Research Module in Econometrics & Statistics

JProf. Dominik Liebl

2018-10-29

Contents

Preface

This is the script for the research module in econometrics & statistics.

Repo that makes this site: https://github.com/lidom/RM_ES_Script

General Topic: Regression analysis and beyond

Description: This research module covers modern methods in statistics and econometrics with a focus on regression analysis. Participants have the opportunity to choose among a set of specific projects. Topics suggested by the participants are generally appreciated, but will be assessed with respect to their practical feasibility. All projects should have a theoretical part describing the model and/or the estimation procedures, a Monte-Carlo simulation study, and an application to real data. Depending on the actual number of participants, it might be that the project work has to be carried out as a group task rather than as an individual task. The first five to six weeks consist of lectures (4h per week). Participation is strongly recommended and active participation is desirable. After the lecture series, the groups will have regular meetings with the supervisor.

Grading: Each student will be evaluated on the basis of a presentation and a seminar paper.

Important: You need to register for this course via BASIS. Registration period: Oct. 15-22.

Time Table:

Date	Time	Topic
08.10.	14:15 - 15:45	General Introduction
10.10.	14:15 - 15:45	Introduction to R
15.10.	14:15 - 15:45	Test Theory
17.10.	14:15 - 15:45	Test Theory
22.10.	14:15 - 15:45	Estimation Theory
24.10.	14:15 - 15:45	Estimation Theory / **Final allocation of topics**
29.10.	14:15 - 15:45	Regression Analysis
31.10.	14:15 - 15:45	Monte-Carlo Simulations
07.11.	14:15 - 15:45	How to Write and Present
21.01.	14:15 - 15:45	Presentations
23.01.	14:15 - 15:45	Presentations

- **Location:** Room 0.042
- **Supervision meetings:** From Nov. to Jan. at the office of JProf. Liebl

Deadline for submission of term papers: Feb. 28, 2019, via e-mail to dliebl@uni-bonn.de

Term Paper:

Every term paper should consist of the following parts:

- Introduction of the general problem and a short overview about the relevant literature.
- Detailed description of the considered method(s).

- Assessment of the method(s) by means of Monte-Carlo simulations.
- Application to real data.

Page Count:

- For groups of 1-2: 10-15 pages (plus bibliography and appendix)
- For groups of 3-4: 15-20 pages (plus bibliography and appendix)
- Long tables, proofs, additional figures, etc. should be placed in the appendix.
- Line-spacing: 1.5

Presentations:

- For groups of 1-2: 10-15 minutes
- For groups of 3-4: 20-25 minutes

Topics

- **Nonparametric Regression** Literature: Li and Racine (2007), Fan and Gijbels (1996), and Wand and Jones (1994)
- **Panel Data Analysis** Literature: Hsiao (2014), Greene (2003), and Baltagi (2008)
- **Multilevel (Mixed Effects) Linear Models** Literature: Gelman and Hill (2006), Verbeke and Molenberghs (2000), and Gałecki and Burzykowski (2013)
- **Covariance Matrix Estimators (HAC and Friends)** Literature: White (2014), Ch. 6 and Vignettes of the sandwich R-package
- **Multiple Testing** Literature: Romano and Wolf (2005), F. Bretz (2010), and Y. Hochberg (1987)
- **Statistical Learning with Sparsity (Lasso and Generalizations)** Literature: Hastie et al. (2015)

Alternative topics suggested by the participants are generally appreciated.

Chapter 1

Introduction to R

This tutorial aims to serve as an introduction to the software package R. Other very good and much more exhaustive tutorials and useful reference-cards can be found at the following links:

- Reference card for R commands (always useful)
- Matlab/R reference card (for those who are more familiar with Matlab)
- The official Introduction to R (very detailed)
- And many more at www.r-project.org (see "Documents")
- An interactive introduction can be done online at: www.datacamp.com
- An excellent book project which covers also advanced issues such as "writing performant code" and "package development": adv-r.had.co.nz

Why R?

- R is **free** of charge from: www.r-project.org
- The celebrated IDE **RStudio** for R is also **free** of charge: www.rstudio.com
- R is equipped with one of the most flexible and powerful graphics routines available anywhere. For instance, check out one of the following repositories:
  - Clean Graphs
  - R graph catalog
  - Publication Ready Plots
- Today, R is the de-facto standard for statistical science.

1.1 Short Glossary

Lets start the tutorial with a (very) short glossary:

- **Console**: The thing with the "$>$" sign at the beginning.
- **Script file**: An ordinary text file with suffix "**.R**". For instance, **yourFavoritFileName.R**.
- **Working directory**: The file-directory you are working in. Useful commands: with `getwd()` you get the location of your current working directory and `setwd()` allows you to set a new location for it.
- **Workspace**: This is a hidden file (stored in the working directory), where all objects you use (e.g., data, matrices, vectors, variables, functions, etc.) are stored. Useful commands: `ls()` shows all elements in our current workspace and `rm(list=ls())` deletes all elements in our current workspace.

1.2 First Steps

A good idea is to use a script file such as **yourFavoritFileName.R** in order to store your R commands. You can send single lines or marked regions of your R-code to the console by pressing the keys **STRG+ENTER**.

To begin with baby steps, do some simple computations:

```
2+2 # and all the others: *,/,-,^2,^3,...
```

```
## [1] 4
```

Note: Everything that is written after the #-sign is ignored by R, which is very useful to comment your code.

The **assignment operator** will be your most often used tool. Here an example to create a **scalar** variable:

```
x <- 4
x
```

```
## [1] 4
```

```
4 -> x # possible but unusual
x
```

```
## [1] 4
```

Note: The R community loves the `<-` assignment operator, which is a very unusual syntax. Alternatively, you can use the `=` operator.

And now a more interesting object - a **vector**:

```
y <- c(2,7,4,1)
y
```

```
## [1] 2 7 4 1
```

The command `ls()` shows the total content of your current workspace, and the command `rm(list=ls())` deletes all elements of your current workspace:

```
ls()
```

```
## [1] "x" "y"
```

```
rm(list=ls())
ls()
```

```
## character(0)
```

Note: RStudio's **Environment** pane also lists all the elements in your current workspace. That is, the command `ls()` becomes a bit obsolete when working with RStudio.

Let's try how we can compute with vectors and scalars in R.

```
x <- 4
y <- c(2,7,4,1)

x*y # each element in the vector, y, is multiplied by the scalar, x.
```

```
## [1]  8 28 16  4
```

```
y*y # this is a term by term product of the elements in y
```

```
## [1]  4 49 16  1
```

Performing vector multiplications as you might expect from your last math-course, e.g., an outer product: $y\,y^\top$:

```
y %*% t(y)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    4   14    8    2
## [2,]   14   49   28    7
## [3,]    8   28   16    4
## [4,]    2    7    4    1
```

Or an inner product $y^\top y$:

```
t(y) %*% y
```

```
##      [,1]
## [1,]   70
```

Note: Sometimes, R's treatment of vectors can be annoying. The product `y %*% y` is treated as the product `t(y) %*% y`.

The term-by-term execution as in the above example, `y*y`, is actually a central strength of R. We can conduct many operations **vector-wisely**:

```
y^2
```

```
## [1]  4 49 16  1
```

```
log(y)
```

```
## [1] 0.6931472 1.9459101 1.3862944 0.0000000
```

```
exp(y)
```

```
## [1]    7.389056 1096.633158   54.598150    2.718282
```

```
y-mean(y)
```

```
## [1] -1.5  3.5  0.5 -2.5
```

```
(y-mean(y))/sd(y) # standardization
```

```
## [1] -0.5669467  1.3228757  0.1889822 -0.9449112
```

This is a central characteristic of so called matrix based languages like R (or Matlab). Other programming languages often have to use **loops** instead:

```
N <- length(y)
1:N

y.sq <- numeric(N)
y.sq

for(i in 1:N){
  y.sq[i] <- y[i]^2
  if(i == N){
    print(y.sq)
  }
}
```

The `for()`-loop is the most common loop. But there is also a `while()`-loop and a `repeat()`-loop. However, loops in R can be rather slow, therefore, try to avoid them!

Useful commands to produce **sequences** of numbers:

```
1:10
-10:10
?seq # Help for the seq()-function
seq(from=1, to=100, by=7)
```

Using the sequence command `1:16`, we can go for our first **matrix**:

```
?matrix
A <- matrix(data=1:16, nrow=4, ncol=4)
A
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    5    9   13
## [2,]    2    6   10   14
## [3,]    3    7   11   15
## [4,]    4    8   12   16
```

```
A <- matrix(1:16, 4, 4)
```

Note that a matrix has always two **dimensions**, but a vector has only one dimension:

```
dim(A)    # Dimension of matrix A?
```

```
## [1] 4 4
```

```
dim(y)    # dim() does not operate on vectors.
```

```
## NULL
```

```
length(y) # Length of vector y?
```

```
## [1] 4
```

Lets play a bit with the matrix `A` and the vector `y`. As we have seen in the loop above, the `[]`-operator **selects elements** of vectors and matrices:

```
A[,1]
A[4,4]
y[c(1,4)]
```

This can be done on a more **logical** basis, too. For example, if you want to know which elements in the first column of matrix `A` are strictly greater than 2:

```
A[,1][A[,1]>2]
```

```
## [1] 3 4
```

```
# Note that this give you a boolean vector:
A[,1]>2
```

```
## [1] FALSE FALSE  TRUE  TRUE
```

```
# And you can use it in a non-sense relation, too:
y[A[,1]>2]
```

```
## [1] 4 1
```

Note: Logical operations return so-called **boolean** objects, i.e., either a `TRUE` or a `FALSE`. For instance, if we ask R whether `1>2` we get the answer `FALSE`.

1.3 Further Data Objects

Besides classical data objects such as scalars, vectors, and matrices there are three further data objects in R:

1. The **array**: As a matrix but with more dimensions. Here is an example of a $2 \times 2 \times 2$-dimensional `array`:

```
myFirst.Array <- array(c(1:8), dim=c(2,2,2)) # Take a look at it!
```

2. The **list**: In `lists` you can organize different kinds of data. E.g., consider the following example:

```
myFirst.List <- list("Some_Numbers" = c(66, 76, 55, 12, 4, 66, 8, 99),
                     "Animals"      = c("Rabbit", "Cat", "Elefant"),
                     "My_Series"    = c(30:1))
```

A very useful function to find specific values and entries within lists is the `str()`-function:

```
str(myFirst.List)
```

```
## List of 3
##  $ Some_Numbers: num [1:8] 66 76 55 12 4 66 8 99
##  $ Animals     : chr [1:3] "Rabbit" "Cat" "Elefant"
##  $ My_Series   : int [1:30] 30 29 28 27 26 25 24 23 22 21 ...
```

3. The **data frame**: A `data.frame` is a `list`-object but with some more formal restrictions (e.g., equal number of rows for all columns). As indicated by its name, a `data.frame`-object is designed to store data:

```
myFirst.Dataframe <- data.frame("Credit_Default"   = c( 0, 0, 1, 0, 1, 1),
                                "Age"              = c(35,41,55,36,44,26),
                                "Loan_in_1000_EUR" = c(55,65,23,12,98,76))
# Take a look at it!
```

1.4 Simple Regression Analysis using R

Alright, let's do some statistics with real data. You can download the data HERE. Save it on your computer, at a place where you can find it, and give the path (e.g. `"C:\textbackslash path\textbackslash auto.data.csv"`, which references to the data, to the *file*-argument of the function `read.csv()`:

```
# ATTENTION! YOU HAVE TO CHANGE "\" TO "/":
auto.data <- read.csv(file="C:/your_path/autodata.txt", header=TRUE)
head(auto.data)
```

If you have problems to read the data into R, go on with these commands. (For this you need a working internet connection!):

```
# install.packages("readr")
library("readr")
auto.data <- suppressMessages(read_csv(file = "https://cdn.rawgit.com/lidom/Teaching_Repo/bc692b56/autoc
# head(auto.data)
```

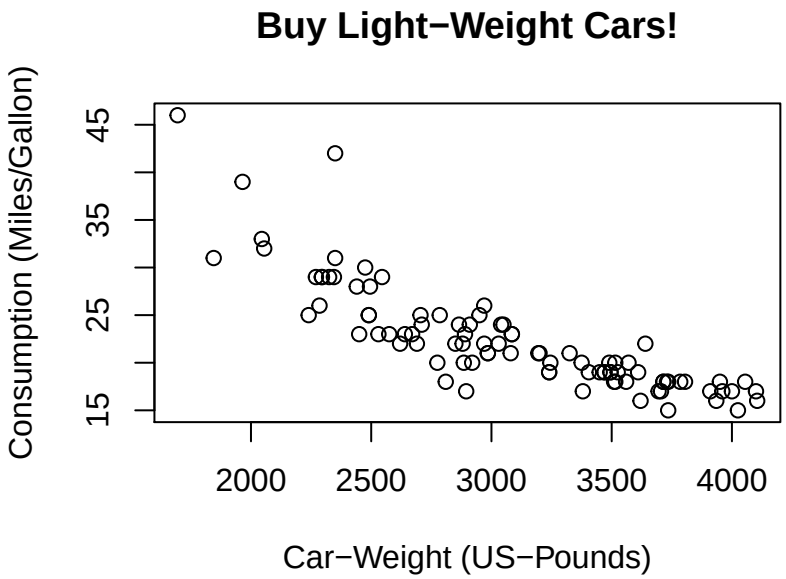
You can select specific variables of the `auto.data` using the `$`-operator:

```
gasolin.consumption      <- auto.data$MPG.city
car.weight               <- auto.data$Weight
## Take a look at the first elements of these vectors:
head(cbind(gasolin.consumption,car.weight))
```

```
##      gasolin.consumption car.weight
## [1,]                  25       2705
## [2,]                  18       3560
## [3,]                  20       3375
## [4,]                  19       3405
## [5,]                  22       3640
## [6,]                  22       2880
```

This is how you can produce your first plot:

```
## Plot the data:
plot(y=gasolin.consumption, x=car.weight,
     xlab="Car-Weight (US-Pounds)",
     ylab="Consumption (Miles/Gallon)",
     main="Buy Light-Weight Cars!")
```



As a first step, we might assume a simple kind of linear relationship between the variables `gasolin.consumption` and `car.weight`. Let us assume that the data was generated by the following simple regression model:

$$y_i = \alpha + \beta_1 x_i + \varepsilon_i, \quad i = 1, \ldots, n$$

where y_i denotes the gasoline-consumption, x_i the weight of car i, and ε_i is a mean zero constant variance noise term. (This is clearly a non-sense model!)

