Assessing the effect of an exposure on multiple outcomes (with **R** code)

Brice Ozenne

February 19, 2019

Summary

We propose a strategy to assess the effect of an exposure (e.g. a disease, a genetic factor) on several outcomes (e.g. psychological outcomes, the binding potential measured in several brain regions) while accounting for possible risk factors and confounders. This strategy, called *multiple univariage regressions* strategy, models the relationship between each outcome and the exposure using a separate model. Once the models have been correctly fitted, a global test can be used to test whether there is any effect of the exposure on the outcomes. After that, multiple tests are performed to test outcome-specific effects of the exposure where the Dunnett adjustment is used to control the type 1 error (Pipper et al., 2012). An adjustment is used to improved the control of the type 1 error in small sample sizes (e.g. n<100). This adjustment has been shown to beneficial in several settings (using simulation studies) but does not always perfectly control the type 1 error rate. It is advised to check that validity of the adjustment when using very small samples or models with many parameters.

The proposed strategy can be used with any type of outcomes for which a can fit a model with asymptotically linear estimators (Tsiatis (2007), section 3). This includes generalized linear model or Cox models. It makes it very flexible and the strategy makes only few assumptions on the joint distribution. The drawback is that it is not the most efficient approach. For instance modelling the joint distribution of the outcomes, e.g. using a latent variable model / mixed model in the case of normally distributed outcomes, will be a more efficient strategy. Another limitation is that with the proposed approach a treatment effect specific to each outcome will be estimated, while in some context the investigator may want to constrain the treatment effect to be the same for some outcomes. Finally, to be feasible the strategy requires the number of outcomes to be not too large (<100) and smaller than the number of observations (low-dimensional setting).

This document we aim at giving a basic understanding of the strategy and how to implement it. In particular, we don't claim that the proposed strategy is valid or optimal results in every application (it is probably not the case) and many other approaches exists but won't be discussed here. The document is organized as follow: first section 1 present how to install and set up the **R** software. Then basic functions to import and process the data are presented in section 2. The goals, a minima, of the descriptive analysis are presented in section 3, as well as code to export figures and tables to Word documents. Section 4 recalls how to perform a univariate linear regression in **R** and present tools to assess the parametric assumptions and section 5 presents the *multiple univariate regressions* strategy. Finally appendix A recalls some statistical concepts about modeling and hypothesis testing.

1 Software

We advise to use the **R** software to implement these strategies. It can be downloaded at https://cloud.r-project.org/. R studio provide a convenient user interface that can be downloaded at https://www.rstudio.com/products/rstudio/. The **R** code used to carry out the two strategies will be display in boxes:

```
1+1 ## comment about the code
```

```
[1] 2
```

while the **R** output will be displayed in dark red below the box.

When starting a fresh **R** session, only the core functionalities of **R** are available. Additional functionalities called packages can be downloaded from the CRAN using the command `install.packages`. We will need the following packages:

```
install.packages(pkgs = c("lava","lavaSearch2","multcomp","qqtest","gof","reshape2"))
```

After having installed the packages, one needs to load them using the command `library` to use them in the current **R** session:

```
library(lava) ## definition and estimation of latent variable models
library(lavaSearch2) ## small sample correction
library(multcomp) ## adjustment for multiple comparisons
library(qqtest) ## diagnostics: qqplots
library(gof) ## diagnostics: linearity assumption
library(reshape2) ## data management
```

I also recommend the following packages:

- *data.table*: for data management. See https://cran.r-project.org/web/packages/data.table/vignettes/datatable-intro.html for an introduction and https://github.com/Rdatatable/data.table/wiki for more documentation. A synthetic description of the functionalities can be found at https://s3.amazonaws.com/assets.datacamp.com/img/blog/data+table+cheat+sheet.pdf
- *ggplot2*: for data visualization. See http://r4ds.had.co.nz/data-visualisation.html for an introduction. A synthetic description of the functionalities can be found at https://www.rstudio.com/wp-content/uploads/2015/03/ggplot2-cheatsheet.pdf

2 Data management

2.1 Working directory

The working directory is where R, by default, look for files to import and export data or figures. The current working directory can be accessed using:

```
getwd()
```

```
[1] "c:/Users/hpl802/AppData/Roaming/R"
```

It can be changed using the function `setwd()`:

```
path <- "c:/"
setwd(path)
```

We can check that the working directory has indeed changed calling again `getwd()`:

```
getwd()
```

```
[1] "c:/"
```

2.2 Importing the data

It is a good idea to start by checking that the working directory contains the data we want to import. For instance the file `data.csv` is storing the data, we can use:

```
file.exists("data.csv")
```

```
[1] TRUE
```

We can also list all files in the current directory with a `.csv` extension using:

```
list.files(pattern = ".csv")
```

```
[1] "data.csv"
```

We can also display the first lines of the file using:

```
readLines("data.csv")[1:3]
```

```
[1] "\"Id\",\"Gender\",\"age\",\"BMI\",\"MDI\",\"Y1\",\"Y2\",\"Y3\",\"Y4\",\"Y5\""
[2] "\"Subj1\",\"female\",30.57,21.76,25.82,7.64,8.73,7.72,10.42,8.44"
[3] "\"Subj2\",\"female\",41.36,25.55,12.38,7.11,8.79,6.99,8.45,8.26"
```

We can see that the columns are separated with `,` and that the `.` indicates the decimal values. Moreover the words such as the columns names or the subject identities are surrounded by `\"` (e.g. `\"Id\"` stand for Id). Finally in this example there is no missing values but if there was it is important to know how they are encoded.

The command to import the data depends on the type of file. Here for a `.csv` file we use `read.csv`. Luckily the default arguments `sep`, `dec`, `quote` are correctly specified:

```
args(read.csv)
```

```
function (file, header = TRUE, sep = ",", quote = "\"", dec = ".",
    fill = TRUE, comment.char = "", ...)
NULL
```

The argument `header` set to `TRUE` indicates that the first line of the dataset contains the column names (and not the actual data). The `...` indicates there are additional arguments that are not shown here (see the documentation using `help(read.csv)`). For instance, in presence of missing values, one would need to specify the argument `na.string`. Here it is sufficient to do:

```
dfW <- read.csv("data.csv")
```

Other functions exists to import other types of data, e.g. `read.table` for `.txt` files, `read.xlsx` from the xlsx package for `.xlsx` file, or `read.spss` from the foreign package for spss data files. One should always inspect if R has correctly imported the data, e.g. using:

```
str(dfW)
```

```
'data.frame':        50 obs. of  10 variables:
 $ Id    : Factor w/ 50 levels "Subj1","Subj10",..: 1 12 23 34 45 47 48 49 50 2 ...
 $ Gender: Factor w/ 2 levels "female","male": 1 1 2 1 1 1 2 1 2 2 ...
 $ age   : num  30.6 41.4 27 40.6 45.8 ...
 $ BMI   : num  21.8 25.6 28.6 23.2 19.8 ...
 $ MDI   : num  25.82 12.38 7.41 16.46 18.56 ...
 $ Y1    : num  7.64 7.11 7.88 8.99 7.6 6.99 3.76 6.94 6.57 6.89 ...
 $ Y2    : num  8.73 8.79 9.89 14.38 8.77 ...
 $ Y3    : num  7.72 6.99 13.51 13.82 8.38 ...
 $ Y4    : num  10.42 8.45 10.79 11.44 7.94 ...
 $ Y5    : num  8.44 8.26 7.9 9.75 6.17 8.78 2.41 5.38 5.04 5.22 ...
```

In this example, the two columns contain character strings (`Factor` is a type of character strings in R) and the rest contains numerical values.

