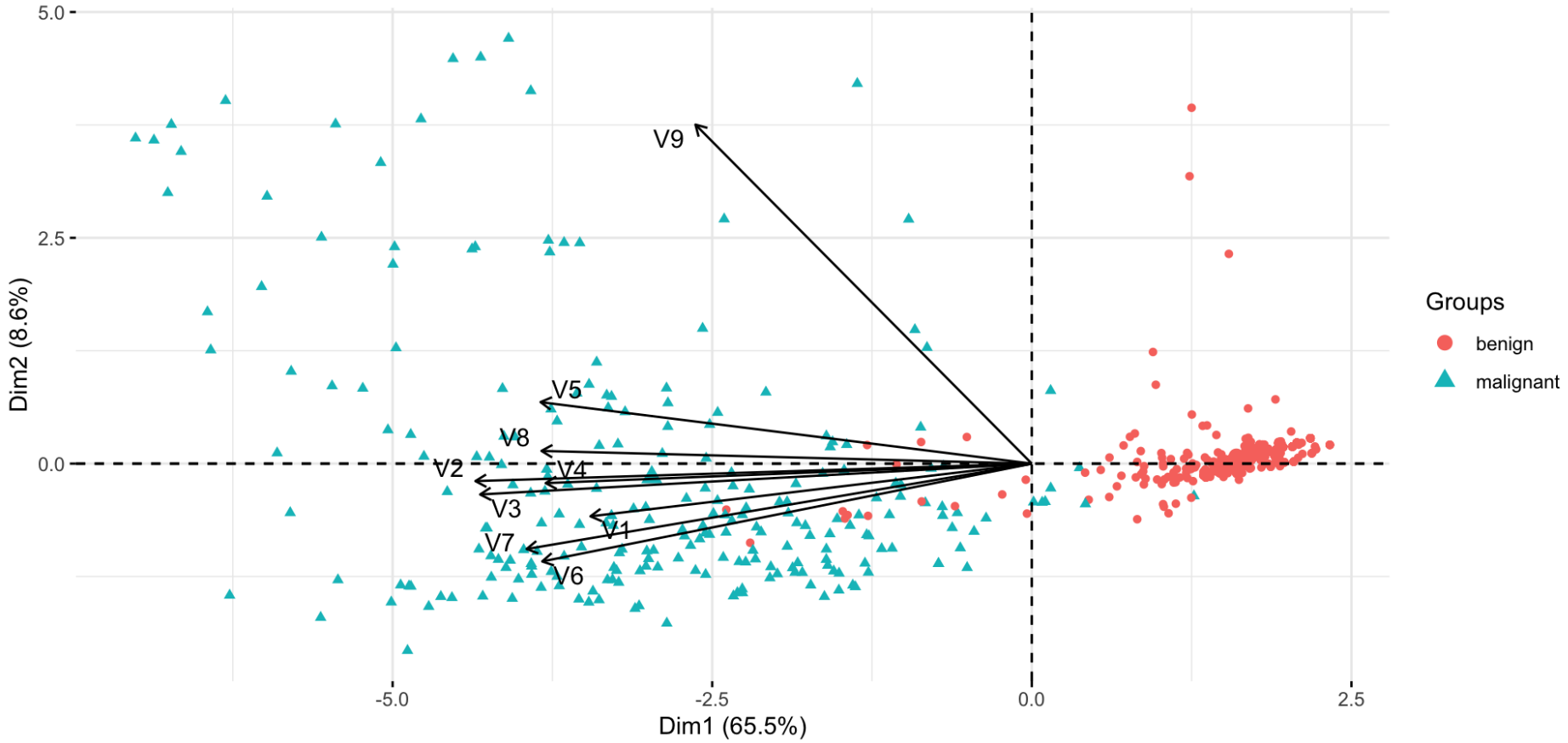
- For this we have the `fviz_pca_biplot()` function from the `factoextra` package
- We will specify the color for the variables, or rather, for the “loading vectors”
- The `habillage` argument allows to highlight with color the grouping by `class`

```
factoextra::fviz_pca_biplot(biopsy_pca,
                repel = TRUE,
                col.var = "black",
                habillage = biopsy_no_na$class,
                title = "Figure 4: Biplot", geom="point")
```

## Biplot: principal components v. original variables

The axes show the principal component scores, and the vectors are the loading vectors

Figure 4: Biplot



# Interpreting biplot output

Biplots have two components: **scores** and **loading vectors**. As in the scores plot each point represents a sample in the space of principal components.

- Biopsies of the same class are located closer to each other, which indicates that they have similar principal component scores.
- The loading vectors represent the original variables in the space of principal components. As expected from PCA, the single `PC1` accounts for variance in almost all original variables, while `V9` has the major projection along `PC2`.
- The loading vectors represent strength and direction of association of original variables with new PC variables.

# Interpreting biplot output (cont.)

```
scores <- biopsy_pca$x

loadings <- biopsy_pca$rotation
# excerpt of first 2 components
loadings[ ,1:2]
```

```
          PC1         PC2
V1 -0.3020626 -0.14080053
V2 -0.3807930 -0.04664031
V3 -0.3775825 -0.08242247
V4 -0.3327236 -0.05209438
V5 -0.3362340  0.16440439
V6 -0.3350675 -0.26126062
V7 -0.3457474 -0.22807676
V8 -0.3355914  0.03396582
V9 -0.2302064  0.90555729
```

# SAMPLE SIZE DETERMINATION IN INFERENTIAL STATISTICS

> *“OK, but how big of a sample do I need?* ...the 1,000,000 $ question”!* 🙀

## Purpose and challenges of Power Analysis

- `Power analysis` helps with the key question **How many observations/subjects do I need for my experiment?** ($= n$)
  - **Too small** of a sample size can under detect the effect of interest in your experiment
  - **Too large** of a sample size may lead to unnecessary wasting of resources
  - We strive to have **just the sufficient** number of observations needed to have a good chance of detecting the effect researched. (Even more so in a very time-consuming or expensive experiment.)
- **When** should we do power analysis?
  - (Ideally), *before* the experiment: `a priori power analysis` allows *to determine the necessary sample size* $n$ *of a test*, given a desired $\alpha$ level, a desired power level ($1 - \beta$), and the size of the effect to be detected (a measure of difference between $H_o$ and $H_1$)
  - In reality, sometimes you can only do `post-hoc power analysis` *after* the experiment, so the sample size $n$ is already given.
    - In this case, given $n$, $\alpha$, and a specified effect size, the analysis will return the power ($1 - \beta$) of the test, or $\beta$ (i.e. the probability of Type II error = incorrectly retaining $H_o$).