The command `lm()` computes the estimates of this linear regression model. The command (in fact it's a *method*) `summary()` computes further quantities of general interest from the *object* that was returned from the `lm()` function.

```
lm.result   <- lm(gasolin.consumption~car.weight)
lm.summary  <- summary(lm.result)
lm.summary
```

```
##
## Call:
## lm(formula = gasolin.consumption ~ car.weight)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.7946 -1.9711  0.0249  1.1855 13.8278
```

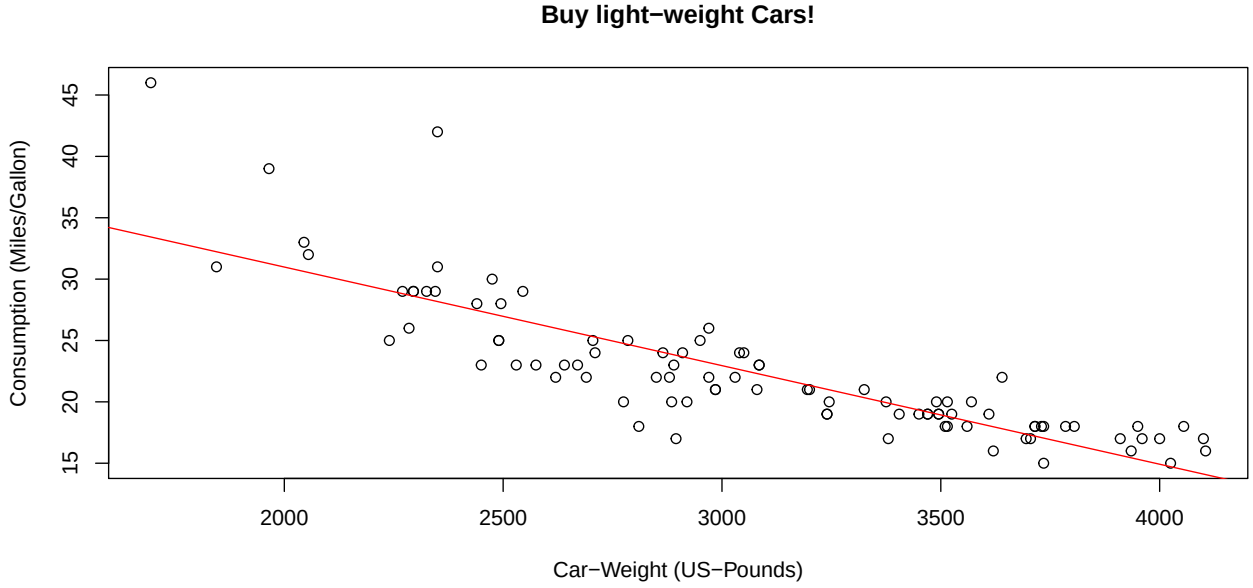
```
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 47.048353   1.679912   28.01   <2e-16 ***
## car.weight  -0.008032   0.000537  -14.96   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.038 on 91 degrees of freedom
## Multiple R-squared:  0.7109, Adjusted R-squared:  0.7077
## F-statistic: 223.8 on 1 and 91 DF,  p-value: < 2.2e-16
```

Of course, we want to have a possibility to access all the quantities computed so far, e.g., in order to plot the results. This can be done as following:

```
## Accessing the computed quantities
names(lm.summary) ## Alternatively: str(lm.summary)
```

```
##  [1] "call"          "terms"         "residuals"     "coefficients"
##  [5] "aliased"       "sigma"         "df"            "r.squared"
##  [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
```

```
alpha <- lm.summary$coefficients[1]
beta  <- lm.summary$coefficients[2]

## Plot all:
plot(y=gasolin.consumption, x=car.weight,
     xlab="Car-Weight (US-Pounds)",
     ylab="Consumption (Miles/Gallon)",
     main="Buy light-weight Cars!")
abline(a=alpha,
       b=beta, col="red")
```



1.5 Programming in R

Let's write, i.e., program our own R-function for estimating linear regression models. In order to be able to validate our function, we start with **simulating data** for which we then *know* all true parameters. Simulating data is like being the "Data-God": For instance, we generate realizations of the error term ε_i, i.e., something which we *never* observe in real data.

Let us consider the following multiple regression model:

$$y_i = \beta_1 + \beta_2 x_{2i} + \beta_3 x_{3i} + \varepsilon_i, \quad i = 1, \ldots, n,$$

where ε_i is a heteroscedastic error term

$$\varepsilon_i \sim N(0, \sigma_i^2), \quad \sigma_i = x_{3i},$$

and where for all $i = 1, \ldots, n = 50$:

- $x_{2i} \sim N(10, 1.5^2)$
- x_{3i} comes from a t-distribution with 5 degrees of freedom and non-centrality parameter 2

```
set.seed(109) # Sets the "seed" of the random number generators:
n   <- 50     # Number of observations

## Generate two explanatory variables plus an intercept-variable:
X.1 <- rep(1, n)                  # Intercept
X.2 <- rnorm(n, mean=10, sd=1.5) # Draw realizations form a normal distr.
X.3 <- rt(n, df=5, ncp=2)         # Draw realizations form a t-distr.
X   <- cbind(X.1, X.2, X.3)       # Save as a Nx3-dimensional data matrix.
```

OK, we have regressors, i.e., data that we also have in real data sets.

Now we define the elements of the β-vector. Be aware of the difference: In real data sets we do not know the true β-vector, but try to estimate it. However, when simulating data, we determine (as "Data-Gods") the true β-vector and can compare our estimate $\hat{\beta}$ with the true β:

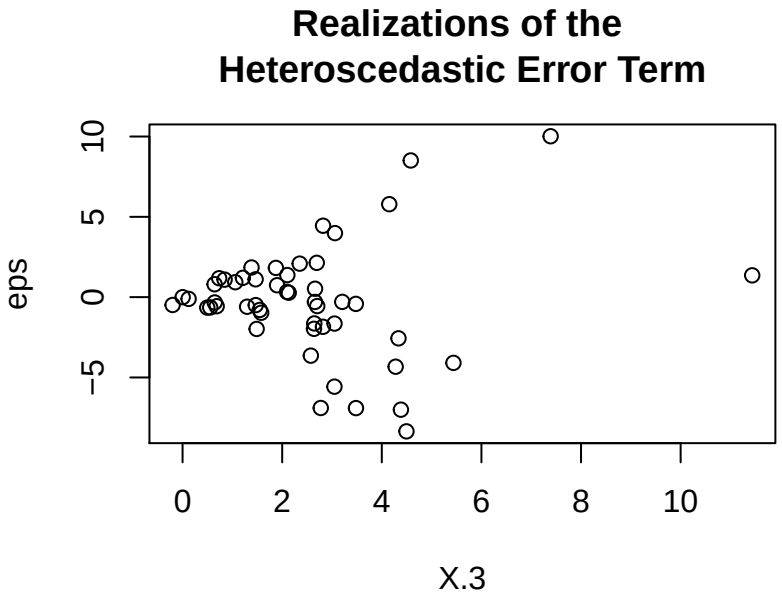
```
## Define the slope-coefficients
beta.vec  <- c(1,-5,5)
```

We still need to simulate realizations of the dependent variable y_i. Remember that $y_i = \beta_1 x_{1i} + \beta_1 x_{2i} + \beta_3 x_{3i} + \varepsilon_i$. That is, we only need realizations from the error terms ε_i in order to compute the realizations from y_i. This is how you can simulate realizations from the heteroscedastic error terms ε_i:

```
## Generate realizations from the heteroscadastic error term
eps       <- (X.3)*rnorm(n, mean=0, sd=1)
```

Take a look at the heteroscedasticity in the error term:

```
plot(y=eps, x=X.3,
     main="Realizations of the \nHeteroscedastic Error Term")
```

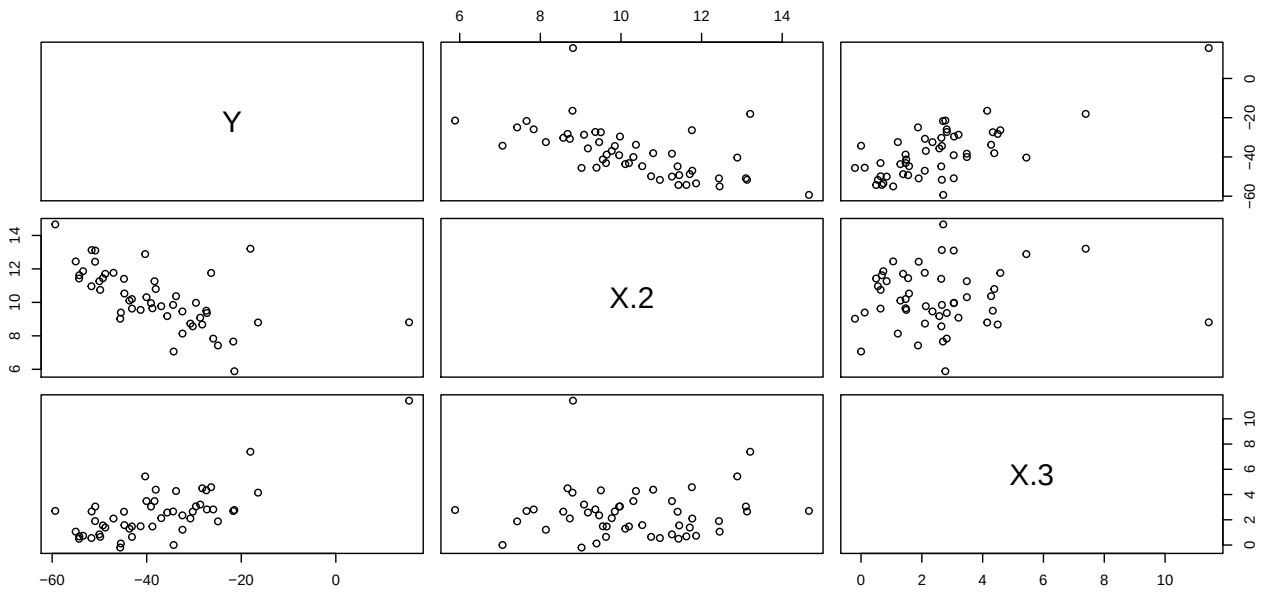


With the (pseudo-random) realizations from ε_i, we can finally generate realizations from the dependent variable y_i:

```
## Dependent variable:
y   <- X %*% beta.vec + eps
```

Let's take a look at the data:

```
mydata    <- data.frame("Y"=y, "X.1"=X.1, "X.2"=X.2, "X.3"=X.3)
pairs(mydata[,-2]) # The '-2' removes the intercept variable "X.1"
```