2.3 Data processing

Often the raw data needs to be transformed before being analyzed:

- A typical example is when one need to deal with the variable:

```
gender <- c(1,0,1,0,1) ## what is 1? what is 0?
```

This is already better:

```
female <- c(1,0,1,0,1) ## we can guess that 1: female and 0: male
```

but it is a good practice in such situation to rename the actual values into something understandable:

```
factor(gender, levels = 0:1, labels = c("Female","Male"))
```

```
[1] Male   Female Male   Female Male
Levels: Female Male
```

- With repeated measurements per individual, one often needs to reshape his dataset from the wide format (one line per individual) to the long format (one line per measurement). This can be done using the `melt` method:

```
dfL <- melt(dfW,
            id.vars = c("Id","Gender","age","BMI","MDI"),
            value.name = "score",
            variable.name = "outcome")
head(dfL)
```

```
    Id Gender   age   BMI   MDI outcome score
1 Subj1 female 30.57 21.76 25.82      Y1  7.64
2 Subj2 female 41.36 25.55 12.38      Y1  7.11
3 Subj3   male 26.97 28.56  7.41      Y1  7.88
4 Subj4 female 40.61 23.22 16.46      Y1  8.99
5 Subj5 female 45.79 19.78 18.56      Y1  7.60
6 Subj6 female 37.14 16.13 17.82      Y1  6.99
```

The opposite operation can be performed using `dcast`.

- It is often a good idea to restrict the dataset to the relevant variables (e.g. remove genetic data if they are not of interest). It is easier to work with and to display in the next steps. This can for instance be done by defining the variables of interest:

```
keep.var <- c("Id","BMI","MDI","Y1","Y2","Y3","Y4","Y5")
```

We can check that the variables defined in `keep.var` are in `dfW`:

```
keep.var %in% names(dfW)
```

```
[1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

and then subset the initial dataset:

```
dfW.red <- dfW[,keep.var]
head(dfW.red)
```

```
    Id   BMI   MDI   Y1    Y2    Y3    Y4   Y5
1 Subj1 21.76 25.82 7.64  8.73  7.72 10.42 8.44
2 Subj2 25.55 12.38 7.11  8.79  6.99  8.45 8.26
3 Subj3 28.56  7.41 7.88  9.89 13.51 10.79 7.90
4 Subj4 23.22 16.46 8.99 14.38 13.82 11.44 9.75
5 Subj5 19.78 18.56 7.60  8.77  8.38  7.94 6.17
6 Subj6 16.13 17.82 6.99  9.97  6.74  8.29 8.78
```

- Often after having imported the data we want to change its column names. First we need to know the current column names. The `names` function can be used to output all the column names:

```
names(dfW.red)
```

```
[1] "Id"  "BMI" "MDI" "Y1"  "Y2"  "Y3"  "Y4"  "Y5"
```

Alternatively the `grep` function will output any column name containing a given string of characters:

```
grep(pattern = "Y", x = names(dfW), value = TRUE)
```

```
[1] "Y1" "Y2" "Y3" "Y4" "Y5"
```

Then, we can rename a specific column using:

```
names(dfW.red)[names(dfW.red) == "Id"] <- "new.Id.name"
head(dfW.red)
```

```
  new.Id.name   BMI   MDI   Y1    Y2    Y3    Y4   Y5
1       Subj1 21.76 25.82 7.64  8.73  7.72 10.42 8.44
2       Subj2 25.55 12.38 7.11  8.79  6.99  8.45 8.26
3       Subj3 28.56  7.41 7.88  9.89 13.51 10.79 7.90
4       Subj4 23.22 16.46 8.99 14.38 13.82 11.44 9.75
5       Subj5 19.78 18.56 7.60  8.77  8.38  7.94 6.17
6       Subj6 16.13 17.82 6.99  9.97  6.74  8.29 8.78
```

To rename several columns at the same time we can use:

```
old2new <- c("Y1" = "y1", "Y2" = "y2", "Y3" = "y3")
names(dfW.red)[match(names(old2new),names(dfW.red))] <- old2new
head(dfW.red)
```

```
  new.Id.name   BMI   MDI   y1    y2    y3    Y4   Y5
1       Subj1 21.76 25.82 7.64  8.73  7.72 10.42 8.44
2       Subj2 25.55 12.38 7.11  8.79  6.99  8.45 8.26
3       Subj3 28.56  7.41 7.88  9.89 13.51 10.79 7.90
4       Subj4 23.22 16.46 8.99 14.38 13.82 11.44 9.75
5       Subj5 19.78 18.56 7.60  8.77  8.38  7.94 6.17
6       Subj6 16.13 17.82 6.99  9.97  6.74  8.29 8.78
```

Other useful functions are `tolower` to convert characters to lower case and `gsub` to remove a specific pattern in a character vector, e.g.:

```
gsub(pattern = ".", replacement = "", x = c("a..","b..."), fixed = TRUE)
```

```
[1] "a" "b"
```

Many of the other data processing steps are specific to each study and we won't discuss them in this document.

3 Descriptive statistics

Before doing any analysis, it is a good practice to describe the data that are to be analyzed. The has several aims:

- **check that that database contains the population of interest**, i.e. individuals in the database are indeed those the we want to study and we have all of them.
- **check that the collected values are plausible**, e.g. if the inclusion criteria include that the age range is between 18 and 99 years, then one should check that this is indeed the case.
- **check that the collected values are coded as expected**, e.g. age is usually coded in years (and not in months).
- **check that the collected values are distributed as expected**, e.g. is there missing values? Are the values uniformly spread? Bimodal? Concentrated at low or high values?

Note: one should checks that for all the variables of interest. This can appear time-consuming but can really save you time at latter stages.

- **produce your table 1** i.e. a descriptive table of your cohort that is almost always included in an article. You can for instance use the function `univariateTable` from the Publish package:

```
library(Publish)
myTable1 <- univariateTable(Gender ~ age + BMI + MDI + Y1 + Y2 + Y3 + Y4 + Y5,
                            data = dfW)
myTable1
```