## Required inputs to define the sample size n

- A specified **effect size** (i.e. the minimum deviation from $H_o$ that you hope to detect for a meaningful result)
  - The larger the effect size, the easier it is to detect an effect and require fewer obs
- 
  - As $standarddeviation$ gets bigger, it is harder to detect a significant difference, so you'll need a bigger sample size.

- $\alpha$ is the **significance level** of the test (i.e. *the probability of incorrectly rejecting the null hypothesis (a false positive).*
  - Understanding if the test is one-tailed (difference has a direction) or two-tailed
- $\beta$ is *the probability of accepting the null hypothesis, even though it is false (a false negative)*, when the real difference is equal to the minimum effect size.
  - $1 - \beta$ is the **power of a test** is *the probability of correctly rejecting the null hypothesis (getting a significant result) when the real difference is equal to the minimum effect size.*
    - a power of 80% (equivalent to a beta of 20%) is probably the most common, while some people use 50% or 90%

# Specifying effect size

So (since $\alpha$ and $1-\beta$ are normally set) the key piece of information we need is the **effect size**, which is essentially a function of the difference between the means of the null and alternative hypotheses over the variation (standard deviation) in the data.

> The tricky part is that effect size is related to biological/practical significance rather than statistical significance

How should you estimate a meaningful `Effect Size`?

- Use preliminary information in the form of pilot study
- Use background information in the form of similar studies in the literature
- (With no prior information), make an estimated guess on the effect size expected (see guidelines next)

> Most R functions for sample size only allow you to enter effect size as input

## Specifying effect size: general guidelines

As a general indicative reference, below are the **"Cohen's Standard Effect Sizes"** (from statistician Jacob Cohen who came up with a rough set of numerical measures for "small", "medium" and "large" effect sizes that are still in use today)

| Test | Effect Size | Small | Medium | Large |
|---|---|---|---|---|
| All t-tests:<br>• one-sample t-test<br>• independent samples t-test<br>• paired samples t-test | Cohen's d | 0.20 | 0.50 | 0.80 |
| Difference between many means (ANOVA) | Cohen's f | 0.10 | 0.25 | 0.40 |
| Chi-squared test | Cohen's w | 0.10 | 0.30 | 0.50 |
| Pearson's correlation coefficient | Pearson's $\rho$ | 0.10 | 0.30 | 0.50 |
| Linear Regression (entire model) | Cohen's $f^2$ | 0.02 | 0.15 | 0.35 |

# The `pwr` package

The `pwr` package (develped by Stéphane Champely), implements power analysis as outlined by Cohen (1988). The key arguments of the function `pwr.t.test` are 4 quantities, plus 2 for the test description:

1. `n` = sample size
2. `d` = effect size (based on Cohen's)
3. `sig.level` = the desired significance level

- The significance level ($\alpha$) defaults to 0.05. Therefore, to calculate the significance level, given an effect size, sample size, and power ($1 - \beta$), use the option `"sig.level=NULL"`.

4. `power` = the desired power
5. `type` = the type of t-test you will eventually be carrying out (one of `two.sample`, `one.sample` or `paired`)
6. `alternative` = the type of alternative hypothesis you want to test (one of `two.sided`, `less` or `greater`)

- The core idea behind its functions is that **you enter 3 of the 4 quantities** (effect size, sample size, significance level, power) **and the 4th is calculated**.

# One Sample Mean: EXE data

GOAL: Imagine this is a *pilot study*, in which we tested fish is (on average) different form 20 cm in length.

The `guanapo_data` dataset contains information on fish lengths from the Guanapo river pilot

```r
# Load data
fishlength_data <- readr::read_csv(here::here("practice", "data_input", "04_datasets",
                                              "fishlength.csv"),
                                   show_col_types = FALSE)

# Select a portion of the (pilot) data
guanapo_data <- fishlength_data %>%
  dplyr::filter(river == "Guanapo")

# Pilot experiment data
names(guanapo_data)
```

```
[1] "id"     "river"  "length"
```

```r
mean_H1 <-  mean(guanapo_data$length) # 18.29655
mean_H1
```

```
[1] 18.29655
```

```r
sd_sample <- sd(guanapo_data$length)  # 2.584636
sd_sample
```

```
[1] 2.584636
```

## One Sample Mean t-test: EXAMPLE cont.

Let's compute the one sample t-test with `stats::t.test` against a hypothetical average fish length ($\text{mean\_H}_o = 20$ )

```
# Hypothetical length mean H0
mean_H0 <- 20

t_stat <- stats::t.test(x = guanapo_data$length,
                        mu = mean_H0,
                        alternative = "two.sided")
# one-sample t-test results
t_stat
```

```
    One Sample t-test

data:  guanapo_data$length
t = -3.5492, df = 28, p-value = 0.001387
alternative hypothesis: true mean is not equal to 20
95 percent confidence interval:
 17.31341 19.27969
sample estimates:
mean of x
 18.29655
```

- There appear to be a statistically significant result here: the mean length of the fish appears to be different from 20 cm.

QUESTION: In a new study of the same fish, what sample size n would you need to get a comparable result?

## One Sample Mean t-test: POWER ANALYSIS (n)

- We input Cohen's d (after calculating it manually) following: $\text{effect size} \approx \frac{\text{Mean}_{H_1} - \text{Mean}_{H_0}}{\text{Std Dev}}$
- We use `pwr::pwr.t.test` to calculate the minimum sample size `n` required:

```r
# Cohen's d formula
eff_size <- (mean_H1 - mean_H0) / sd_sample # -0.6590669

# power analysis to actually calculate the minimum sample size required:
pwr::pwr.t.test(d = eff_size,
                sig.level = 0.05,
                power = 0.8,
                type = "one.sample")
```

```
One-sample t test power calculation

          n = 20.07483
          d = 0.6590669
  sig.level = 0.05
      power = 0.8
alternative = two.sided
```

> We would need `n = 21` (rounding up) observations for an experiment (e.g. in different river) to detect an effect size as the pilot study at a 5% significance level and 80% power.

## One Sample Mean t-test: POWER ANALYSIS, stricter conditions

What if we wanted the results to be even more stringent?