Once we have data, we can compute the OLS estimate of the true β vector. Remember the formula:

$$\hat{\beta} = (X^\top X)^{-1} X^\top y$$

In R-Code this is: $(X^\top X)^{-1}$ =`solve(t(X) %*% X)`, i.e.:

```
## Computation of the beta-Vector:
beta.hat <- solve(t(X) %*% X) %*% t(X) %*% y
beta.hat
```

```
##          [,1]
## X.1 -2.735042
## X.2 -4.685719
## X.3  5.091811
```

Well done. Using the above lines of code we can easily program our own `myOLSFun()` function!

```
myOLSFun <- function(y, x, add.intercept=FALSE){

  ## Number of Observations:
  n         <- length(y)

  ## Add an intercept to x:
  if(add.intercept){
    Intercept <- rep(1, n)
    x         <- cbind(Intercept, x)
  }

  ## Estimation of the slope-parameters:
  beta.hat.vec <- solve(t(x) %*% x) %*% t(x) %*% y

  ## Return the result:
  return(beta.hat.vec)
}

## Run the function:
myOLSFun(y=y, x=X)
```

```
##          [,1]
## X.1 -2.735042
## X.2 -4.685719
## X.3  5.091811
```

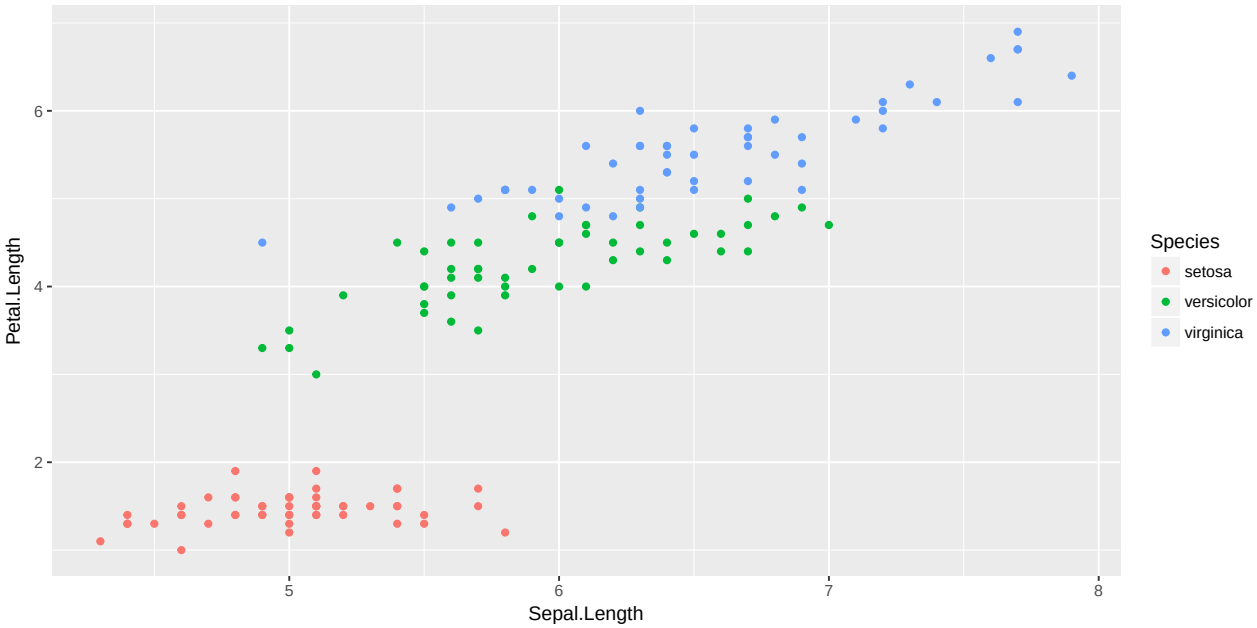
Can you extend the function for the computation of the covariance matrix of the slope-estimates, several measures of fits (R^2, adj.-R^2, etc.), t-tests, ...?

1.6 R-packages

One of the best features in R are its contributed packages. The list of all packages on CRAN is impressive! Take a look at it HERE

For instance, nice plots can be produced using the R-package is `ggplot2`. You can find an intro do this package HERE.

```
# install.packages("ggplot2")
library("ggplot2")

qplot(Sepal.Length, Petal.Length, data = iris, color = Species)
```



Of course, `ggplot2` concerns "only" plotting, but you'll find R-packages for almost any statistical method out there.

1.7 Tidyverse

The `tidyverse` package is a collection of packages that lets you import, manipulate, explore, visualize and model data in a harmonized and consistent way which helps you to be more productive.

Installing the `tidyverse` package:

```
install.packages("tidyverse")
```

To use the `tidyverse` package load it using the `library()` function:

```
library(tidyverse)
```

```
## -- Attaching packages ----------------- tidyverse 1.2.1 --

## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.2.0
## v purrr   0.2.5     v forcats 0.3.0

## -- Conflicts -------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Chick Weight Data

R comes with many datasets installed. We will use the `ChickWeight` dataset to learn about the tidyverse. The help system gives a basic summary of the experiment from which the data was collect:

> *"The body weights of the chicks were measured at birth and every second day thereafter until day 20. They were also measured on day 21. There were four groups of chicks on different protein diets."*

You can get more information, including references by typing:

```
help("ChickWeight")
```

The Data: There are 578 observations (rows) and 4 variables:

- `Chick` – unique ID for each chick.
- `Diet` – one of four protein diets.
- `Time` – number of days since birth.
- `weight` – body weight of chick in grams.

Note: `weight` has a lower case `w` (recall R is case sensitive).

Store the data locally:

```
ChickWeight %>%
  select(Chick, Diet, Time, weight) %>%
  arrange(Chick, Diet, Time) %>%
  write_csv("ChickWeight.csv")
```

First we will import the data from a file called `ChickWeight.csv` using the `read_csv()` function from the `readr` package (part of the `tidyverse`). The first thing to do, outside of R, is to open the file `ChickWeight.csv` to check what it contains and that it makes sense. Now we can import the data as follows:

```
CW <- read_csv("ChickWeight.csv")
```

```
## Parsed with column specification:
## cols(
##   Chick = col_integer(),
##   Diet = col_integer(),
##   Time = col_integer(),
##   weight = col_integer()
## )
```

If all goes well then the data is now stored in an R object called `CW`. If you get the following error message then you need to change the working directory to where the data is stored.

```
Error: 'ChickWeight.csv' does not exist in current
working directory ...
```

Changing the working directory: In RStudio you can use the menu bar ("Session - Set Working Directory - Choose Directory…"). Alternatively, you can use the function `setwd()`.

Looking at the Dataset: To look at the data type just type the object (dataset) name:

```
CW
```

```
## # A tibble: 578 x 4
##    Chick  Diet  Time weight
##    <int> <int> <int>  <int>
##  1    18     1     0     39
##  2    18     1     2     35
##  3    16     1     0     41
##  4    16     1     2     45
##  5    16     1     4     49
##  6    16     1     6     51
##  7    16     1     8     57
##  8    16     1    10     51
##  9    16     1    12     54
## 10    15     1     0     41
## # ... with 568 more rows
```

If there are too many variables then not all them may be printed. To overcome this issue we can use the `glimpse()` function which makes it possible to see every column in your dataset (called a "data frame" in R speak).

```
glimpse(CW)
```

```
## Observations: 578
## Variables: 4
## $ Chick  <int> 18, 18, 16, 16, 16, 16, 16, 16, 16, 15, 15, 15, 15, 15,...
## $ Diet   <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ Time   <int> 0, 2, 0, 2, 4, 6, 8, 10, 12, 0, 2, 4, 6, 8, 10, 12, 14,...
## $ weight <int> 39, 35, 41, 45, 49, 51, 57, 51, 54, 41, 49, 56, 64, 68,...
```

The function `View()` allows for a spread-sheet type of view on the data:

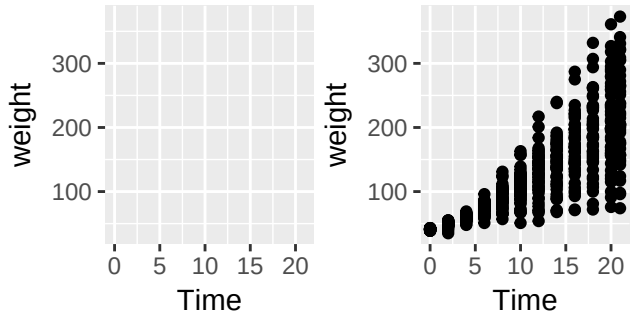
```
View(CW)
```

1.7.1 Tidyverse: Plotting Basics

To **visualise** the chick weight data, we will use the `ggplot2` package (part of the `tidyverse`). Our interest is in seeing how the *weight changes over time for the chicks by diet*. For the moment don't worry too much about the details just try to build your own understanding and logic. To learn more try different things even if you get an error messages.

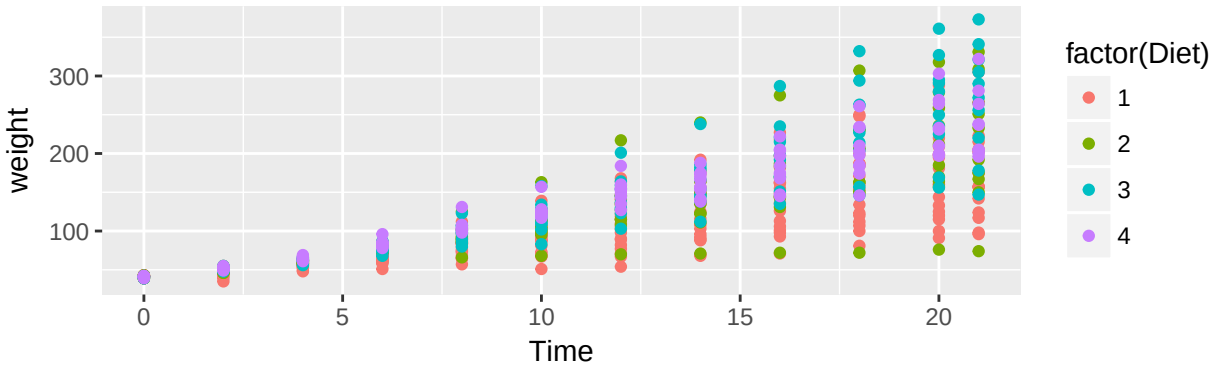
Let's plot the weight data (vertical axis) over time (horizontal axis).

```
# An empty plot (the plot on the left)
ggplot(CW, aes(Time, weight))
# With data (the plot on the right)
ggplot(CW, aes(Time, weight)) + geom_point()
```



Add color for `Diet`. The graph above does not differentiate between the diets. Let's use a different color for each diet.

```
# Adding colour for diet
ggplot(CW,aes(Time,weight,colour=factor(Diet))) +
  geom_point()
```



It is difficult to conclude anything from this graph as the points are printed on top of one another (with diet 1 underneath and diet 4 at the top).

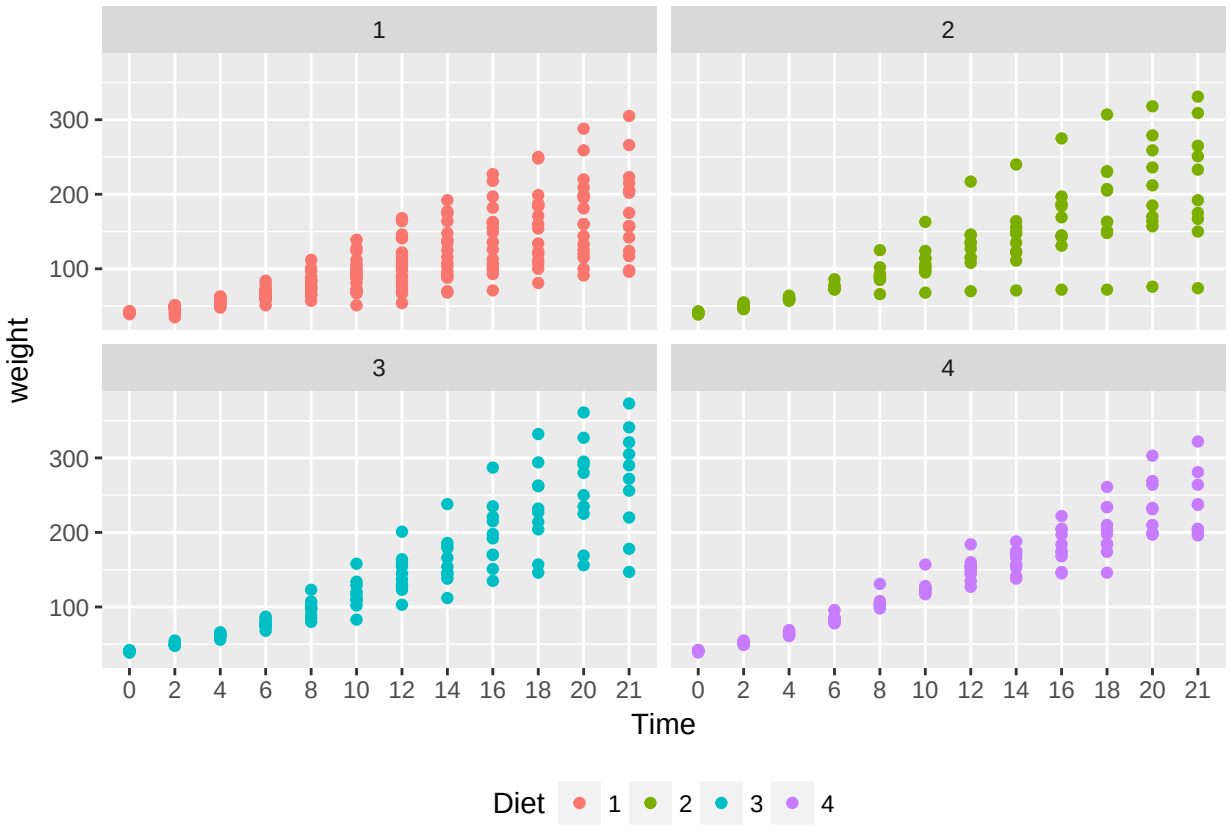
Factor Variables: Before we continue, we have to make an important change to the `CW` dataset by making `Diet` and `Time` *factor variables*. This means that R will treat them as categorical variables (see the `<fct>` variables below) instead of continuous variables. It will simplify our coding. The next section will explain the `mutate()` function.