```
  Variable     Level female (n=30) male (n=20) Total (n=50) p-value
1      age mean (sd)    36.2 (5.8)  34.6 (5.0)   35.6 (5.5) 0.31204
2      BMI mean (sd)    21.5 (3.3)  23.1 (3.2)   22.2 (3.4) 0.09325
3      MDI mean (sd)    19.2 (5.8)  19.2 (5.7)   19.2 (5.7) 0.97596
4       Y1 mean (sd)     7.2 (1.6)   7.2 (2.0)    7.2 (1.8) 0.93155
5       Y2 mean (sd)     9.6 (2.9)   9.5 (2.1)    9.6 (2.6) 0.82411
6       Y3 mean (sd)     8.5 (3.4)   8.4 (3.2)    8.4 (3.3) 0.91260
7       Y4 mean (sd)     8.8 (2.1)   9.3 (1.7)    9.0 (1.9) 0.39083
8       Y5 mean (sd)     7.4 (2.9)   7.0 (2.9)    7.2 (2.9) 0.65171
```

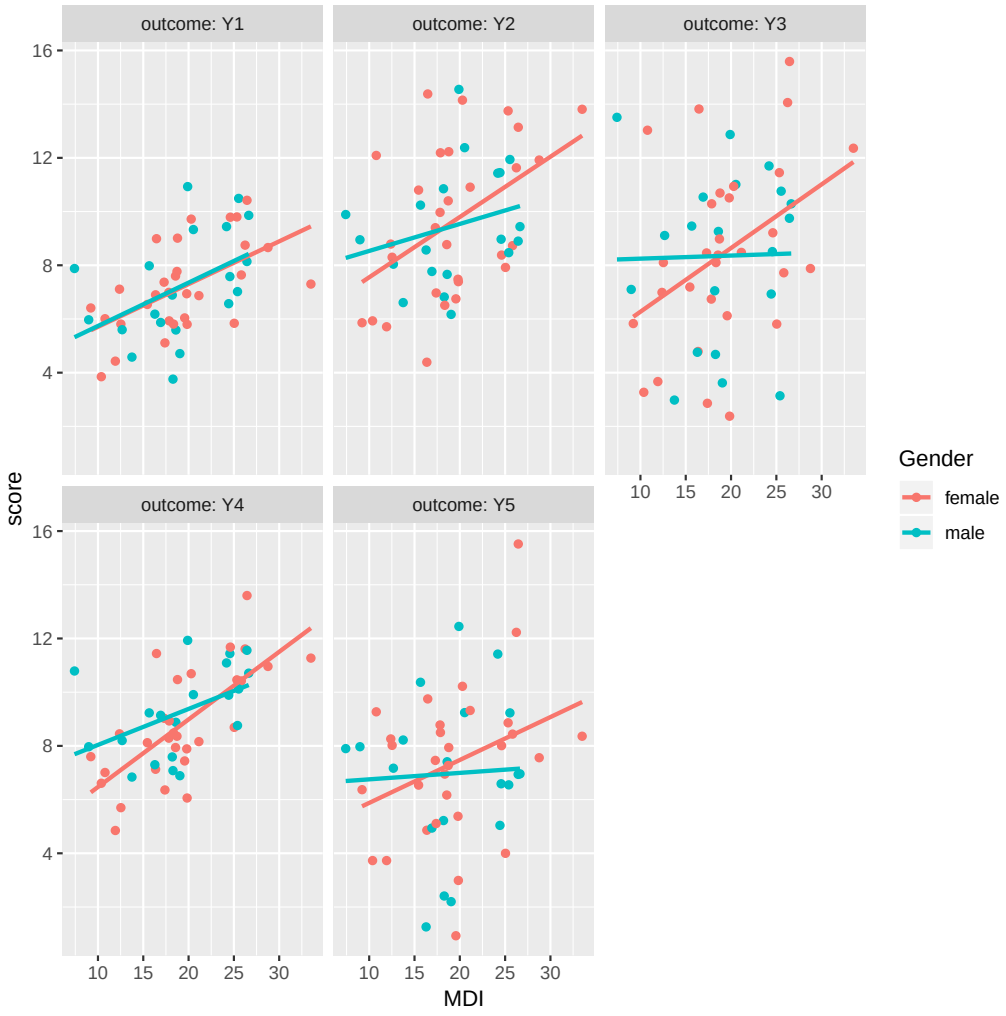
You can also export this table in a word document with the package officer:

```
library(officer)
myTable1.doc <- body_add_table(x = read_docx(),
                               value =  summary(myTable1))
print(myTable1.doc, target = "./Table1.docx")
```

To keep the code simple, we only present here a very basic application of these tools. More complex tables with a nicer display in word can be obtain with a bit of coding.

- **make synthetic representations of your data** using graphs or images. This can be useful to visualize your data and help your collaborators to understand what you have collected or what you are trying to show.

```
library(ggplot2)
gg <- ggplot(dfL, aes(x = MDI, y = score, color = Gender, group = Gender))
gg <- gg + geom_point()
gg <- gg + facet_wrap(~outcome, labeller = label_both)
gg <- gg + geom_smooth(method = "lm", se = FALSE)
gg
```



You can then export the figure using:

```
pdf("./figures/descriptive.pdf")
gg
dev.off()
```

4 Univariate analysis using a univariate linear regression

Imagine we want to assess the effect of MDI on Y_1 adjusting for age and BMI using a univariate linear regression. Mathematically the model can be written:

$$Y_1 = \alpha + \beta_{age} age + \beta_{BMI} BMI + \beta_{MDI} MDI + \varepsilon \tag{1}$$

where ε are the residuals. that are assumed to be:

- A0: independent and identically distributed (iid)
- A1: normally distributed.

Note that for equation (1) to be valid we assume:

- A2: linear effect of the covariates (e.g. no interaction)

(A0-A2) are modeling assumptions and only (A1-A2) can be tested in practice. Univariate linear regression are also not recommended in presence of extreme values (A3) or very correlated covariates (A4).

4.1 Fitting a univariate linear regression in R

We can use:

```
e.lm <- lm(Y1 ~ age + BMI + MDI, data = dfW)
```

We can extract the value of the model coefficients using `coef`:

```
coef(e.lm)
```

```
 (Intercept)          age          BMI          MDI
-1.413215636  0.006305252  0.247124506  0.151044284
```

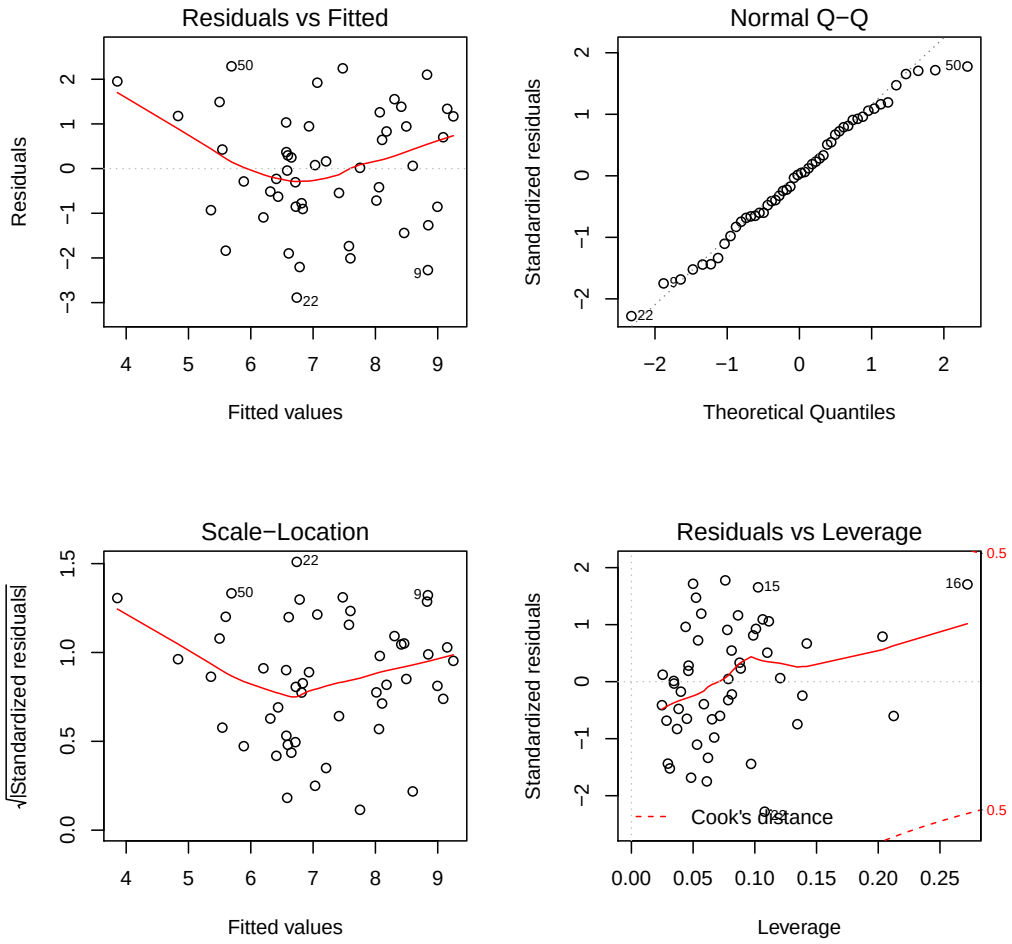
4.2 Interpretation of the regression coefficients

If the assumptions (A0-A2) hold we can interpret β_{MDI} as a correlation coefficient. This means that for fixed age and BMI, if we observe an individual A with value of MDI higher by one unit compared to individuals B then we would also expect that its value for Y_1 differ by β_{MDI} compared the other individual. If we in addition make causal assumptions (mainly no unobserved confounder) then we can interpret β_{MDI} as the effect of MDI on the outcome. This means that if we could change the MDI of an individual by one unit then its variation in outcome should be β_{MDI}.

4.3 Diagnostics tools for univariate linear regression in R

R provides a graphical display that giving an overview of the model fit:

```
par(mfrow = c(2,2))
plot(e.lm)
```



The top left plot is useful to detect a misspecification of the linear predictor (e.g. a U shape would indicate a missing quadratic effect). The top right plot enable to check the normality of the residuals, we will describe a more informative qqplot below. The bottom left can be used to detect heteroschedasticity (e.g. a trumpet shape) and the bottom right plot can be used to identify observation that have a huge influence on the fitted values.

4.3.1 Testing (A1)

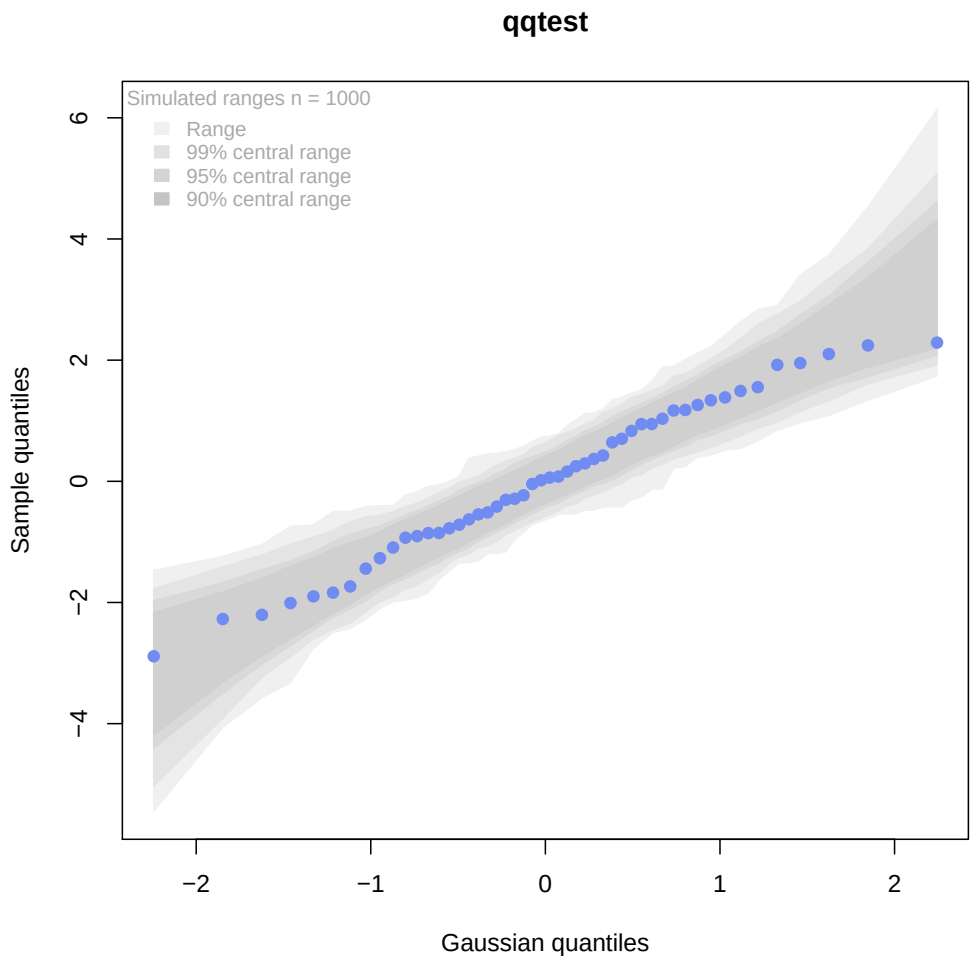
The qqtest package provides a more readable qqplot. To use it, we first need to extract the residuals. This can be achieved using the `residuals` method:

```
dfW$resid.lm <- residuals(e.lm, type = "response")
```

The `type` argument indicates the type of residuals we want to extract. Raw residuals are $\hat{\varepsilon} = Y - \hat{Y}$, i.e. the observed minus the fitted values. In models more complex than a univariate linear regression, the raw residuals may not be iid. This makes it difficult to assess the validity of the assumptions. In such cases we display instead diagnostics for normalized residuals that, if the assumptions of the model are correct, should follow a standard normal distribution.

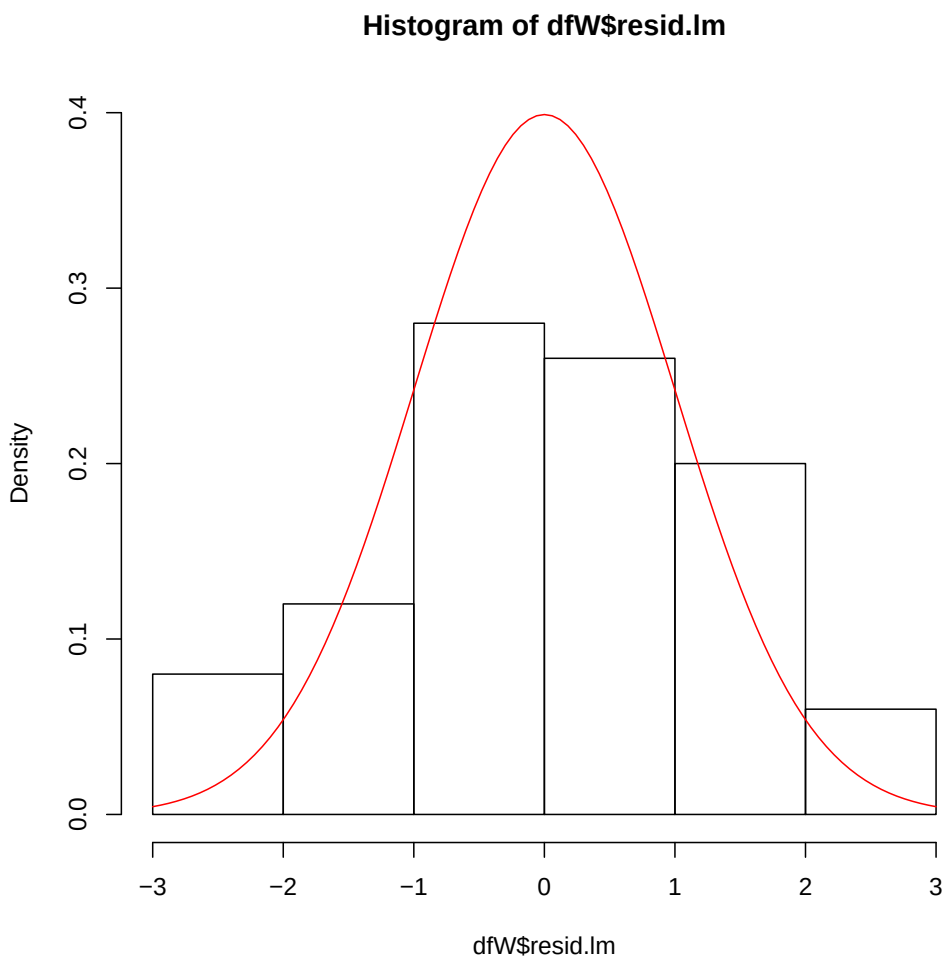
Having extracted the residuals, we can then obtain the qqplot using the `qqtest` function:

```
qqtest(dfW$resid.lm)
```



The shaded area indicates where, if the normality assumption was correct, we would expect to observe the points. Alternatively, an histogram of the residuals can be used to assess the normality of the residuals:

```
hist(dfW$resid.lm, prob=TRUE, ylim = c(0,0.4))
curve(dnorm(x, mean=0, sd=1), add=TRUE, col = "red")
```



Statistical tests can also be used to assess deviation from normality:

```
shapiro.test(dfW$resid.lm)
```

```
        Shapiro-Wilk normality test

data:  dfW$resid.lm
W = 0.98104, p-value = 0.5967
```

Here the null hypothesis is that the residuals follow a normal distribution.

4.3.2 Testing (A2)

A statistical test can also be used to assess whether there is evidence for a more complex functional form for the linear predictor:

```
cumres(e.lm)
```

```
Kolmogorov-Smirnov-test: p-value=0.195
Cramer von Mises-test: p-value=0.06
Based on 1000 realizations. Cumulated residuals ordered by predicted-variable.
---
Kolmogorov-Smirnov-test: p-value=0.016
Cramer von Mises-test: p-value=0.021
Based on 1000 realizations. Cumulated residuals ordered by age-variable.
---
Kolmogorov-Smirnov-test: p-value=0.151
Cramer von Mises-test: p-value=0.29
Based on 1000 realizations. Cumulated residuals ordered by BMI-variable.
---
Kolmogorov-Smirnov-test: p-value=0.708
Cramer von Mises-test: p-value=0.833
Based on 1000 realizations. Cumulated residuals ordered by MDI-variable.
---
```

4.3.3 Testing (A3)

The `influence` method can be used to output what is the impact of each observation on each estimated parameter:

```
head(influence(e.lm)$coefficient)
```

```
  (Intercept)           age           BMI           MDI
1 -0.06431419  0.0021223892  0.0011037299 -0.0023268582
2 -0.02333311  0.0005276379  0.0007260234 -0.0005075087
3  0.02225432 -0.0037807751  0.0134437130 -0.0084481492
4 -0.31091908  0.0087011324  0.0065717767 -0.0053971745
5 -0.16703438  0.0080674308 -0.0026717037 -0.0019605965
6  0.43406252  0.0002100576 -0.0176591598 -0.0008988258
```

Large values (positive or negative) indicate influential observations.

4.3.4 Testing (A4)

The correlation among the explanatory variables can be assessed using the VIF (variance inflation factor):

```
car::vif(e.lm)
```

```
     age      BMI      MDI
1.076066 1.034264 1.051645
```

Values higher than 5 are considered as high (the threshold of 5 is arbitrary).

4.4 Hypothesis testing

We want to formally test whether there is an effect of MDI on the outcome. This is equivalent to test the null hypothesis:

$$(\mathcal{H}_0)\ \beta_{MDI,0} = 0$$

Since the parameters are estimated by ML and assuming that the model is correctly specified, we know that the asymptotic distribution of the parameter is Gaussian. This means that for large sample size, the fluctuation of the estimated values follows a normal distribution. For instance:

$$\hat{\beta} \underset{n\to\infty}{\sim} \mathcal{N}\left(\beta, \sigma^2_\beta\right)$$

where σ^2_β is the variance of the MLE, i.e. the uncertainty surrounding our estimation of the association. It follows that:

$$t_\beta = \frac{\hat{\beta} - \beta_0}{\sigma^2_\beta} \underset{n\to\infty}{\sim} \mathcal{N}(0,1) \tag{2}$$

So under the null hypothesis of no association between the outcome and the exposure the statistic t_β should follow a standard normal distribution. Very low or very large values are unlikely to be observed and would indicate that the null hypothesis does not hold. This is called a (univariate) Wald test. The result of this tests can be obtained using the `summary` method [1]:

```
summary(e.lm)$coef
```

```
               Estimate Std. Error    t value     Pr(>|t|)
(Intercept) -1.413215636 1.95732599 -0.7220134 4.739407e-01
age          0.006305252 0.03613264  0.1745030 8.622360e-01
BMI          0.247124506 0.05790521  4.2677422 9.756957e-05
MDI          0.151044284 0.03441289  4.3891775 6.598863e-05
```

95% confidence intervals for the model parameters can then be obtained using the `confint` method:

```
confint(e.lm)
```

```
                 2.5 %     97.5 %
(Intercept) -5.35310851 2.52667723
age         -0.06642598 0.07903648
BMI          0.13056736 0.36368165
MDI          0.08177473 0.22031384
```

[1] In reality R is automatically performing a correction that improves the control of the type 1 error. Indeed we usually don't know σ^2_β and plugging-in its estimate in equation (2) modifies the distribution of t_β in small samples. The correction uses a Student's t distribution instead of a Gaussian distribution.

5 Multivariate analysis using multiple univariate linear regressions

We now want to simultaneously test the effect of MDI on all the five outcomes. To achieve it, we fit separately for each outcome a univariate linear regression. Mathematically the model can be written:

$$\begin{bmatrix} Y_1 & = \alpha_{Y_1} + \beta_{Y_1,age} age + \beta_{Y_1,BMI} BMI + \beta_{Y_1,MDI} MDI + \varepsilon_{Y_1} \\ Y_2 & = \alpha_{Y_2} + \beta_{Y_2,age} age + \beta_{Y_2,BMI} BMI + \beta_{Y_2,MDI} MDI + \varepsilon_{Y_2} \\ Y_3 & = \alpha_{Y_3} + \beta_{Y_3,age} age + \beta_{Y_3,BMI} BMI + \beta_{Y_3,MDI} MDI + \varepsilon_{Y_3} \\ Y_4 & = \alpha_{Y_4} + \beta_{Y_4,age} age + \beta_{Y_4,BMI} BMI + \beta_{Y_4,MDI} MDI + \varepsilon_{Y_4} \\ Y_5 & = \alpha_{Y_5} + \beta_{Y_5,age} age + \beta_{Y_5,BMI} BMI + \beta_{Y_5,MDI} MDI + \varepsilon_{Y_5} \end{bmatrix}$$

where $\varepsilon_1, \varepsilon_2, \varepsilon_3, \varepsilon_4, \varepsilon_5$ are the residual errors. The residuals are assumed to have zero mean and finite variance, respectively, $\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2, \sigma_5^2$. Here we make no assumption on the correlation structure between the residuals.

5.1 Fitting multiple linear regression in R

We can estimate all the 5 models and store them into a list:

```
ls.lm <- list(Y1 = lm(Y1 ~ age + BMI + MDI, data = dfW),
              Y2 = lm(Y2 ~ age + BMI + MDI, data = dfW),
              Y3 = lm(Y3 ~ age + BMI + MDI, data = dfW),
              Y4 = lm(Y4 ~ age + BMI + MDI, data = dfW),
              Y5 = lm(Y5 ~ age + BMI + MDI, data = dfW)
              )
```

5.2 Interpretation of the regression coefficients

Same as in the univariate case (see section 4.2).

5.3 Diagnostics tools for univariate linear regression in R

Same as in the univariate case (see section 4.3). This model checking needs to be done for each outcome.

5.4 Hypothesis testing

We now want to test:

$$(\mathcal{H}_0)\ \beta_{Y_1,MDI,0} = 0 \text{ and } \beta_{Y_2,MDI,0} = 0 \text{ and } \beta_{Y_3,MDI,0} = 0 \text{ and } \beta_{Y_4,MDI,0} = 0 \text{ and } \beta_{Y_5,MDI,0} = 0$$

The p-values returned by `summary` are no more valid since we are performing multiple tests (here 5 tests). A basic solution would be to collect the p-values:

```
vec.p.value <- unlist(lapply(ls.lm, function(x){
    summary(x)$coef["MDI","Pr(>|t|)"]
}))
```

and adjust them for multiple comparisons using Bonferroni:

```
p.adjust(vec.p.value, method = "bonferroni")
```

```
          Y1           Y2           Y3           Y4           Y5
3.299432e-04 4.218369e-02 3.552579e-01 2.276690e-07 8.565878e-01
```

While easy to use this approach tends to be too conservative (i.e. give to large p-values) when the test statistics are correlated. This is usually the case when the outcomes are correlated. We will therefore use a more efficient correction called the Dunnett approach. First we need to define the null hypothesis that we want to test via a contrast matrix. For simple null hypotheses like the one we are considering in this example, we can use the function `createContrast` that will create the matrix for us:

```
resC <- createContrast(ls.lm, var.test = "MDI", add.variance = TRUE)
```

This function defines for each model the appropriate contrast matrix:

```
resC$mlf
```

```
$Y1
    (Intercept) age BMI MDI sigma2
MDI           0   0   0   1      0

$Y2
    (Intercept) age BMI MDI sigma2
MDI           0   0   0   1      0

$Y3
    (Intercept) age BMI MDI sigma2
MDI           0   0   0   1      0

$Y4
    (Intercept) age BMI MDI sigma2
MDI           0   0   0   1      0

$Y5
    (Intercept) age BMI MDI sigma2
MDI           0   0   0   1      0

attr(,"class")
[1] "mlf"
```

and right hand side of the null hypothesis:

```
resC$null
```

```
Y1: MDI Y2: MDI Y3: MDI Y4: MDI Y5: MDI
      0       0       0       0       0
```

We will now call `glht2` to perform the adjustment for multiple comparisons but first we need to convert the list into a `mmm` object:

```
class(ls.lm) <- "mmm"
e.glht_lm <- glht2(ls.lm, linfct = resC$contrast, rhs = resC$null)
e.glht_lm
```

```
         General Linear Hypotheses

Linear Hypotheses:
             Estimate
Y1: MDI == 0  0.15104
Y2: MDI == 0  0.16770
Y3: MDI == 0  0.14907
Y4: MDI == 0  0.19860
Y5: MDI == 0  0.09806
```

We can now correct for multiple comparisons using the (single-step) Dunnett approach:

```
summary(e.glht_lm, test = adjusted("single-step"))
```

```
         Simultaneous Tests for General Linear Hypotheses

Linear Hypotheses:
             Estimate Std. Error t value Pr(>|t|)
Y1: MDI == 0  0.15104    0.03441   4.389   <0.001 ***
Y2: MDI == 0  0.16770    0.06093   2.752   0.0286 *
Y3: MDI == 0  0.14907    0.08067   1.848   0.1996
Y4: MDI == 0  0.19860    0.03039   6.535   <0.001 ***
Y5: MDI == 0  0.09806    0.07057   1.390   0.4208
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```

Note that the p-value for the global test equals to the smallest p-value. This means that we reject the global null hypothesis whenever we reject the null hypothesis for any of the outcome (after adjustment for multiple comparisons!).