- e.g. require higher significance level (0.01) and power (0.90) with the same effect?

```
# power analysis to actually calculate the minimum sample size required:
pwr::pwr.t.test(d = eff_size,
                sig.level = 0.01,
                power = 0.9,
                type = "one.sample")
```

```
     One-sample t test power calculation

              n = 37.62974
              d = 0.6590669
      sig.level = 0.01
          power = 0.9
    alternative = two.sided
```

> This time, we would need n = 38 observations for an experiment to detect the same effect size at the stricter level of significance and power.

# Two Independent Samples: EXE data

Let's look at the entire `fishlength_data` with the lengths of fish from 2 separate rivers.

```
# Explore complete data
fishlength_data %>%
  dplyr::group_by (river) %>%
  dplyr::summarise (N = n(),
                    mean_len = mean(length),
                    sd_len = sd(length))
```

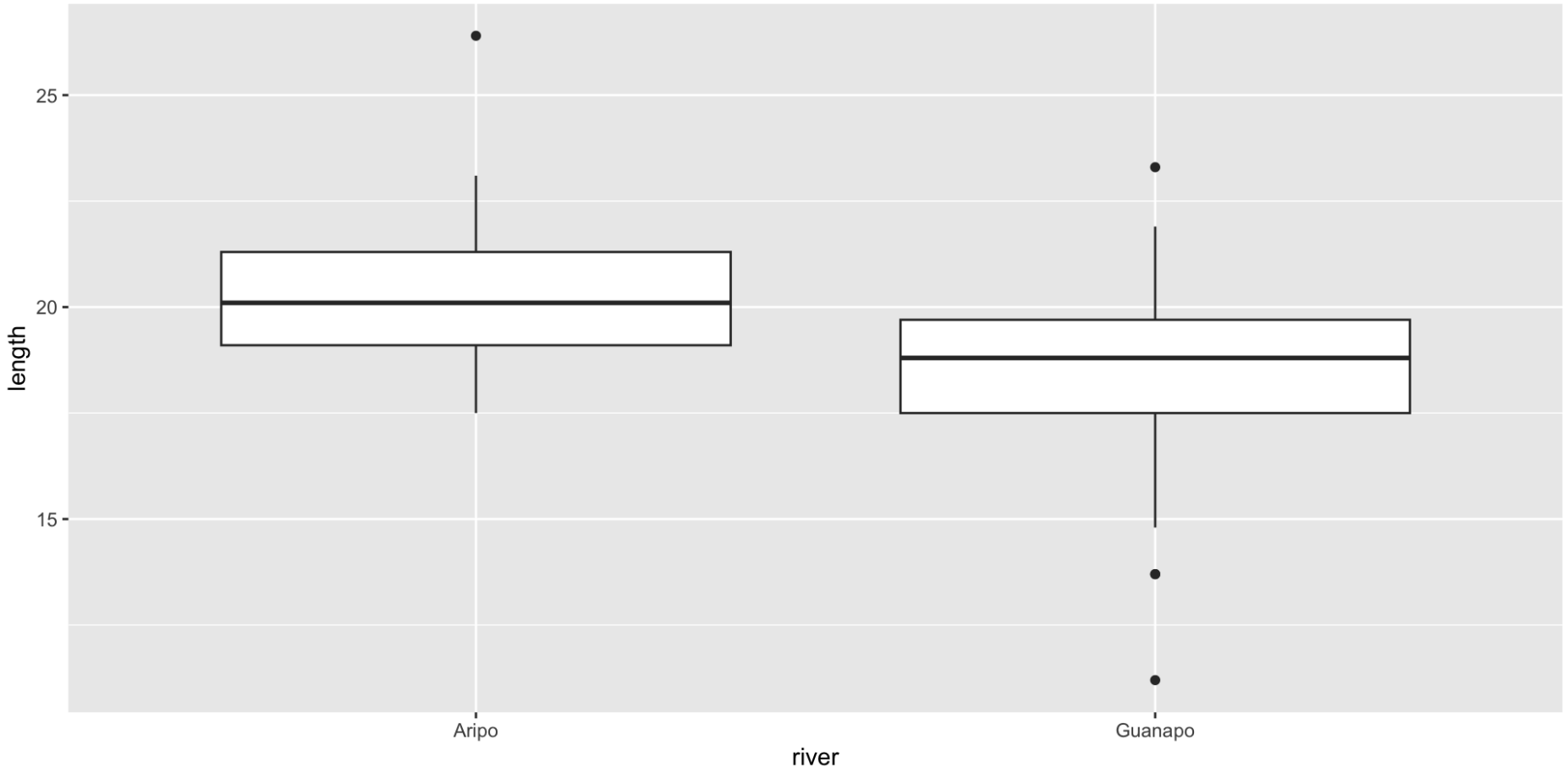
```
# A tibble: 2 × 4
  river       N mean_len sd_len
  <chr>   <int>    <dbl>  <dbl>
1 Aripo      39     20.3   1.78
2 Guanapo    29     18.3   2.58
```

Visualize quickly the 2 samples (rivers) with a boxplot

```
# visualize the data
fishlength_data %>%
  ggplot(aes(x = river, y = length)) +
  geom_boxplot()
```

# Two Independent Samples: EXE data

The fish in the 2 samples appear to have different mean length



# Two Independent Samples: t-test

Let's confirm it with a two sample t-test against $H_0$: *The two population means are equal*

```
# Perform two-samples unpaired test
fishlength_data %>%
  rstatix::t_test(length ~ river,
                  paired = FALSE
                    )
```

```
# A tibble: 1 × 8
  .y.    group1 group2     n1    n2 statistic    df        p
* <chr>  <chr>  <chr>   <int> <int>     <dbl> <dbl>    <dbl>
1 length Aripo  Guanapo    39    29      3.64  46.9 0.00067
```

> The t-test analysis confirms that the difference is significant.

QUESTION: Can we use this information to design a more `efficient` experiment? I.e. run an experiment powerful enough to pick up the same observed difference in means but with **fewer observations**?