```
CW <- mutate(CW, Diet = factor(Diet))
CW <- mutate(CW, Time = factor(Time))
glimpse(CW)
```

```
## Observations: 578
## Variables: 4
## $ Chick  <int> 18, 18, 16, 16, 16, 16, 16, 16, 16, 15, 15, 15, 15, 15,...
## $ Diet   <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1...
## $ Time   <fct> 0, 2, 0, 2, 4, 6, 8, 10, 12, 0, 2, 4, 6, 8, 10, 12, 14,...
## $ weight <int> 39, 35, 41, 45, 49, 51, 57, 51, 54, 41, 49, 56, 64, 68,...
```

The `facet_wrap()` function: To plot each diet separately in a grid using `facet_wrap()`:

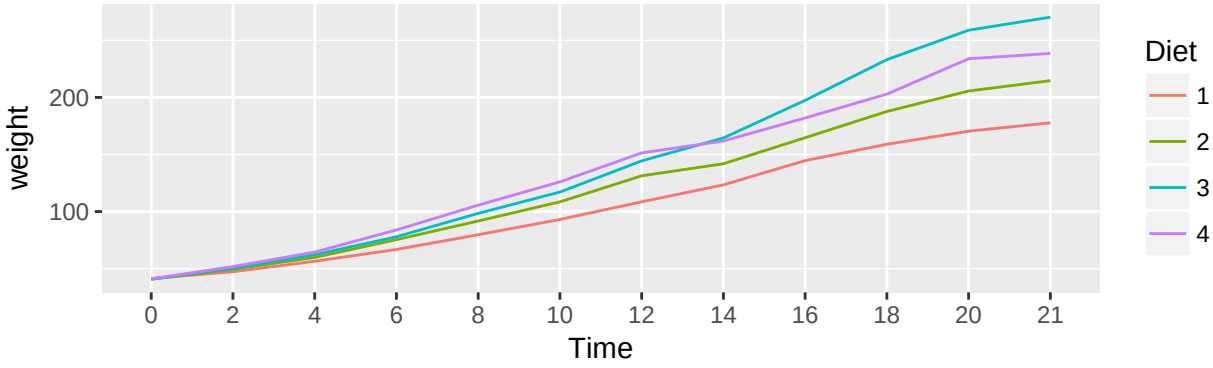
```
# Adding jitter to the points
ggplot(CW, aes(Time, weight, colour=Diet)) +
  geom_point() +
  facet_wrap(~Diet) +
  theme(legend.position = "bottom")
```



Interpretation: Diet 4 has the least variability but we can't really say anything about the mean effect of each diet although diet 3 seems to have the highest.

Next we will plot the **mean changes** over time for each diet using the `stat_summary()` function:

```
ggplot(CW, aes(Time, weight,
               group=Diet, colour=Diet)) +
  stat_summary(fun.y="mean", geom="line")
```

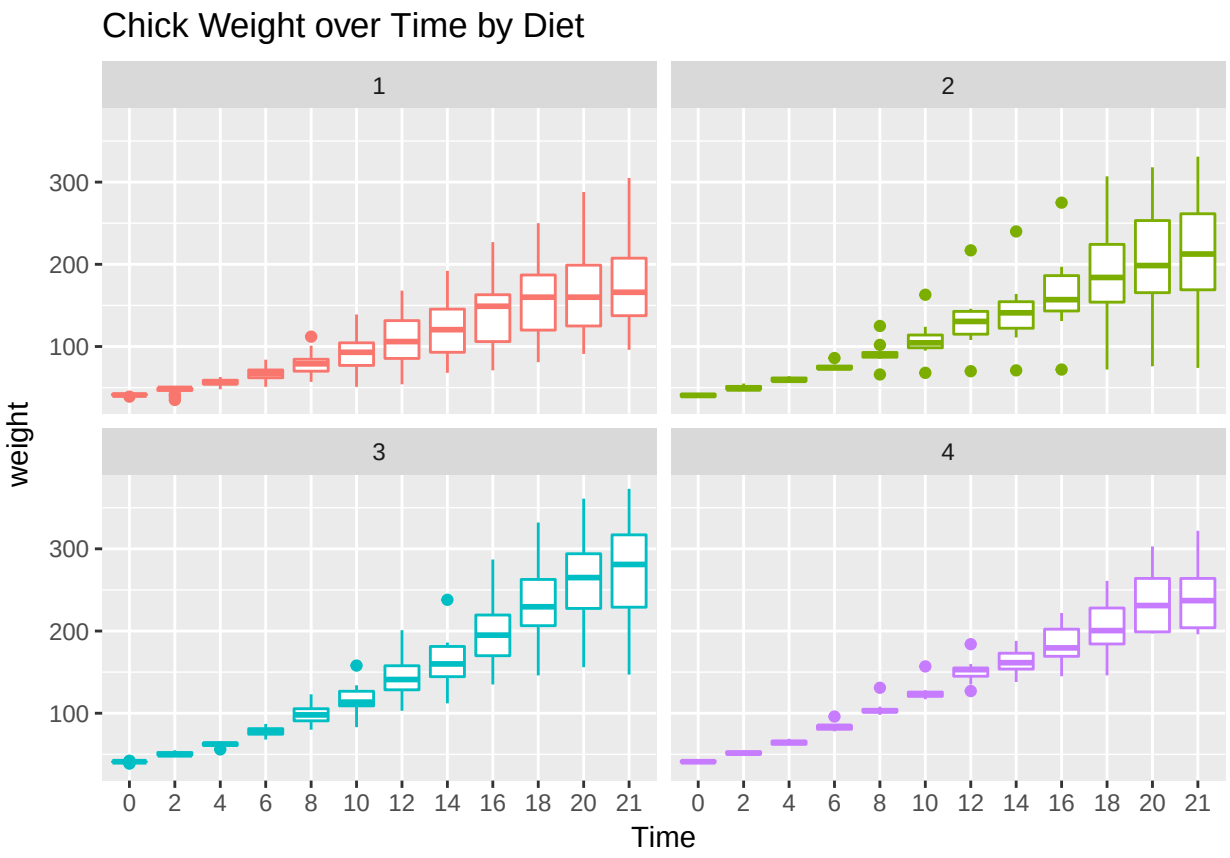


Interpretation: We can see that diet 3 has the highest mean weight gains by the end of the experiment. However, we don't have any information about the variation (uncertainty) in the data.

To see variation between the different diets we use `geom_boxplot` to plot a box-whisker plot. A note of caution is that the number of chicks per diet is relatively low to produce this plot.

```
ggplot(CW, aes(Time, weight, colour=Diet)) +
  facet_wrap(~Diet) +
  geom_boxplot() +
```

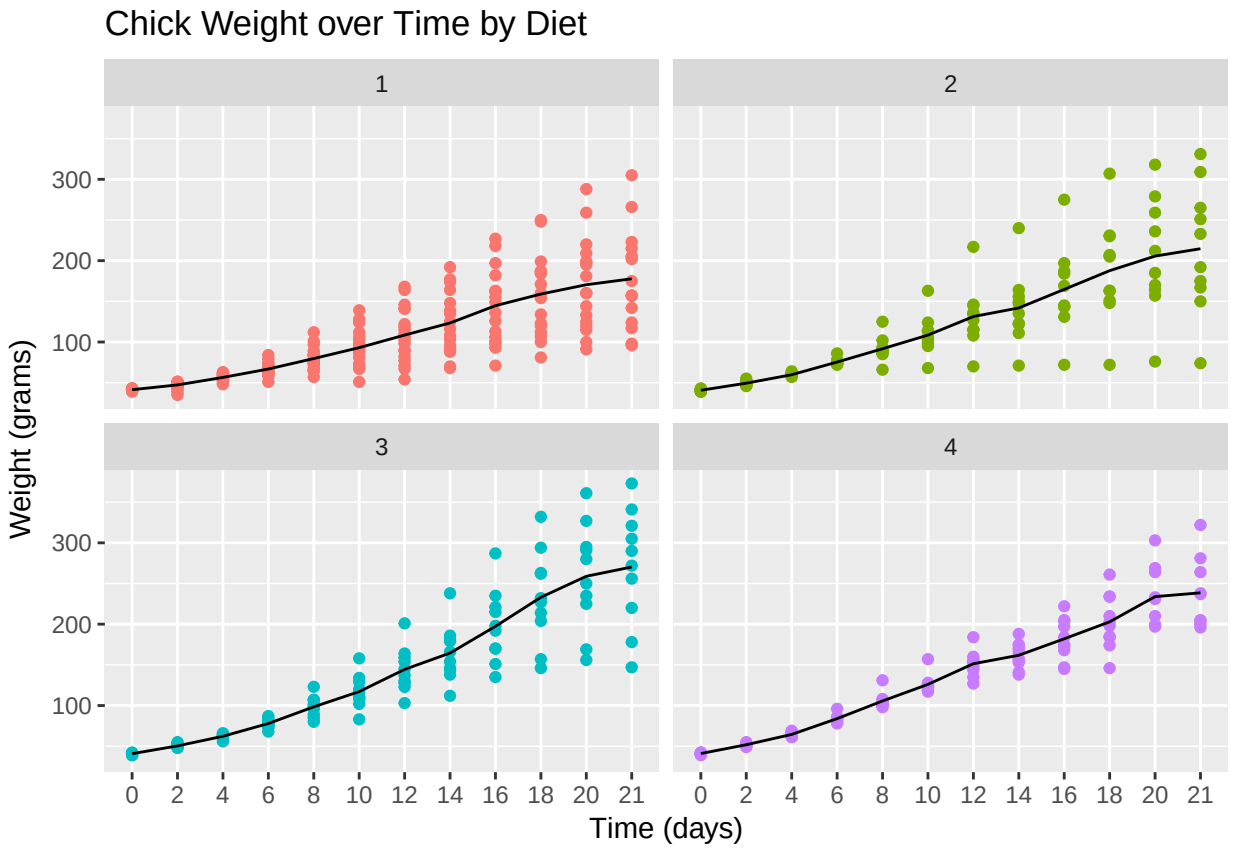
```
  theme(legend.position = "none") +
  ggtitle("Chick Weight over Time by Diet")
```



Interpretation: Diet 3 seems to have the highest "average" weight gain but it has more variation than diet 4 which is consistent with our findings so far.

Let's finish with a plot that you might include in a publication.

```
ggplot(CW, aes(Time, weight, group=Diet,
                             colour=Diet)) +
  facet_wrap(~Diet) +
  geom_point() +
  # geom_jitter() +
  stat_summary(fun.y="mean", geom="line",
               colour="black") +
  theme(legend.position = "none") +
  ggtitle("Chick Weight over Time by Diet") +
  xlab("Time (days)") +
  ylab("Weight (grams)")
```



1.7.2 Tidyverse: Data Wrangling Basics

In this section we will learn how to wrangle (manipulate) datasets using the `tidyverse` package. Let's start with the `mutate()`, `select()`, `rename()`, `filter()` and `arrange()` functions.

`mutate()`: Adds a new variable (column) or modifies an existing one. We already used this above to create factor variables.