For comparison one can change the argument in `adjust` to apply the Bonferroni adjustment:

```
summary(e.glht_lm, test = adjusted("bonferroni"))
```

```
         Simultaneous Tests for General Linear Hypotheses

Linear Hypotheses:
             Estimate Std. Error t value Pr(>|t|)
Y1: MDI == 0  0.15104    0.03441   4.389  0.00033 ***
Y2: MDI == 0  0.16770    0.06093   2.752  0.04218 *
Y3: MDI == 0  0.14907    0.08067   1.848  0.35526
Y4: MDI == 0  0.19860    0.03039   6.535 2.28e-07 ***
Y5: MDI == 0  0.09806    0.07057   1.390  0.85659
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- bonferroni method)
```

Finally, confidence intervals can be obtained using the `confint` function:

```
confint(e.glht_lm)
```

```
         Simultaneous Confidence Intervals

Fit: NULL

Quantile = 2.5215
95% family-wise confidence level


Linear Hypotheses:
             Estimate lwr      upr
Y1: MDI == 0  0.15104  0.06427  0.23782
Y2: MDI == 0  0.16770  0.01407  0.32133
Y3: MDI == 0  0.14907 -0.05434  0.35248
Y4: MDI == 0  0.19860  0.12197  0.27524
Y5: MDI == 0  0.09806 -0.07987  0.27599
```

Note that by default the `confint` function output confidence intervals using the (single-step) Dunnett approach.

6 References

Pipper, C. B., Ritz, C., and Bisgaard, H. (2012). A versatile method for confirmatory evaluation of the effects of a covariate in multiple models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 61(2):315–326.

Tsiatis, A. (2007). *Semiparametric theory and missing data*. Springer Science & Business Media.

A Statistics: definitions and notations

A.1 Variables

We can differentiate several types of random variables: outcomes, exposure, risk factors, confounders, and mediators. To explicit the difference between these types of variables we consider a set of random variables (Y, E, X_1, X_2, M) whose relationships are displayed on Figure 1:

- **outcome** (Y): random variables that are observed with noise. It can be for instance the 5HT-4 binding in a specific brain region. When considering several outcomes we will denote in bold variable that stands for a vector of random variables: $\mathbf{Y} = (Y_1, Y_2, \ldots, Y_m)$. This happens for instance when studying the binding in several brain regions. In such a case we expect the outcomes to be correlated.

- **exposure** (E): a variable that may affect the outcome or be associated with the outcome *and* we are interested in studying this effect/association. It can for instance be a genetic factor that is hypothesized to increase the 5HT-4 binding, or a disease like depression that is associated with a change in binding (we don't know whether one causes the other or whether they have a common cause, e.g. a genetic variant).

- **risk factor/confounder** (X_1, X_2): a variable that may affect the outcome or be associated with the outcomes *but* we are *not* interested in studying their effect/association. Risk factors (denoted by X_1) are only associated with the outcomes and confounders that are both associated with the outcome and the exposure. We usually need to account for confounders the statistical model in order to obtain unbiased estimates while accounting for risk factors only enables to obtain more precise estimates (at least in linear models).

- **mediator** (M): a variable that modulate the effect of the exposure, i.e. stands on the causal pathway between the exposure and the outcome. For instance, the permeability of the blood-brain barrier may modulate the response to drugs and can act as a mediator. It is important to keep in mind that when we are interested in the (total) effect of E on Y, we should *not* adjust the analysis on M[2]. Doing so we would remove the effect of E mediated by M and therefore bias the estimate of the total effect (we would only get the direct effect).

In the following we will assume that we do not measure any mediator variable and therefore ignore this type of variable. Also we will call **covariates** the variables E, X_1, X_2.

[2]This may not be true in specific types of confounding but we will ignore that.

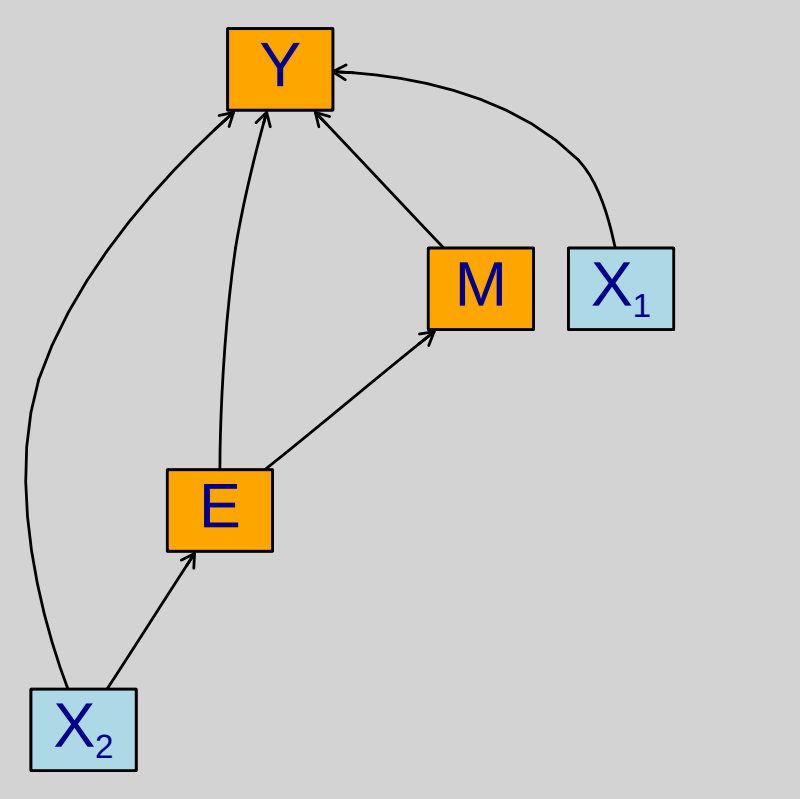


Figure 1: Path diagram relating the variables Y, E, M, X_1 and X_2

A.2 Assumptions

We can distinguish two types of assumptions:

- **causal assumptions**: saying which variables are related and in which direction. This can be done by drawing a path diagram similar to Figure 1. In simple univariate models it may seems unnecessary to draw the path diagram since the system of variables is very simple to visualize. In multivariate model, it is often very useful to draw it. Some of these assumptions are untestable, e.g. often we cannot decide whether it is E that impacts Y or whether it is Y that impacts E just based on the data.
- **modeling assumptions**: specifying the type of relationship between variables (e.g. linear) and the marginal or joint distribution (e.g. Gaussian). Often these assumptions can be tested and relaxed using a more flexible model. While appealing, there are some drawbacks with using a very flexible model: more data are needed to get precise estimates and the interpretation of the results is more complex.

A.3 Statistical model

A statistical model $\mathcal{M}$ is set of possible probability distributions. For instance when we fit a Gaussian linear model for Y_1 with just an intercept $\mathcal{M} = \left\{\mathcal{N}\left(\mu, \sigma^2\right); \mu \in \mathbb{R},\ \sigma^2 \in \mathbb{R}^+\right\}$: $\mathcal{M}$ is the set containing all possible univariate normal distributions.