## Two Independent Samples: POWER ANALYSIS 1/2

1. Let's work out exactly the **effect size** of this study by estimating Cohen's d using this data.

- (We use a function from the package `rstatix::cohens_d` to estimate Cohen's d)

```
# Estimate cohen's d
fishlength_data %>%
  rstatix::cohens_d(length ~ river, var.equal = TRUE)
```

```
# A tibble: 1 × 7
  .y.    group1 group2  effsize    n1    n2 magnitude
* <chr>  <chr>  <chr>     <dbl> <int> <int> <ord>
1 length Aripo  Guanapo   0.942    39    29 large
```

> The `effsize` column contains the information that we want, in this case **0.94**

## Two Independent Samples: POWER ANALYSIS 2/2 (n)

2. Actually answer the question about **how many fish** we really need to catch in the future

```r
# run power analysis
pwr::pwr.t.test(d = 0.94, power = 0.8, sig.level = 0.05,
                type = "two.sample", alternative = "two.sided")
```

```
     Two-sample t test power calculation

              n = 18.77618
              d = 0.94
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number in *each* group
```

> The n output ( = **19 observations per group**) -as opposed to 39 + 29- would be sufficient if we wanted to confidently detect the difference observed in the previous study

# Two Paired Samples t-test: EXE data

The `cortisol_data` dataset contains information about cortisol levels measured on 20 participants in the morning and evening

```
# Load data
cortisol_data <- read.csv(file = here::here("practice", "data_input", "04_datasets",
                                            "cortisol.csv"),
                          header = TRUE, # 1st line is the name of the variables
                          sep = ",", # which is the field separator character.
                          na.strings = c("?","NA" ), # specific MISSING values
                          row.names = NULL)

# Explore data
names(cortisol_data)
```

```
[1] "patient_id" "time"       "cortisol"
```

```
cortisol_data %>%
  dplyr::group_by (time) %>%
  dplyr::summarise (
    N = n(),
    mean_cort = mean(cortisol),
    sd_cort = sd(cortisol))
```

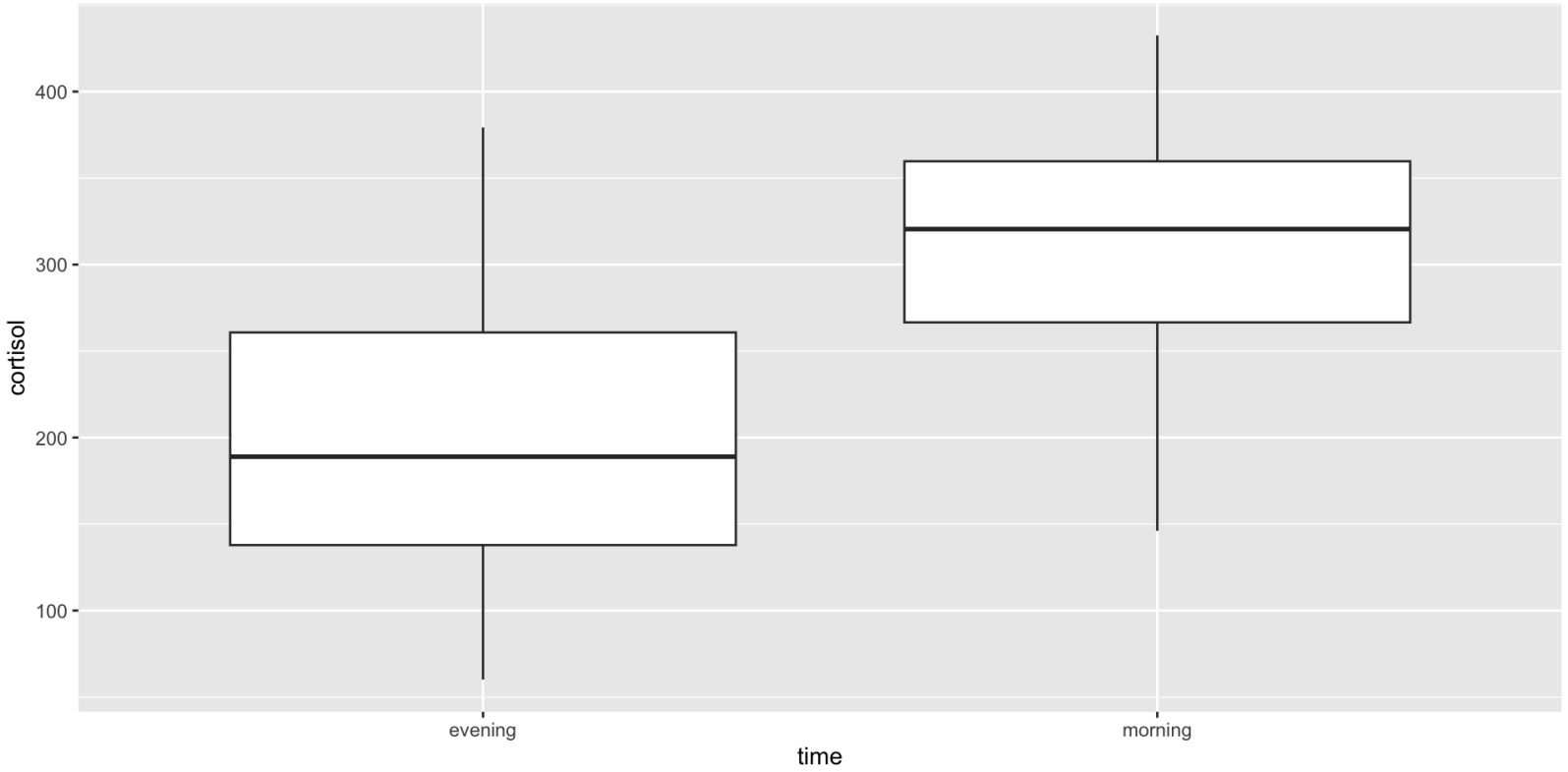
```
# A tibble: 2 × 4
  time        N mean_cort sd_cort
  <chr>   <int>     <dbl>   <dbl>
1 evening    20      197.    87.5
2 morning    20      313.    73.8
```

> Notice the difference in the paired sample means is quite large

## Two Paired Samples t-test: visualization

Visualize quickly the 2 paired samples (morning and evening) with a boxplot

```r
# visualize the data
cortisol_data %>%
  ggplot(aes(x = time, y = cortisol)) +
  geom_boxplot()
```




The cortisol levels in the 2 paired amples appear quite different

## Two Paired Samples: POWER ANALYSIS (d)

GOAL: Flipping the question, if we know the given `n` (20 patients observed twice): How big should the `effect size` be to be detected at `power` of 0.8 and `significance level` 0.05?

- We use `pwr::pwr.t.test`, with the argument specification `type = "paired"`, but this time to estimate the **effect size**

```r
# power analysis to actually calculate the effect size at the desired conditions:
pwr::pwr.t.test(n = 20,
                #d =  eff_size,
                sig.level = 0.05,
                power = 0.8,
                type = "paired")
```

```
     Paired t test power calculation

              n = 20
              d = 0.6604413
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number of *pairs*
```

> The functions returns the effect size (Cohen's metric): `d = 0.6604413`. So, with this experimental design we would be able to detect a **medium-large effect size**.