```
# Added a column
CWm1 <- mutate(CW, weightKg = weight/1000)
CWm1
```

```
## # A tibble: 578 x 5
##   Chick Diet  Time  weight weightKg
##   <int> <fct> <fct>  <int>    <dbl>
## 1    18 1     0         39    0.039
## 2    18 1     2         35    0.035
## 3    16 1     0         41    0.041
## # ... with 575 more rows
```

```
# Modify an existing column
CWm2 <- mutate(CW, Diet = str_c("Diet ", Diet))
CWm2
```

```
## # A tibble: 578 x 4
##   Chick Diet   Time  weight
##   <int> <chr>  <fct>  <int>
## 1    18 Diet 1 0         39
```

```
## 2    18 Diet 1 2         35
## 3    16 Diet 1 0         41
## # ... with 575 more rows
```

`select()`: Keeps, drops or reorders variables.

```
# Drop the weight variable from CWm1 using minus
select(CWm1, -weight)
```

```
## # A tibble: 578 x 4
##   Chick Diet  Time  weightKg
##   <int> <fct> <fct>    <dbl>
## 1    18 1     0        0.039
## 2    18 1     2        0.035
## 3    16 1     0        0.041
## # ... with 575 more rows
```

```
# Keep variables Time, Diet and weightKg
select(CWm1, Chick, Time, Diet, weightKg)
```

```
## # A tibble: 578 x 4
##   Chick Time  Diet  weightKg
##   <int> <fct> <fct>    <dbl>
## 1    18 0     1        0.039
## 2    18 2     1        0.035
## 3    16 0     1        0.041
## # ... with 575 more rows
```

`rename()`: Renames variables whilst keeping all variables.

```
rename(CW, Group = Diet, Weight = weight)
```

```
## # A tibble: 578 x 4
##   Chick Group Time  Weight
##   <int> <fct> <fct>  <int>
## 1    18 1     0         39
## 2    18 1     2         35
## 3    16 1     0         41
## # ... with 575 more rows
```

`filter()`: Keeps or drops observations (rows).

```
filter(CW, Time==21 & weight>300)
```

```
## # A tibble: 8 x 4
##   Chick Diet  Time  weight
##   <int> <fct> <fct>  <int>
## 1     7 1     21       305
## 2    29 2     21       309
## 3    21 2     21       331
## # ... with 5 more rows
```

For comparing values in vectors use: `<` (less than), `>` (greater than), `<=` (less than and equal to), `>=` (greater than and equal to), `==` (equal to) and `!=` (not equal to). These can be combined logically using `&` (and) and `|` (or).

`arrange()`: Changes the order of the observations.

```
arrange(CW, Chick, Time)
```

```
## # A tibble: 578 x 4
##   Chick Diet  Time  weight
##   <int> <fct> <fct>  <int>
## 1     1 1     0         42
## 2     1 1     2         51
## 3     1 1     4         59
## # ... with 575 more rows
```

```
arrange(CW, desc(weight))
```

```
## # A tibble: 578 x 4
##   Chick Diet  Time  weight
##   <int> <fct> <fct>  <int>
## 1    35 3     21       373
## 2    35 3     20       361
## 3    34 3     21       341
## # ... with 575 more rows
```

What does the `desc()` do? Try using `desc(Time)`.

1.7.3 The pipe operator %>%

In reality you will end up doing multiple data wrangling steps that you want to save. The pipe operator `%>%` makes your code nice and readable:

```
CW21 <- CW %>%
  filter(Time %in% c(0, 21)) %>%
  rename(Weight = weight) %>%
  mutate(Group = factor(str_c("Diet ", Diet))) %>%
  select(Chick, Group, Time, Weight) %>%
  arrange(Chick, Time)
CW21
```

```
## # A tibble: 95 x 4
##   Chick Group  Time  Weight
##   <int> <fct>  <fct>  <int>
## 1     1 Diet 1 0         42
## 2     1 Diet 1 21       205
## 3     2 Diet 1 0         40
## # ... with 92 more rows
```

Hint: To understand the code above we should read the pipe operator `%>%` as "then".

> Create a new dataset (object) called `CW21` using dataset `CW` ***then*** keep the data for days 0 and 21 ***then*** rename variable `weight` to `Weight` ***then*** create a variable called `Group` ***then*** keep variables `Chick`, `Group`, `Time` and `Weight` and ***then*** finally arrange the data by variables `Chick` and `Time`.

This is the same code:

```
CW21 <- CW %>%
  filter(., Time %in% c(0, 21)) %>%
  rename(., Weight = weight) %>%
  mutate(., Group=factor(str_c("Diet ",Diet))) %>%
  select(., Chick, Group, Time, Weight) %>%
  arrange(., Chick, Time)
```

The pipe operator, `%>%`, replaces the dots (`.`) with whatever is returned from code preceding it. For

example, the dot in `filter(., Time %in% c(0, 21))` is replaced by `CW`. The output of the `filter(...)` then replaces the dot in `rename(., Weight = weight)` and so on. Think of it as a data assembly line with each function doing its thing and passing it to the next.

1.7.4 The `group_by()` function

From the data visualizations above we concluded that the diet 3 has the highest mean and diet 4 the least variation. In this section, we will quantify the effects of the diets using **summmary statistics**. We start by looking at the number of observations and the mean by **diet** and **time**.

```
mnsdCW <- CW %>%
  group_by(Diet, Time) %>%
  summarise(N = n(), Mean = mean(weight)) %>%
  arrange(Diet, Time)
mnsdCW
```

```
## # A tibble: 48 x 4
## # Groups:   Diet [4]
##   Diet  Time      N  Mean
##   <fct> <fct> <int> <dbl>
## 1 1     0        20  41.4
## 2 1     2        20  47.2
## 3 1     4        19  56.5
## # ... with 45 more rows
```

For each distinct combination of `Diet` and `Time`, the chick weight data is summarized into the number of observations (`N`) and the mean (`Mean`) of `weight`.

Further summaries: Let's also calculate the standard deviation, median, minimum and maximum values but only at days 0 and 21.

```
sumCW <-  CW %>%
  filter(Time %in% c(0, 21)) %>%
  group_by(Diet, Time) %>%
  summarise(N = n(),
            Mean = mean(weight),
            SD = sd(weight),
            Median = median(weight),
            Min = min(weight),
            Max = max(weight)) %>%
  arrange(Diet, Time)
sumCW
```

```
## # A tibble: 8 x 8
## # Groups:   Diet [4]
##   Diet  Time      N  Mean     SD Median   Min   Max
##   <fct> <fct> <int> <dbl>  <dbl>  <dbl> <dbl> <dbl>
## 1 1     0        20  41.4  0.995   41      39    43
## 2 1     21       16 178.  58.7    166      96   305
## 3 2     0        10  40.7  1.49    40.5    39    43
## # ... with 5 more rows
```

Let's make the summaries "prettier", say, for a report or publication.

```
library("knitr") # to use the kable() function
prettySumCW <- sumCW %>%
 mutate(`Mean (SD)` = str_c(format(Mean, digits=1),
```

```
           " (", format(SD, digits=2), ")")) %>%
 mutate(Range = str_c(Min, " - ", Max)) %>%
 select(Diet, Time, N, `Mean (SD)`, Median, Range) %>%
 arrange(Diet, Time) %>%
 kable(format = "latex")
prettySumCW
```

Diet	Time	N	Mean (SD)	Median	Range
1	0	20	41 (0.99)	41.0	39 - 43
1	21	16	178 (58.70)	166.0	96 - 305
2	0	10	41 (1.5)	40.5	39 - 43
2	21	10	215 (78.1)	212.5	74 - 331
3	0	10	41 (1)	41.0	39 - 42
3	21	10	270 (72)	281.0	147 - 373
4	0	10	41 (1.1)	41.0	39 - 42
4	21	9	239 (43.3)	237.0	196 - 322

Interpretation: This summary table offers the same interpretation as before, namely that diet 3 has the highest mean and median weights at day 21 but a higher variation than group 4. However it should be noted that at day 21, diet 1 lost 4 chicks from 20 that started and diet 4 lost 1 from 10. This could be a sign of some health related issues.

1.8 Further Links

1.8.1 Further R-Intros

- https://eddelbuettel.github.io/gsir-te/Getting-Started-in-R.pdf
- https://www.datacamp.com/courses/free-introduction-to-r
- https://swcarpentry.github.io/r-novice-gapminder/
- https://support.rstudio.com/hc/en-us/articles/200526207-Using-Projects

1.8.2 Version Control (Git/GitHub)

- https://support.rstudio.com/hc/en-us/articles/200532077-Version-Control-with-Git-and-SVN
- http://happygitwithr.com/
- https://www.gitkraken.com/

1.8.3 R-Ladies

- https://rladies.org/

Chapter 2

Statistical Hypothesis Testing

2.1 Hypotheses and Test-Statistics

Assume an independently and identically distributed (i.i.d.) random sample $X_1, \ldots, X_n$, where the distributions of $X_1, \ldots, X_n$ depend on some unknown parameter $\theta \in \Omega$, where Ω is some parameter space.

General Testing Problem:

$$H_0 : \theta \in \Omega_0$$

against

$$H_1 : \theta \in \Omega_1$$

H_0 is the null hypothesis, while H_1 is the alternative. $\Omega_0 \subset \Omega$ and $\Omega_1 \subset \Omega$ are used to denote the possible values of θ under H_0 and H_1. Necessarily, $\Omega_0 \cap \Omega_1 = \emptyset$.

For a large number of tests we have $\Omega = \mathbb{R}$ and the respective null hypothesis states that θ has a specific value $\theta_0 \in \mathbb{R}$, i.e., $\Omega_0 = \{\theta_0\}$ and $H_0 : \theta = \theta_0$. Depending on the alternative one then often distinguishes between one-sided ($\Omega_1 = (\theta_0, \infty)$ or $\Omega_1 = (-\infty, \theta_0)$) and two-sided tests ($\Omega_1 = \{\theta \in \mathbb{R} | \theta \neq \theta_0\}$).

The data $X_1, \ldots, X_n$ is used in order to decide whether to accept or to reject H_0.

Test Statistic: Every statistical hypothesis test relies on a corresponding test statistic

$$T = T(X_1, \ldots, X_n).$$

Any test statistic is a real valued random variable, and for given data the resulting observed value T_{obs} is used to decide between H_0 and H_1. Generally, the distribution of T under H_0 is analyzed in order to define a **rejection region** C:

- $T_{obs} \notin C \Rightarrow H_0$ is not rejected
- $T_{obs} \in C \Rightarrow H_0$ is rejected

For one-sided tests C is typically of the form $(-\infty, c_0]$ or $[c_1, \infty)$. For two-sided tests C typically takes the form of $(-\infty, c_0] \cup [c_1, \infty)$. The limits c_0 and c_1 of the respective intervals are called **critical values**, and are obtained from quantiles of the **null distribution**, i.e., the distribution of T under H_0.

Decision Errors:

Decision Errors	Verbal Definition	Formal Definition
Type I error	H_0 is rejected even though H_0 is true.	$P(T \in C \mid H_0 \text{ true})$
Type II error	The test fails to reject a false H_0.	$P(T \notin C \mid H_1 \text{ true})$

2.2 Significance Level, Size and p-Values

Significance Level: In a statistical significance test, the probability of a type I error is controlled by the *significance level* α (e.g., $\alpha = 5\%$).

$$P\left(\text{Type I error}\right) = P\left(T \in C |\ H_0 \text{ true}\right) \leq \alpha$$

Size: The *size* of a statistical test is defined as

$$\sup_{\theta \in \Omega_0} P(T \in C | \theta \in \Omega_0).$$

That is, the preselected significance level α is an upper bound for the size, which may not be attained (i.e., size $< \alpha$) if, for instance, the relevant probability function is discrete.

Practically important significance levels:

- $\alpha = 0.05$: It is common to say that a test result is "significant" if a hypothesis test of level $\alpha = 0.05$ rejects H_0.
- $\alpha = 0.01$: It is common to say that a test result is "strongly significant" if a hypothesis test of level $\alpha = 0.01$ rejects H_0.

p-Value: The *p-value* is the probability of obtaining a test statistic at least as "extreme" as the one that was actually observed, assuming that the null hypothesis is true.

- For one-sided tests:
  - $P(T \geq T_{\text{obs}} | H_0 \text{ true})$ or
  - $P(T \leq T_{\text{obs}} | H_0 \text{ true})$
- For two-sided tests:
  - $2\min\{P(T \leq T_{\text{obs}} | H_0 \text{ true}),\ P(T \geq T_{\text{obs}} | H_0 \text{ true})$

Remarks:

- The p-value is random as it depends on the observed data. That is, different random samples will lead to different p-values.
- For given data, having determined the p-value of a test we also know the test decisions for all possible levels α:
  - $\alpha >$ p-value $\Rightarrow H_0$ is rejected
  - $\alpha <$ p-value $\Rightarrow H_0$ is accepted

Example: Let $X_i \sim N(\mu, \sigma^2)$ independently for all $i = 1, \ldots, 5 = n$. Observed realizations from this i.i.d. random sample: $X_1 = 19.20$, $X_2 = 17.40$, $X_3 = 18.50$, $X_4 = 16.50$, $X_5 = 18.90$. That is, the empirical mean is given by $\bar{X} = 18.1$.

Testing problem: $H_0 : \mu = \mu_0$ against $H_1 : \mu \neq \mu_0 17$ (i.e., a two-sided test), where $\mu_0 = 17$.

Since the variance is unknown, we have to use a **t-test** in order to test H_0. Test statistic of the t-test:

$$T = \frac{\sqrt{n}(\bar{X} - \mu_0)}{S},$$

where $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \bar{X})^2$ is the unbiased estimator of σ^2.

$$T_{obs} = \frac{\sqrt{5}(18.1 - 17)}{1.125} = 2.187$$

$$\Rightarrow \text{p-value} = 2\min\{P(T_{n-1} \leq 2.187),\ P(T_{n-1} \geq 2.187)\} = 0.094$$

The above computations in R

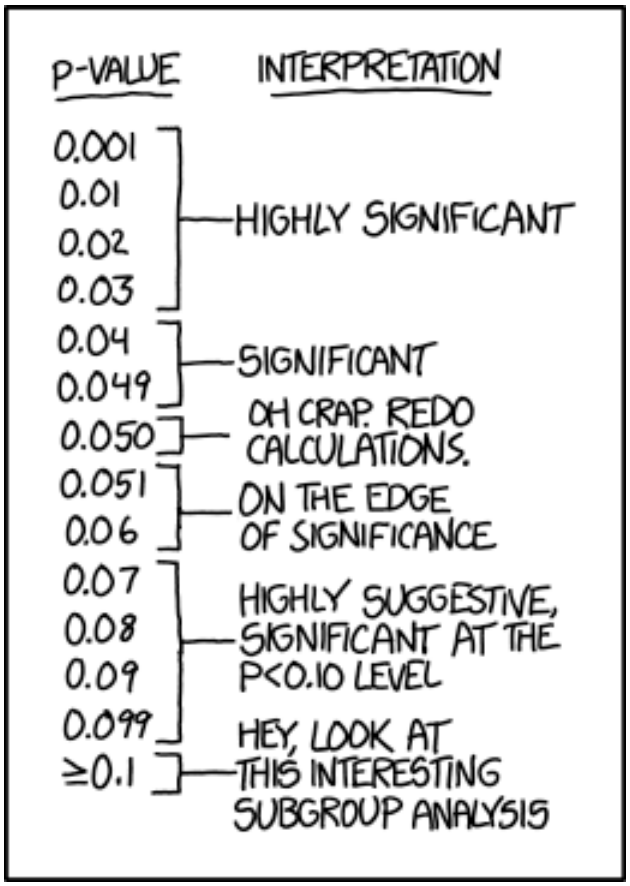


Figure 2.1: From: https://xkcd.com/1478/