A.4 Model parameters

The model parameters are the (non random) variables that enable the statistical model to "adapt" to different settings. They will be denoted Θ. They are the one that are estimated when we fit the statistical model using the data or that we specify when we simulate data. In the previous example, we could simulate data corresponding to a Gaussian linear model using the `rnorm` function in R:

```
rnorm
```

```
function (n, mean = 0, sd = 1)
.Call(C_rnorm, n, mean, sd)
<bytecode: 0x000000001d7eb938>
<environment: namespace:stats>
```

We would need to specify:

- n the sample size
- $\Theta = (\mu, \sigma^2)$ the model parameters, here μ corresponds to `mean` and σ to `sd`.

The true model parameters are the model parameters that have generated the observed data. They will be denoted Θ_0. For instance if in reality the binding potential is normally distributed with mean 5 and variance $2^2 = 4$, then $\Theta_0 = (\mu_0, \sigma_0^2) = (5, 4)$. Then doing our experiment we observed data such as:

```
set.seed(10)
Y_1.XP1 <- rnorm(10, mean = 5, sd = 2)
Y_1.XP1
```

```
[1] 5.037492 4.631495 2.257339 3.801665 5.589090 5.779589 2.583848 4.272648 1.746655 4.487043
```

If we were to re-do the experiment we would observe new data but Θ_0 would not change:

```
Y_1.XP2 <- rnorm(10, mean = 5, sd = 2)
Y_1.XP2
```

```
[1] 7.203559 6.511563 4.523533 6.974889 6.482780 5.178695 3.090112 4.609699 6.851043 5.965957
```

The estimated parameters are the parameters that we estimate when we fit the statistical model. They will be denoted $\hat{\Theta}$. We usually try to find parameters whose value maximize the chance of simulating the observed data under the estimated model (maximum likelihood estimation, MLE). For instance in the first experiment all values are positive so we would not estimate a negative mean value. In our example, $\hat{\mu}$ the MLE of μ reduces to the empirical average and $\hat{\sigma}^2$ the MLE of σ^2 to the empirical variance:

```
Theta_hat.XP1 <- c(mu_hat = mean(Y_1.XP1),
                   sigma2_hat = var(Y_1.XP1))
Theta_hat.XP1
```

```
  mu_hat sigma2_hat
4.018686   1.959404
```

Clearly the estimated coefficients vary across experiments:

```
Theta_hat.XP2 <- c(mu_hat = mean(Y_1.XP2),
                   sigma2_hat = var(Y_1.XP2))
Theta_hat.XP2
```

```
  mu_hat sigma2_hat
5.739183   1.799311
```

A.5 Parameter of interest

The statistical model may contain many parameters, most of them are often not of interest but are needed to obtain valid estimates (e.g. account for confounders). In most settings, the parameter of interest is one (or several) model parameter(s) - or simple transformation of them. For instance if we are interested in the average binding potential in the population our parameter of interest is μ.

Often, the aim of a study is to obtain the best estimate of the parameter of interest μ. Best means:

- **unbiased**: if we were able to replicate the study many times, i.e. get several estimates $\hat{\mu}_1, \hat{\mu}_2, \ldots, \hat{\mu}_K$, the average estimate $< \hat{\mu} >= \frac{\hat{\mu}_1 + \hat{\mu}_2 + \ldots + \hat{\mu}_K}{K}$ would coincide with the true one μ_0.
- **minimal variance**: if we were able to replicate the study many times, the variance of the estimates $\frac{(\hat{\mu}_1 - <\hat{\mu}>)^2 + \ldots + (\hat{\mu}_K - <\hat{\mu}>)^2}{K-1}$ should be as low as possible.

There will often be a trade-off between these two objectives. A very flexible method is more likely to give an unbiased estimate (e.g. being able to model non-linear relationship) at the price of greater uncertainty about the estimates. Often we favor unbiasedness over minimal variance. Indeed, if several studies are published with the same parameter of interest, one can pool the results to obtain an estimate with lower variance. Note that we have no guarantee that it will reduce the bias.

A.6 Contrast matrix

When dealing with many parameters it is convenient to define the null hypothesis via a contrast matrix. An example of null hypothesis is:

$$(\mathcal{H}_0)\ \beta_{MDI,0} = 0$$

If we consider $\Theta = (\alpha, \beta_{age}, \beta_{BMI}, \beta_{MDI})$, this null hypothesis can be equivalently written:

$$c = [0\ 0\ 0\ 1]$$

such that:

$$(\mathcal{H}_0)\ c\Theta_0^\intercal = 0$$

Indeed

$$c\Theta_0^\intercal = 0 * \alpha_0 + 0 * \beta_{age,0} + 0 * \beta_{BMI,0} + 1 * \beta_{MDI,0} = \beta_{MDI,0}$$

An example where the contrast matrix is useful is

- when one wish to test linear combination of parameters, e.g. consider the null hypothesis:

$$(\mathcal{H}_0)\ \beta_{MDI,0} = \beta_{BMI,0}$$

Here the contrast matrix would be:

$$c = [0\ 0\ -1\ 1]$$

- when one wish to test several hypotheses simultaneously, e.g. consider the null hypothesis:

$$(\mathcal{H}_0)\ \beta_{BMI,0} = 0 \text{ or } \beta_{MDI,0} = 0$$

Here the contrast matrix would be:

$$C = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

In **R**, the method `createContrast` helps to define the contrast matrix:

```
Clin <- createContrast(e.lm, par = c("MDI - BMI = 0"),
                       add.variance = FALSE, rowname.rhs = FALSE)
Clin$contrast
```

```
           (Intercept) age BMI MDI
- BMI + MDI           0   0  -1   1
```

```
Csim <- createContrast(e.lm, par = c("BMI = 0","MDI = 0"),
                       add.variance = FALSE, rowname.rhs = FALSE)
Csim$contrast
```

```
    (Intercept) age BMI MDI
BMI           0   0   1   0
MDI           0   0   0   1
```

Then the contrast matrix can be send to `glht` to obtain p-values and confidence intervals:

```
elin.glht <- glht(e.lm, linfct = Clin$contrast)
summary(elin.glht)
```

```
         Simultaneous Tests for General Linear Hypotheses

Fit: lm(formula = Y1 ~ age + BMI + MDI, data = dfW)

Linear Hypotheses:
                 Estimate Std. Error t value Pr(>|t|)
- BMI + MDI == 0 -0.09608    0.06993  -1.374    0.176
(Adjusted p values reported -- single-step method)
```

```
esim.glht <- glht(e.lm, linfct = Csim$contrast)
summary(esim.glht)
```

```
         Simultaneous Tests for General Linear Hypotheses

Fit: lm(formula = Y1 ~ age + BMI + MDI, data = dfW)

Linear Hypotheses:
         Estimate Std. Error t value Pr(>|t|)
BMI == 0  0.24712    0.05791   4.268 0.000195 ***
MDI == 0  0.15104    0.03441   4.389 0.000132 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
(Adjusted p values reported -- single-step method)
```