## Two Paired Samples t-test: EXAMPLE cont.

Looking instead at the **actual sample data**, what would be the observed effect size?

- To compute "observed d" we can use the function `rstatix::cohens_d`

```
d <- cortisol_data %>%
  # estimate cohen's d
  rstatix::cohens_d(cortisol ~ time, paired = TRUE)

d
```

```
# A tibble: 1 × 7
  .y.      group1  group2  effsize    n1    n2 magnitude
* <chr>    <chr>   <chr>     <dbl> <int> <int> <ord>
1 cortisol evening morning   -1.16    20    20 large
```

The obtained d (-1.16) is extremely large, so ***we likely have more participants in this study than actually needed*** given such a large effect.

## Two Paired Samples t-test: POWER ANALYSIS (n)

Let's re-compute the power analysis, but leave n as the unknown quantity, given the effect size (d) we have observed

```r
# power analysis to calculate minimunm n given the observed effect size in the sample
pwr::pwr.t.test(# n = 20,
                d =  -1.16,
                sig.level = 0.05,
                power = 0.8,
                type = "paired")
```

```
     Paired t test power calculation

              n = 7.960846
              d = 1.16
      sig.level = 0.05
          power = 0.8
    alternative = two.sided

NOTE: n is number of *pairs*
```

> As a matter of fact, would have only needed n = 8 pairs of observations in this study, given the size of effect we were trying to detect.

# One-way ANOVA test: EXE data

The `mussels_data` dataset contains information about the length of the *anterior adductor muscle scar* in the mussel `Mytilus trossulus` across five locations around the world!

```r
# Load data
mussels_data <- read.csv(file = here::here("practice", "data_input", "04_datasets",
                                           "mussels.csv"),
                         header = TRUE, # 1st line is the name of the variables
                         sep = ",", # which is the field separator character.
                         na.strings = c("?","NA" ), # specific MISSING values
                         row.names = NULL)

# Explore data
names(mussels_data)
```

```
[1] "length"   "location"
```

```r
stats <- mussels_data %>%
  dplyr::group_by (location) %>%
  dplyr::summarise (
    N = n(),
    mean_len = mean(length),
    sd_len = sd(length))

stats
```

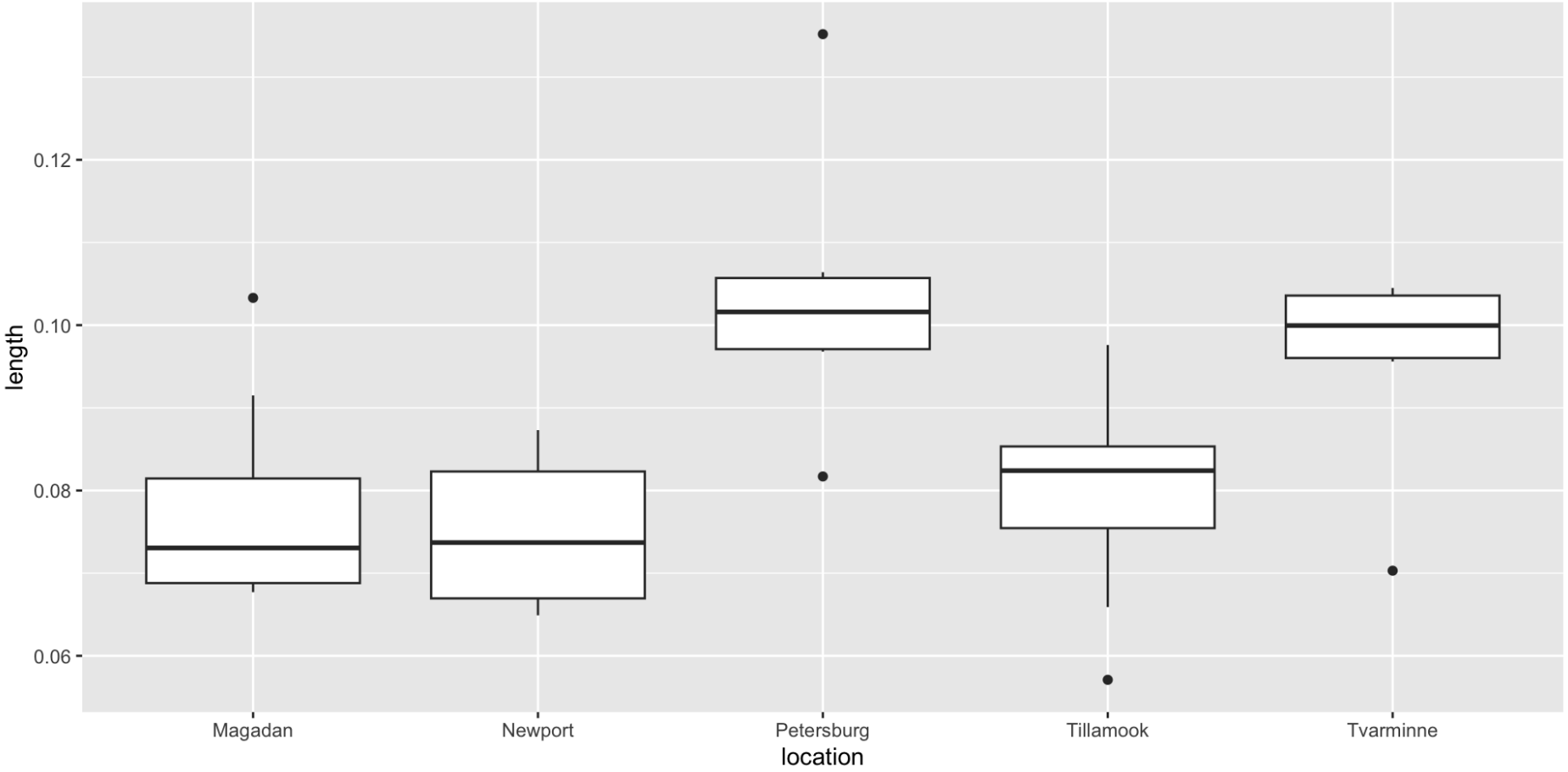
```
# A tibble: 5 × 4
  location       N mean_len  sd_len
  <chr>      <int>    <dbl>   <dbl>
1 Magadan        8   0.0780 0.0129
2 Newport        8   0.0748 0.00860
3 Petersburg     7   0.103  0.0162
4 Tillamook     10   0.0802 0.0120
5 Tvarminne      6   0.0957 0.0130
```