```r
library("magrittr", quietly = TRUE)# for using the pipe-operator: %>%

X           <- c(19.20, 17.40, 18.50, 16.50, 18.90)
mu_0        <- 17        # hypothetical mean
n           <- length(X) # sample size
X_mean      <- mean(X)   # empirical mean
X_sd        <- sd(X)     # empirical sd
# t-test statistic
t_test_stat <- sqrt(n)*(X_mean - mu_0)/X_sd

# p-value for two-sided test
c(pt(q = t_test_stat, df = n-1, lower.tail = TRUE),
  pt(q = t_test_stat, df = n-1, lower.tail = FALSE)) %>%
  min * 2 -> p_value

p_value %>% round(., digits = 3)
```

```
## [1] 0.094
```

Of course, there is also a `t.test()` function in R:

```r
t.test(X, mu = mu_0, alternative = "two.sided")
```

```
##
##  One Sample t-test
##
## data:  X
## t = 2.1869, df = 4, p-value = 0.09402
## alternative hypothesis: true mean is not equal to 17
## 95 percent confidence interval:
##  16.70347 19.49653
## sample estimates:
## mean of x
##      18.1
```

2.3 The Power Function

For every possible value $\theta \in \Omega_0 \cup \Omega_1$, all sample sizes n and each significance level α the corresponding value of the **power function** β is defined by the following probability:

$$\beta_{n,\alpha}(\theta) := P(H_0 \text{ is rejected, if the true parameter value equals } \theta)$$

Obviously, $\beta_{n,\alpha}(\theta) \leq \alpha$ for all $\theta \in \Omega_0$. Furthermore, for any $\theta \in \Omega_1$, $1 - \beta_{n,\alpha}(\theta)$ is the probability of committing a type II error.

The power function is an important tool for accessing the quality of a test and for comparing different test procedures.

Conservative Test: If possible, a test is constructed in such a way that size equals level, i.e., $\beta_{n,\alpha}(\theta) = \alpha$ for some $\theta \in \Omega_0$. In some cases, however, as for discrete test statistics or complex, composite null hypothesis, it is not possible to reach the level, and $\sup_{\theta \in \Omega_0} \beta_{n,\alpha}(\theta) < \alpha$. In this case the test is called *conservative*.

Unbiased Test: A significance test of level $\alpha > 0$ is called *unbiased* if $\beta_{n,\alpha}(\theta) \geq \alpha$ for all $\theta \in \Omega_1$.

Consistent Test: A significance test of level $\alpha > 0$ is called *consistent* if

$$\lim_{n\to\infty} \beta_{n,\alpha}(\theta) = 1$$

for all $\theta \in \Omega_1$.

Most Powerful Test: When choosing between different testing procedures for the same testing problem, one will usually prefer the *most powerful test*. Consider a fixed sample size n. For a specified $\theta \in \Omega_1$, a test with power function $\beta_{n,\alpha}(\theta)$ is said to be **most powerful** for θ if for any alternative test with power function $\beta^*_{n,\alpha}(\theta)$,

$$\beta_{n,\alpha}(\theta) \geq \beta^*_{n,\alpha}(\theta)$$

holds for all levels $\alpha > 0$.

Uniformly Most Powerful: A test with power function $\beta_{n,\alpha}(\theta)$ is said to be *uniformly most powerful* against the set of alternatives Ω_1 if for any alternative test with power function $\beta^*_{n,\alpha}(\theta)$,

$$\beta_{n,\alpha}(\theta) \geq \beta^*_{n,\alpha}(\theta) \quad \text{holds for all } \theta \in \Omega_1, \alpha > 0$$

Unfortunately, uniformly most powerful tests only exist for very special testing problems.

Example: Let $X_1, \ldots, X_n$ be an i.i.d. random sample. Assume that $n = 9$, and that $X_i \sim N(\mu, 0.18^2)$. Hence, in this simple example only the mean $\mu = E(X)$ is unknown, while the standard deviation has the known value $\sigma = 0.18$.

Testing problem: $H_0 : \mu = \mu_0$ against $H_1 : \mu \neq \mu_0$ for $\mu_0 = 18.3$ (i.e., a two-sided test).

Since the variance is known, a test may rely on the Gauss (or Z) test statistic:

$$Z = \frac{\sqrt{n}(\bar{X} - \mu_0)}{\sigma} = \frac{3(\bar{X} - 18.3)}{0.18}$$

Under H_0 we have $Z \sim N(0, 1)$, and for the significance level $\alpha = 0.05$ the null hypothesis is rejected if

$$|Z| \geq z_{1-\alpha/2} = 1.96,$$

where $z_{1-\alpha/2}$ denotes the $(1 - \alpha/2)$-quantile of the standard normal distribution. Note that the size of this test equals its level $\alpha = 0.05$.

For determining the rejection region of a test it suffices to determine the distribution of the test statistic under H_0. But in order to calculate the power function one needs to quantify the distribution of the test statistic for all possible values $\theta \in \Omega$. For many important problems this is a formidable task. For the Gauss test, however, it is quite easy. Note that for any (true) mean value $\mu \in \mathbb{R}$ the corresponding distribution of $Z \equiv Z_\mu = \sqrt{n(\bar{X} - \mu_0)}/\sigma$ is

$$Z_\mu = \frac{\sqrt{n}(\mu - \mu_0)}{\sigma} + \frac{\sqrt{n}(\bar{X} - \mu)}{\sigma} \sim N\left(\frac{\sqrt{n}(\mu - \mu_0)}{\sigma}, 1\right)$$

This implies that

$$\begin{aligned}\beta_{n,\alpha}(\mu) &= P\left(|Z_\mu| > z_{1-\alpha/2}\right) \\ &= 1 - \Phi\left(z_{1-\alpha/2} - \frac{\sqrt{n}(\mu - \mu_0)}{\sigma}\right) + \Phi\left(-z_{1-\alpha/2} - \frac{\sqrt{n}(\mu - \mu_0)}{\sigma}\right),\end{aligned}$$

where Φ denotes the distribution function of the standard normal distribution.

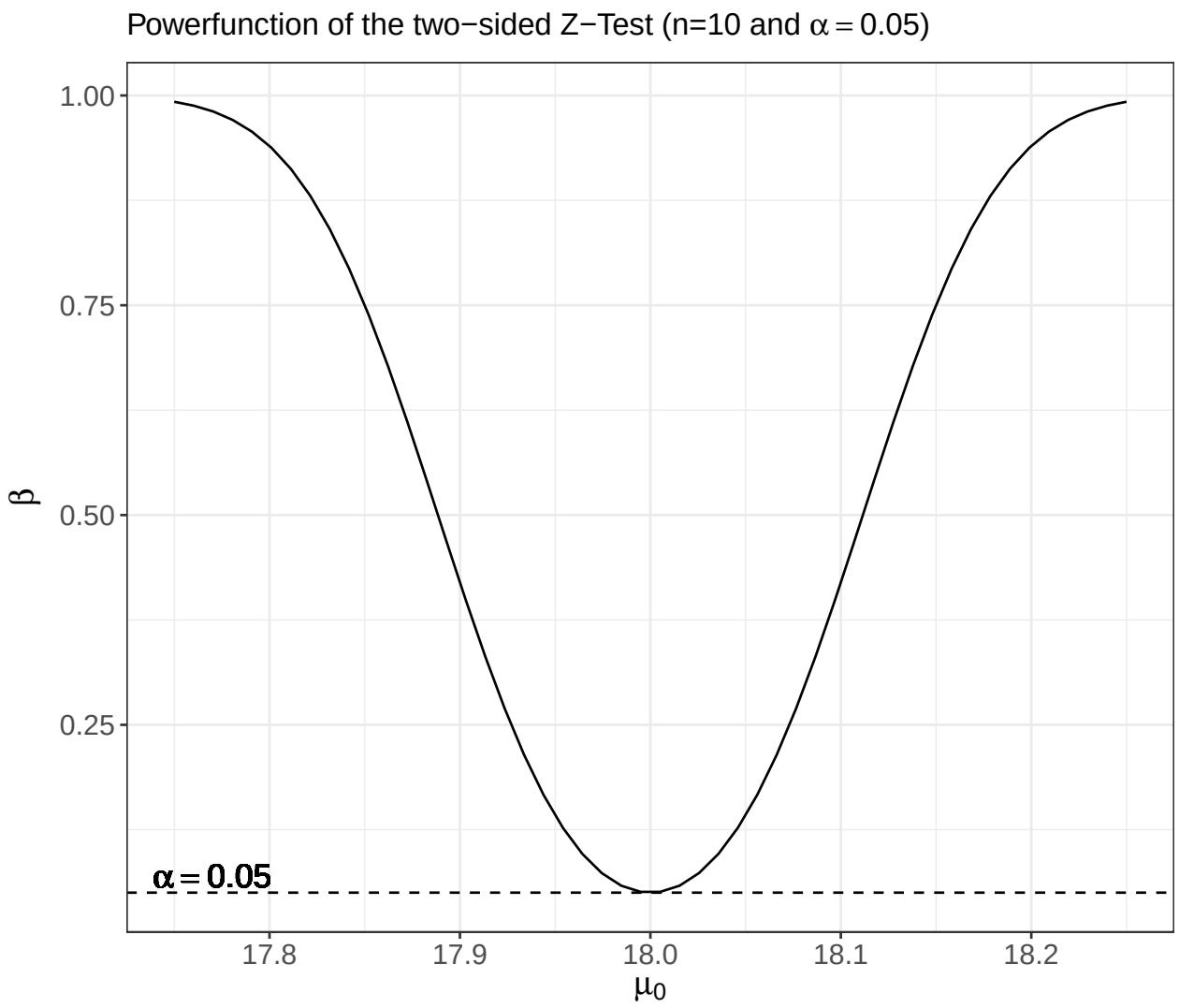
Implementing the power function of the two-sided Z-test in R:

```r
# The power function
beta_Ztest_TwoSided <- function(n, alpha, sigma, mu_0, mu){
  # (1-alpha/2)-quantile of N(0,1):
  z_upper        <- qnorm(p = 1-alpha/2)
  # location shift under H_1:
  location_shift <- sqrt(n) * (mu - mu_0)/sigma
  # compute power
  power          <- 1 - pnorm( z_upper - location_shift) +
                        pnorm(-z_upper - location_shift)
  return(power)
}

# Apply the function
n     <-  9
sigma <-  0.18
mu_0  <- 18.3
##
c(beta_Ztest_TwoSided(n = n, alpha = 0.05, sigma = sigma, mu_0 = mu_0, mu=18.35),
  beta_Ztest_TwoSided(n = n, alpha = 0.05, sigma = sigma, mu_0 = mu_0, mu=18.50),
  beta_Ztest_TwoSided(n = n, alpha = 0.01, sigma = sigma, mu_0 = mu_0, mu=18.50)) %>%
  round(., digits = 3)
```

```
## [1] 0.133 0.915 0.776
```

Plotting the graph of the power function

```r
suppressPackageStartupMessages(
  library("tidyverse")
)
# Vectorize the function with respect to mu_0:
beta_Ztest_TwoSided <- Vectorize(FUN = beta_Ztest_TwoSided,
                                 vectorize.args = "mu_0")

mu_0_vec <- seq(from = 17.75, to = 18.25, len = 50)

beta_vec <- beta_Ztest_TwoSided(n     =   10,
                                alpha = 0.05,
                                sigma = 0.18,
                                mu    =  18,
                                mu_0  = mu_0_vec)

beta_df <- data.frame("mu_0"  = mu_0_vec,
                      "Beta"  = beta_vec)

ggplot(data = beta_df, aes(x=mu_0, y=Beta)) +
  geom_line() +
  geom_hline(yintercept = 0.05, lty=2) +
  geom_text(aes(x=17.77, y=0.07, label='alpha==0.05'), parse=TRUE, size=5) +
  labs(title = expression(
    paste("Powerfunction of the two-sided Z-Test (n=10 and ",alpha==0.05,")")),
       x = expression(paste(mu[0])),
       y = expression(paste(beta)), size=8)    +
  theme_bw() +
  theme(axis.text  = element_text(size=12),
          axis.title = element_text(size=14))
```



This example illustrates the power function of a sensible test, since:

- Under $H_0 : \mu = \mu_0$ we have $\beta_{n,\alpha}(\mu_0) = \alpha$.
- The test is unbiased, since $\beta_{n,\alpha}(\mu) \geq \alpha$ for any $\mu \neq \mu_0$.
- The test is consistent, since $\lim_{n\to\infty} \beta_{n,\alpha}(\mu) = 1$ for every fixed $\mu \neq \mu_0$.
- For fixed sample size n, $\beta_{n,\alpha}(\mu)$ increases as the distance $|\mu - \mu_0|$ increases.
- If $|\mu - \mu_0| > |\mu^* - \mu_0|$ then $\beta_{n,\alpha}(\mu) > \beta_{n,\alpha}(\mu^*)$.
- $\beta_{n,\alpha}(\mu)$ decreases as the significance level α of the test decreases. I.e., if $\alpha > \alpha^*$ then $\beta_{n,\alpha}(\mu) > \beta_{n,\alpha^*}(\mu)$.

Assuming that the basic assumptions (i.e., normality and known variance) are true, the above Gauss-test is the most prominent example of a *uniformly most powerful* test. Under its (restrictive) assumptions, no other possible test can achieve a larger value of $\beta_{n,\alpha}(\mu)$ for any possible value of μ.

2.4 Asymptotic Null Distributions

Generally, the underlying distributions are unknown. In this case it is usually not possible to compute the power function of a test for fixed n. (Exceptions are so called "distribution-free" tests in nonparametric statistics.) The only way out of this difficulty is to rely on large sample asymptotics and corresponding asymptotic distributions, which allow to approximate the power function and to study the **asymptotic efficiency** of a test. The finite sample behavior of a test for different sample sizes n is then evaluated by means of **simulation studies**.

For a real-valued parameter θ most tests of $H_0 : \theta = \theta_0$ rely on estimators $\hat{\theta}$ of θ. Under suitable regularity conditions on the underlying distribution, central limit theorems usually imply that

$$\sqrt{n}(\hat{\theta} - \theta) \to_D N(0, v^2) \quad \text{as} \quad n \to \infty,$$

where v^2 is the asymptotic variance of the estimator.

Often a consistent estimator $\hat{v}^2$ of v^2 can be determined from the data. For large n we then approximately have

$$\frac{\sqrt{n}(\hat{\theta} - \theta)}{v} \overset{a}{\sim} N(0, 1).$$

For a given α, a one-sided test of $H_0 : \theta = \theta_0$ against $H_1 : \theta > \theta_0$ then rejects H_0 if

$$Z = \frac{\sqrt{n}(\hat{\theta} - \theta_0)}{v} > z_{1-\alpha}.$$

The corresponding asymptotic approximation (valid for sufficiently large n) of the true power function is then given by

$$\beta_{n,\alpha}(\theta) = 1 - \Phi\left(z_{1-\alpha} - \frac{\sqrt{n}(\theta - \theta_0)}{v}\right)$$

Note that in practice the (unknown) true value v^2 is generally replaced by an estimator $\hat{v}^2$ determined from the data. As long as $\hat{v}^2$ is a consistent estimator of v^2 this leads to the same asymptotic power function. The resulting test is asymptotically unbiased and consistent.

Usually there are many different possible estimators for a parameter θ. Consider an alternative estimator $\tilde{\theta}$ of θ satisfying

$$\sqrt{n}(\tilde{\theta} - \theta) \to_D N(0, \tilde{v}^2) \quad \text{as} \quad n \to \infty.$$

If the asymptotic variance v^2 of the estimator $\hat{\theta}$ is smaller than the asymptotic variance $\tilde{v}^2$ of $\tilde{\theta}$, i.e., $v^2 < \tilde{v}^2$, then $\hat{\theta}$ is a **more efficient** estimator of θ. Then necessarily the test based on $\hat{\theta}$ is **more powerful** than the test based on $\tilde{\theta}$, since asymptotically for all $\theta > \theta_0$

$$\begin{aligned}\tilde{\beta}_{n,\alpha}(\theta) &= 1 - \Phi\left(z_{1-\alpha} - \frac{\sqrt{n}(\theta - \theta_0)}{\tilde{v}}\right) \\ &< 1 - \Phi\left(z_{1-\alpha} - \frac{\sqrt{n}(\theta - \theta_0)}{v}\right) = \beta_{n,\alpha}(\theta)\end{aligned}$$

Example: Let $X_1, \ldots, X_n$ be an iid random sample. Consider testing $H_0 : \mu = \mu_0$ against $H_1 : \mu > \mu_0$, where $\mu := E(X_i)$. For a given level α the t-test then rejects H_0 if

$$T = \frac{\sqrt{n}(\bar{X} - \mu_0)}{S} > t_{n-1;1-\alpha},$$

where $t_{n-1;1-\alpha}$ is the $1 - \alpha$ quantile of a t-distributions with $n - 1$-degrees of freedom. This is an exact test if the distribution of X_i is normal. In the general case, the justification of the t-test is based on asymptotic arguments. Under some regularity conditions the central limit theorem implies that

$$\sqrt{n}(\bar{X} - \mu) \to_D N(0, \sigma^2) \quad \text{as} \quad n \to \infty$$

with $\sigma^2 = Var(X_i)$. Moreover, S^2 is a consistent estimator of σ^2 and $t_{n-1;1-\alpha} \to z_{1-\alpha}$ as $n \to \infty$. Thus even if the distribution of X_i is non-normal, for sufficiently large n, $T = \frac{\sqrt{n}(\bar{X}-\mu_0)}{S}$ is approximately $N(0,1)$-distributed and the asymptotic power function of the t-test is given by

$$\beta_{n,\alpha}(\theta) = 1 - \Phi\left(z_{1-\alpha} - \frac{\sqrt{n}(\mu - \mu_0)}{\sigma}\right).$$

2.5 Multiple Comparisons

In statistics, the multiple comparisons, multiplicity or multiple testing problem occurs when one considers a set of statistical inferences simultaneously or infers a subset of parameters selected based on the observed values. Errors in inference, including confidence intervals that fail to include their corresponding population parameters or hypothesis tests that incorrectly reject the null hypothesis are more likely to occur when one considers the set as a whole.

In empirical studies often dozens or even hundreds of tests are performed for the same data set. When **searching** for significant test results, one may come up with **false discoveries**.

Example: m different, independent test of significance level $\alpha > 0$. (Independence means that the test statistics used are mutually independent – this is usually not true in practice). Let's assume that a common null hypothesis H_0 holds for each of the m tests. Then

$$P\begin{pmatrix}\text{Type I error}\\ \text{by at least}\\ \text{one of the } m \text{ tests}\end{pmatrix} = 1 - (1-\alpha)^m =: \alpha_m > \alpha$$

Therefore, as m increases also the probability of a type I error increases:

Number of tests m	Probability of at least one type I error (α_m)
1	0.050
3	0.143
5	0.226
10	0.401
100	**0.994**

Analogous problem: Construction of m many $(1-\alpha)$ **confidence intervals**.

$$P\begin{pmatrix}\text{at least one of the } m \text{ confidence}\\ \text{intervals does not contain}\\ \text{the true parameter value}\end{pmatrix} = 1 - (1-\alpha)^m > \alpha$$

This represents the general problem of multiple comparisons. In practice, it will not be true that all considered test statistics are mutually independent. (This even complicates the problem.) However, we will still have the effect that the probability of at least one falsely significant result increases with the number m of tests, but it will not be equal to $1-(1-\alpha)^m$.

A statistically rigorous **solution** of this problem consists in modifying the constructions of tests or confidence intervals in order to arrive at **simultaneous tests**:

$$P\begin{pmatrix}\text{Type I error by}\\ \text{at least one of the } m \text{ tests}\end{pmatrix} \leq \alpha$$

or **simultaneous confidence intervals**:

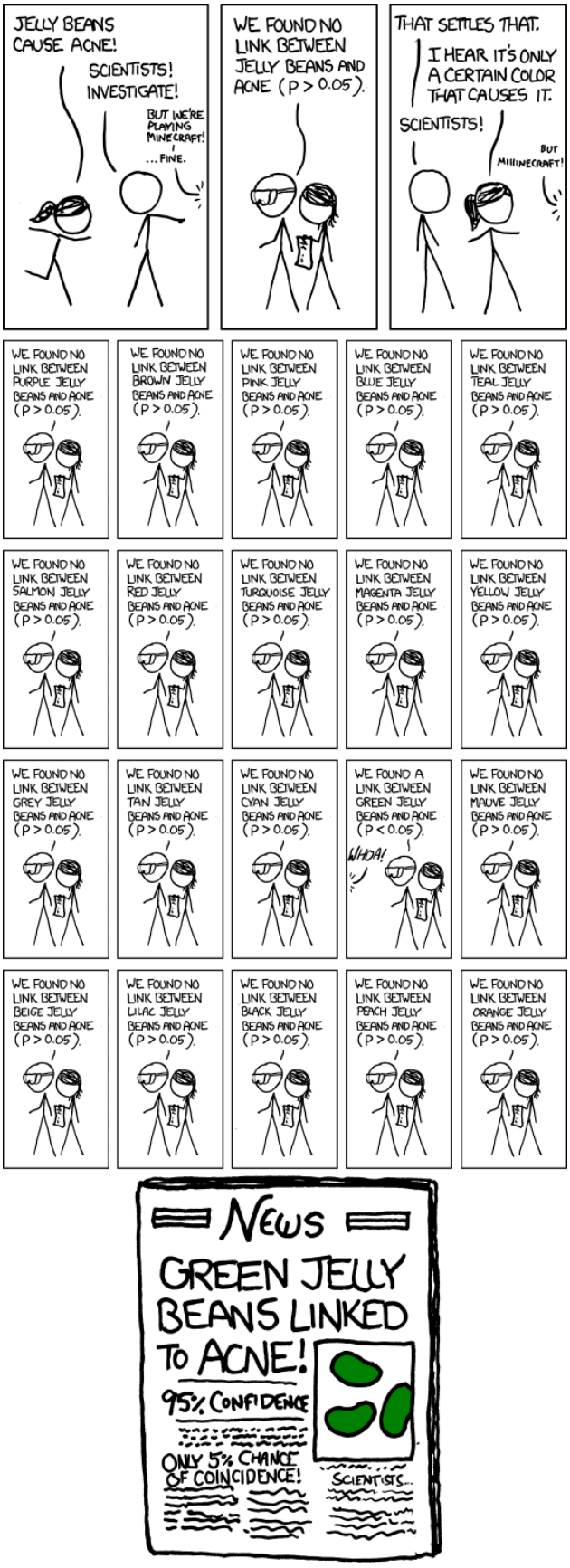


Figure 2.2: From: https://xkcd.com/882/

$$P\left(\begin{array}{c}\text{At least one of the } m \text{ confidence}\\ \text{intervals does not contain}\\ \text{the true parameter value}\end{array}\right) \leq \alpha$$

$$\Leftrightarrow \quad P\left(\begin{array}{c}\text{All confidence intervals}\\ \text{simultaneously contain the}\\ \text{true parameter values}\end{array}\right) \geq 1-\alpha$$

For certain problems (e.g., analysis of variance) there exist specific procedures for constructing simultaneous confidence intervals. However, the only generally applicable procedure seems to be the **Bonferroni correction**. It is based on Boole's inequality.

Theorem (Boole): Let $A_1, A_2, \ldots, A_m$ denote m different events. Then

$$P(A_1 \cup A_2 \cup \cdots \cup A_m) \leq \sum_{i=1}^{m} P(A_i).$$

This inequality also implies that:

$$P(A_1 \cap A_2 \cap \cdots \cap A_m) \geq 1 - \sum_{i=1}^{m} P(\bar{A}_i),$$

where $\bar{A}_i$ denotes the complementary event "not A_i".

Example: Bonferroni adjustment for m different tests of level $\alpha^* = \alpha/m$.

$$P\left(\begin{array}{c}\text{Type I error by}\\ \text{at least one of the } m \text{ tests}\end{array}\right) \leq \sum_{i=1}^{m} \alpha^* = \alpha$$

Analogously: Construction of m many $(1-\alpha^*)$-confidence intervals with $\alpha^* = \alpha/m$:

$$P\left(\begin{array}{c}\text{At least one of the } m \text{ confidence}\\ \text{intervals does not contain}\\ \text{the true parameter value}\end{array}\right) \leq \sum_{i=1}^{m} \alpha^* = \alpha$$

$$\Leftrightarrow \quad P\left(\begin{array}{c}\text{All confidence interval}\\ \text{simultaneously contain the}\\ \text{true parameter values}\end{array}\right) \geq 1 - \sum_{i=1}^{m} \alpha^* = 1-\alpha$$

Example: Regression analysis with $K = 100$ regressors, where none of the variables has an effect on the dependent variable y.