# One-way ANOVA test: visualization

> There appears to be a noticeable difference in lenght at average measurements *at least* between some of the locations

```
# Visualize the data with a boxplot
mussels_data %>%
  ggplot(aes(x = location, y = length)) +
  geom_boxplot()
```

# One-way ANOVA test: visualization



# One-way ANOVA test: EXAMPLE cont.

Assuming we verified the required assumptions, let's run the ANOVA test to confirm the visual intuition

- With the `stats::aov` followed by the command `summary`

```
# Summary of test outputs:
summary_ANOVA <- summary(stats::aov(length ~ location,
                    data = mussels_data))

# From which we extract all the output elements
# F value
summary_ANOVA[[1]][["F value"]] # 7.121019
```

```
[1] 7.121019       NA
```

```
# p value
summary_ANOVA[[1]][["Pr(>F)"]]  # 0.0002812242
```

```
[1] 0.0002812242           NA
```

```
# df of numerator and denominator
summary_ANOVA[[1]][["Df"]]      # 4, 34
```

```
[1]  4 34
```

```
# Sum of Square BETWEEN groups
SSB <- summary_ANOVA[[1]][["Sum Sq"]][1]  # 0.004519674
# Sum of Square WITHIN groups
SSW <- summary_ANOVA[[1]][["Sum Sq"]][2]  # 0.005394906
```

- A one-way ANOVA test confirms that **the mean lengths of muscle scar differed significantly between locations** ( F = 7.121, with df = [4, 34], and p = 0.000281).

## One-way ANOVA test: POWER ANALYSIS (`effect`)

In ANOVA it may be tricky to decide what kind of effect size we are looking for:

- if we care about an overall significance test, the sample size needed is a function of the standard deviation of the group means
- if we're interested in the comparisons of means, there are other ways of expressing the effect size (e.g. a difference between the smallest and largest means)

Here let's consider an overall test in which we could reasonably collect the same n. of observations in each group

```
n_loc <- nrow(stats)

means_by_loc <- c(0.0780, 0.0748, 0.103, 0.0802, 0.0957)
overall_mean <-  mean(means_by_loc)
sd_by_loc <- c(0.0129, 0.00860, 0.0162, 0.0120, 0.0130)
overall_sd <-  mean(sd_by_loc)
```

## One-way ANOVA test: POWER ANALYSIS (effect)

```r
# Effect Size f formula
Cohen_f = sqrt( sum( (1/n_loc) * (means_by_loc - overall_mean)^2) ) /overall_sd
Cohen_f # EXTREMELY BIG
```

```
[1] 0.877622
```

```r
# Power analysis with given f
pwr::pwr.anova.test(k = n_loc,
                    n = NULL,
                    f = Cohen_f,
                    sig.level = 0.05,
                    power = 0.80)
```

```
     Balanced one-way analysis of variance power calculation

              k = 5
              n = 4.166759
              f = 0.877622
      sig.level = 0.05
          power = 0.8

NOTE: n is number in each group
```

> The n output ( = **5 observations per group**) -as opposed to >6 per group- would be sufficient if we wanted to confidently detect the difference observed in the previous study

## Linear Regression with grouped data: EXE data

The ideas covered before apply also to **linear models**, although here:

- we use `pwr.f2.test()` to do the power calculation
- the `effect sizes` ($f^2$) is based on $R^2$

$$f^2 = \frac{R^2}{1 - R^2}$$

```r
# define the linear model
lm_mussels <- lm(length ~ location,
                 data = mussels_data)
```

```r
# summarise the model
summary(lm_mussels)
```

# Linear Regression with grouped data: EXE data

```
Call:
lm(formula = length ~ location, data = mussels_data)

Residuals:
      Min        1Q    Median        3Q       Max
-0.025400 -0.007956  0.000100  0.007000  0.031757

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         0.078012   0.004454  17.517  < 2e-16 ***
locationNewport    -0.003213   0.006298  -0.510  0.61331
locationPetersburg  0.025430   0.006519   3.901  0.00043 ***
locationTillamook   0.002187   0.005975   0.366  0.71656
locationTvarminne   0.017687   0.006803   2.600  0.01370 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.0126 on 34 degrees of freedom
Multiple R-squared:  0.4559,    Adjusted R-squared:  0.3918
```

## Linear Regression with grouped data: POWER ANALYSIS

From the linear model we get that the $R^2$ value is 0.4559 and we can use this to calculate Cohen's $f^2$ value using the formula

```
f2 <- 0.4559 / (1 - 0.4559)
f2
```

```
[1] 0.8378974
```

Our model has 5 parameters (because we have 5 groups) and so the numerator degrees of freedom $u$ will be 4 (5−1=4).

Hence, we carry out the power analysis with the function `pwr.f2.test`:

```
pwr::pwr.f2.test(u = 4, v = NULL,
                 f2 = 0.8378974,
                 sig.level = 0.05 , power = 0.8)
```

## Linear Regression with grouped data: POWER ANALYSIS interpret

Recall that, in the F statistic evaluating the model,

- **u** the df for the numerator: $\mathrm{df}_{\text{between}} = k - 1 = 5 - 1 = 4$
- **v** the df for the denominator: $\mathrm{df}_{\text{within}} = n - k = ?$
  - so $n = v + 5$

```
pwr::pwr.f2.test(u = 4, f2 = 0.8378974,
                 sig.level = 0.05 , power = 0.8)
```

```
Multiple regression power calculation

          u = 4
          v = 14.62182
         f2 = 0.8378974
  sig.level = 0.05
      power = 0.8
```

> This tells us that the denominator degrees of freedom **v** should be 15 (14.62 rounded up), and this means that we would only need 20 observations **n = v+5** in total across all 5 groups to detect this effect size

# SAMPLE SPLITTING IN MACHINE LEARNING

> Embracing a different philosophical approach...

## 2 different approaches with different takes on empirical data

*(Simplifying a little)*

### Inferential statistics

- `GOAL`: Convincingly explain
- `APPROACH`: Strong emphasis on defining assumptions (about variables distributions) and/or hypotheses on the relationship between them
- `DATA`:
  - The **collection strategy is designed *ex-ante*** , according to the experiment goal
  - Usually, ALL AVAILABLE DATA are used to estimate effect of interest (as `sampling` was designed to be representative of a population).

### Machine Learning

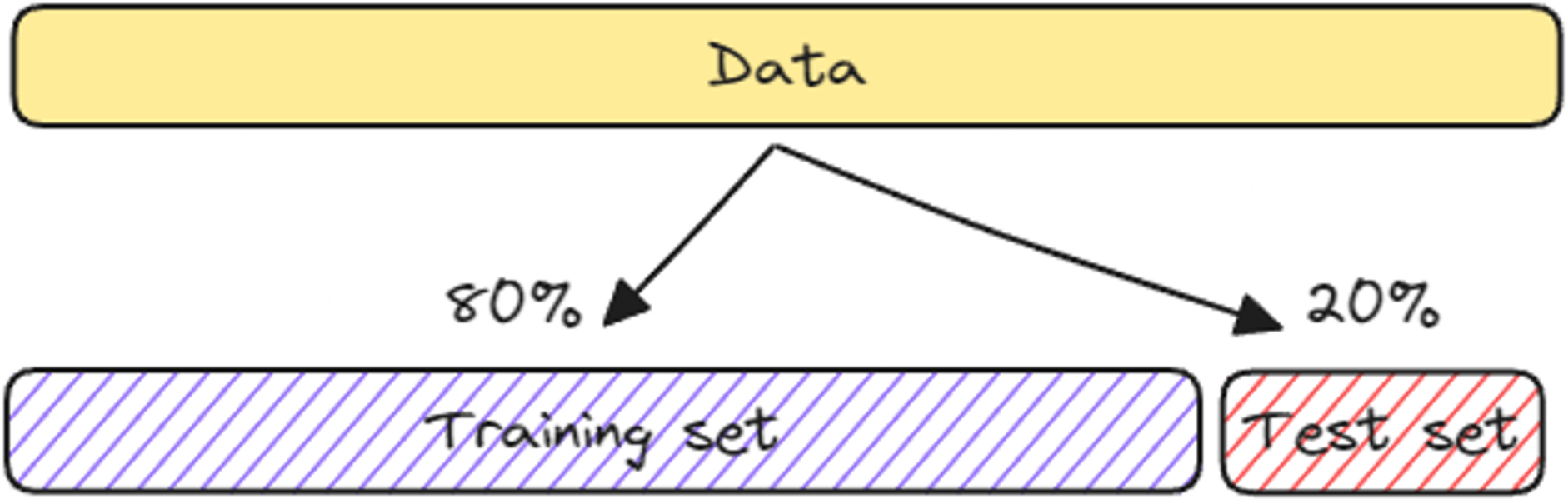
- `GOAL`: Accurately predict
- `APPROACH`: Focus on labeling observations or uncovering ("learn") a pattern, without worrying about explaining them
- `DATA`:
  - Data drives the search for patterns, but there is a huge risk of *"overfitting"* models (too specific to initial data!)
  - It is critical to SPLIT THE DATA (usually 75% for `training` and 25% for `testing` the algorithms) **leaving aside a sub-sample to test the model** with unseen new data

# Data Splitting in ML approaches

Consistent with the ML approach (**learning from (data) examples**), it is critical to split the available data to obtain:

1. 60-80% ➜ `training sample` for *fitting a model* and making prediction on the training data itself
2. 20-40% ➜ `testing sample` for *evaluating the performance* of the selected model(s) and test it works on new data too

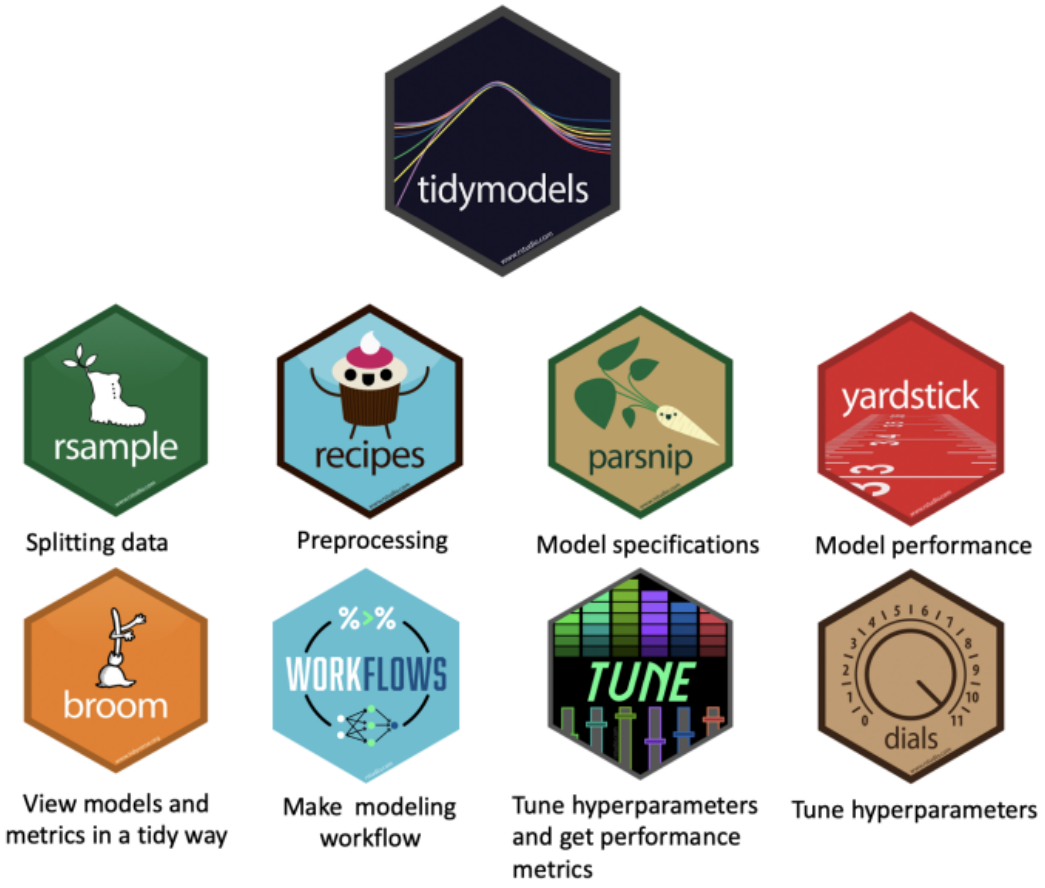
- *Since in ML we don't claim to know what works in advance, it is essential to "test" a candidate predictive model on fresh new data and see if it holds*



## Introducing R (metapackage) `tidymodels` for modeling and ML

The package `tidymodels` (much like the `tidyverse`) is an ecosystem of packages meant to enable a wide variety of approaches for modeling and statistical analysis.

- One package in this system is `rsample` is one of its building blocks for resampling data



## Revisiting NHANES for a quick demonstration of predictive modeling

Let's re-load a dataset from Lab # 3 (the NHANES dataset) for a quick demonstration of data splitting in an ML predictive modeling scenario

- We can try predicting `BMI` from `age` (in years), `PhysActive`, and `gender`, using linear regression model (which is a `Supervised ML algorithm`)
- (we already saved this dataset)

```r
# (we already saved this dataset in our project folders)

# Use `here` in specifying all the subfolders AFTER the working directory
nhanes <- read.csv(file = here::here("practice", "data_input", "03_datasets",
                                     "nhanes.samp.csv"),
                           header = TRUE, # 1st line is the name of the variables
                           sep = ",", # which is the field separator character.
                           na.strings = c("?","NA" ), # specific MISSING values
                           row.names = NULL)
```

# Splitting the dataset into training and testing samples

- With this approach, it is best practice to **"hold back" some data for testing** to get a better estimate of how models will perform on new data
- We can easily specify training and testing sets using `rsample`'s function `initial_split`

```
nhanes_split <- nhanes %>%
  # define the training proportion as 75%
  rsample::initial_split(prop = 0.75,
  # ensuring both sets are balanced in gender
                         strata = Gender)

nhanes_train <- rsample::training(nhanes_split)
dim(nhanes_train)
```

```
[1] 374  77
```

```
nhanes_test <- rsample::testing(nhanes_split)
dim(nhanes_test)
```

```
[1] 126  77
```

## Fitting a linear model on the training data

In this case the **regression models** serves for predicting numeric, continuous quantities

```r
# fitting  linear regression model specification
lin_mod <- lm(BMI ~ Age + Gender + PhysActive, data = nhanes_train)

summary(lin_mod)
```

```
Call:
lm(formula = BMI ~ Age + Gender + PhysActive, data = nhanes_train)

Residuals:
    Min      1Q  Median      3Q     Max
-13.031  -4.657  -1.152   4.024  37.691

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  30.54367    1.27063  24.038  < 2e-16 ***
Age           0.01199    0.02133   0.562  0.57445
Gendermale   -0.96772    0.73068  -1.324  0.18619
PhysActiveYes -2.04670    0.73970  -2.767  0.00594 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.94 on 368 degrees of freedom
  (2 observations deleted due to missingness)
Multiple R-squared:  0.03158,   Adjusted R-squared:  0.02368
F-statistic:     4 on 3 and 368 DF,  p-value: 0.008011
```

## Predicting BMI estimates for new data set

Using the above model, we can predict the BMI for different individuals (those left in the testing data)

- with the function `predict`, where we specify the argument `newdata = nhanes_test`)
- adding the prediction `interval` (the 95% CI), which gives uncertainty around a single value of the prediction

```r
# Obtain predictions from the results of a model fitting function
pred_bmi <- stats::predict(lin_mod,
                newdata = nhanes_test,
                interval = "confidence" )
head(pred_bmi)
```

```
       fit      lwr      upr
1 29.06036 27.77712 30.34359
2 27.86489 26.40850 29.32128
3 29.09632 27.80792 30.38471
4 27.99674 26.74223 29.25125
5 30.46299 28.88405 32.04193
6 29.19221 27.83200 30.55242
```

## Evaluating the predictive performance in testing data

The ultimate goal of holding data back from the model training process was to **evaluate its predictive performance on new data**.

A common measure used is the `RMSE (Root Mean Square Error)` = a measure of the distance between observed values and predicted values **in the testing dataset**

```
# Computing the Root Mean Square Error

RMSE_test <- sqrt(mean((nhanes_test$BMI - predict(lin_mod, nhanes_test))^2, na.rm = T))
RMSE_test # 6.579572
```

```
[1] 6.069719
```

> The RMSE (= 6.579572) tells us, (roughly speaking) by how much, on average, the new observed BMI values differ from those predicted by our model

# ... and what about in trainig data?

Let's see the RMSE in the training dataset (for comparison)

```
RMSE_train <- sqrt(mean((nhanes_train$BMI - predict(lin_mod, nhanes_train))^2, na.rm = T))
RMSE_train # 6.736376
```

```
[1] 6.902941
```

```
# R squared is also quite low
summary(lin_mod)$r.squared     # R^2 0.03374094
```

```
[1] 0.03157804
```

> This is not what expected 🤔, since RMSE on the training data is sliglthly bigger that in the testing data!

A possible explanation is that out model is `underfitting` in the first place (model's $R^2$ was quite low too), so we should definitely try different models...

# WRAPPING UP TODAY'S KEY MESSAGE

# Recap of the workshop's content

## TOPICS WE COVERED

1. Motivated the choice of learning/using **R for scientific quantitative analysis**, and lay out some fundamental concepts in biostatistics with concrete R coding examples.
2. Consolidated understanding of **inferential statistic**, through R coding examples conducted on real biostatistics research data.
3. Discussed the **relationship between any two variables**, and introduce a widely used analytical tool: **regression**.
4. Presented a popular ML technique for dimensionality reduction (**PCA**), performed both with `MetaboAnalyst` and `R`.
5. Introduction to **power analysis** to define the correct sample size for hypotheses testing and discussion of how ML approaches deal with available data.

# Final thoughts

- While the workshop only allowed for a synthetic overview of fundamental ideas, it hopefully provided a solid foundation on the most common statistical analysis you will likely run in your daily work:
  - Thorough **understanding of the input data** and the data collection process
  - Univariate and bivariate **exploratory analysis** (accompanied by visual intuition) to form hypothesis
  - Upon verifying the assumptions, we **fit data** to hypothesized model(s)
  - **Assessment of the model performance** ($R^2$, $\text{Adj.}\, R^2$, $\text{F} - \text{Statistic}$, etc.)
- You should now have a solid grasp on the R language to keep using and exploring the huge potential of this programming ecosystem
- We only scratched the surface in terms of ML classification and prediction models, but we got a hang of the **fundamental steps** and some **useful tools** that might serve us also in more advanced analysis