```
library("tidyverse", quietly = TRUE)
K <- 100
n <- 500

set.seed(123)

# Generate regression data, where none of the X-variables
# has an effect on the dependent variable Y:
my_df <- matrix(rnorm(n = n*K), nrow = n, ncol = K) %>%
  as_tibble %>%
```

```r
  mutate(Y = rnorm(n)) %>%
  select(Y, everything())

# OLS regression
OLS_result_df <- lm(Y ~ . , data = my_df) %>%
  summary %>%
  broom::tidy()


Count_Signif <- OLS_result_df %>%
  filter(term != '(Intercept)') %>%
  count(p.value < 0.05)
```

```
## # A tibble: 2 x 2
##   `p.value < 0.05`     n
##   <lgl>            <int>
## 1 FALSE               96
## 2 TRUE                 4
```

2.6 R-Lab: The Gauss-Test

Let's reconsider the simplest test statistic you will ever meet: The **Gauss-Test** (Or "Z-Test").

Setup: Let $X_1, \dots, X_n$ be an iid random sample with $X_i \sim N(\mu, \sigma^2)$ and $\sigma^2 < \infty$.

Idea: Under the above setup, $\bar{X}_n = n^{-1} \sum_{i=1}^n X_i$ consistently estimates the (unknown) true mean value μ. That is, $\bar{X}_n \to_p \mu$.

- Under the null hypothesis (i.e., $\mu_0 = \mu$), the difference $\bar{X}_n - \mu_0$ should be "small".
- Under the alternative hypothesis (i.e., $\mu_0 \neq \mu$), the difference $\bar{X}_n - \mu_0$ should be "large".

Under the null hypothesis H_0 we have that $\mu_0 = \mu$. Therefore:

$$Z = \frac{\sqrt{n}\,(\bar{X}_n - \mu_0)}{\sigma} = \underbrace{\frac{\sqrt{n}\,(\bar{X}_n - \mu)}{\sigma}}_{\sim N(0,1)}$$

Under the alternative H_1 we have that $\mu_0 \neq \mu$. Therefore:

$$\begin{aligned} Z &= \frac{\sqrt{n}\,(\bar{X}_n - \mu_0)}{\sigma} \\ &= \frac{\sqrt{n}\,(\bar{X}_n - \mu_0 + \mu - \mu)}{\sigma} \\ &= \frac{\sqrt{n}\,(\bar{X}_n - \mu)}{\sigma} + \frac{\sqrt{n}\,(\mu - \mu_0)}{\sigma} \sim N\left(\frac{\sqrt{n}\,(\mu - \mu_0)}{\sigma}, 1\right) \end{aligned}$$

The different distributions (under H_0 and H_1) of the test statistic Z can be investigated in the following dynamic plot:

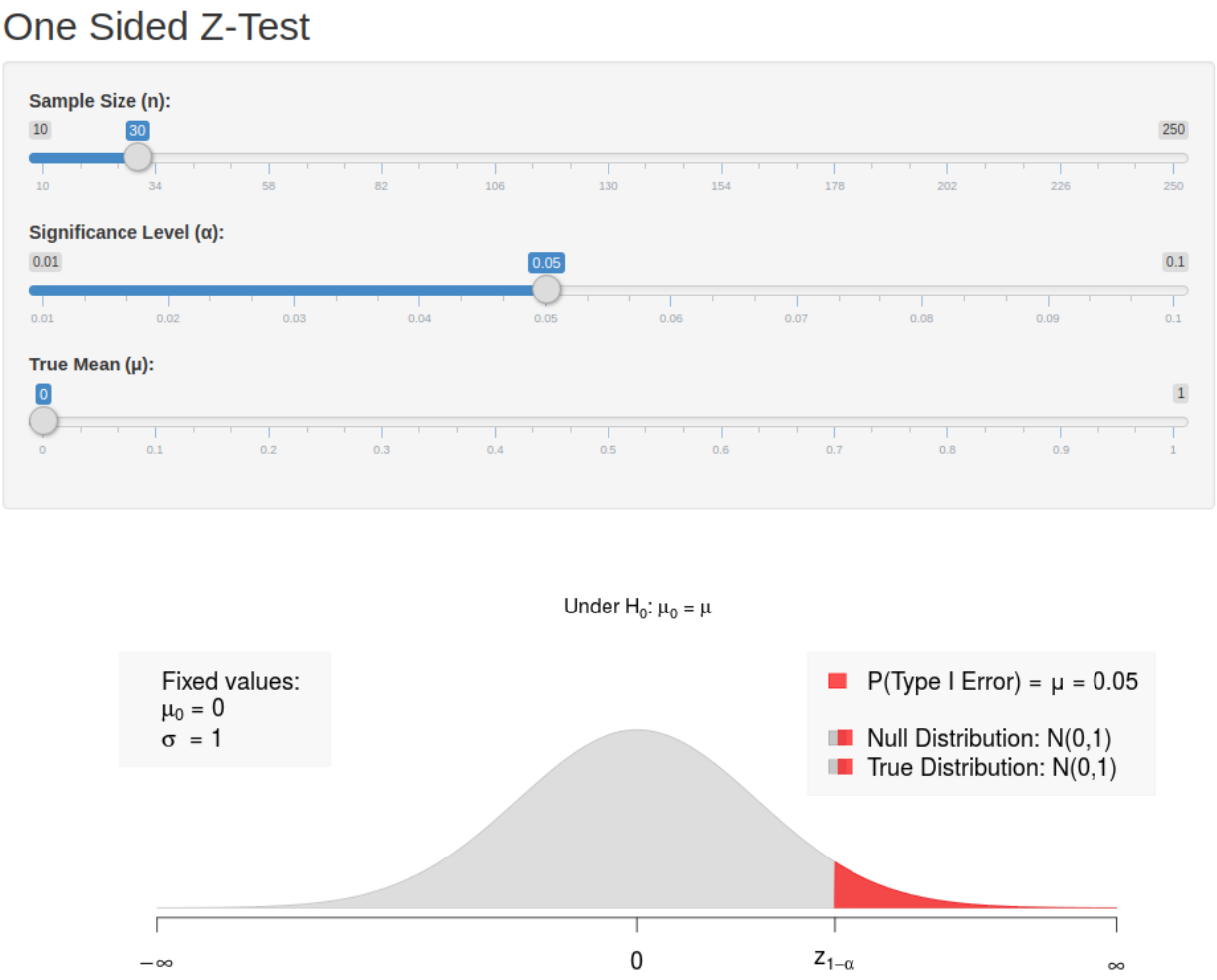


Figure 2.3: See: https://dliebl.shinyapps.io/Gauss-Test-Distr/

Chapter 3

Estimation Theory

3.1 Bias, Variance and MSE

Given a sample $X_1, \ldots, X_n$ consider an estimator $\widehat{\theta}_n \equiv \widehat{\theta}(X_1, \ldots, X_n)$ of a real-valued parameter $\theta \in \Omega \subset \mathbb{R}$.

The distribution of any estimator of course depends on the true parameter vector θ, i.e., more precisely,

$$\widehat{\theta}_n \equiv \widehat{\theta}(X_1, \ldots, X_n; \theta).$$

This dependence is usually not explicitly written, but all properties of estimators discussed below have to hold for all possible parameter values $\theta \in \Omega$. This will go without saying.

Statistical inference requires to assess the accuracy of an estimator.

The **bias** of an estimator is defined by

$$\text{Bias}(\widehat{\theta}_n) = E(\widehat{\theta}_n) - \theta$$

An estimator is called **unbiased** if $E(\widehat{\theta}_n) = \theta$ and hence $\text{Bias}(\widehat{\theta}_n) = 0$ (for all possible $\theta \in \Omega$).

The **variance** of an estimator is given by

$$\text{var}(\widehat{\theta}_n) = E\left((\widehat{\theta}_n - E(\widehat{\theta}_n))^2\right).$$

Performance of an estimator is most frequently evaluated with respect to the **quadratic loss** (also called L_2 loss)

$$(\widehat{\theta}_n - \theta)^2.$$

The corresponding risk is the **Mean Squared Error (MSE)**

$$E\left((\widehat{\theta}_n - \theta)^2\right) = \text{Bias}(\widehat{\theta}_n)^2 + \text{var}(\widehat{\theta}_n)$$

For an unbiased estimator the mean squared error is obviously equal to the variance of the estimator.

Example: Assume an i.i.d. sample $X_1, \ldots, X_n$ with mean $\mu = E(X_i)$ and variance $\sigma^2 = \text{var}(X_i) < \infty$.

- The sample mean $\bar{X}$ is an unbiased estimator of the true mean μ, since the equation

  $$E(\bar{X}) = \mu$$

  holds for any possible value of the true mean μ.
- The variance of the estimator $\bar{X}$ is given by

  $$\text{var}(\bar{X}) = \sigma^2/n$$

- The mean squared error of the estimator $\bar{X}$ is given by

  $$E\left((\bar{X} - \mu)^2\right) = \text{var}(\bar{X}) = \sigma^2/n$$

3.2 Consistency of Estimators

Asymptotic theory is concerned with theoretical results valid for "large sample sizes". Important keywords of asymptotic theory are:

- consistency
- rates of convergence
- asymptotic distributions

They all rely on elaborated concepts on the stochastic convergence of random variables.

Stochastic convergence. Let $\{Z_n\}_{n=1,2,3,...}$ be a sequence of **random variables**. Mathematically, there are different kinds of convergence of $\{Z_n\}$ to a fixed value c. The three most important are:

- Convergence in probability (abbreviated $Z_n \to_P c$):
$$\lim_{n\to\infty} P\left(|Z_n - c| > \epsilon\right) = 0 \quad \text{for all} \quad \epsilon > 0$$
- Almost sure convergence (abbreviated $Z_n \to_{a.s.} c$):
$$P\left(\lim_{n\to\infty} Z_n = c\right) = 1$$
- Convergence in quadratic mean (abbreviated $Z_n \to_{q.m.} c$):
$$\lim_{n\to\infty} E\left((Z_n - c)^2\right) = 0$$

Note that:

- $Z_n \to_{a.s.} c$ implies $Z_n \to_P c$
- $Z_n \to_{q.m.} c$ implies $Z_n \to_P c$

Consistency of estimators. Based on a sample $X_1, \ldots, X_n$ let $\hat{\theta}_n \equiv \theta_n(X_1, \ldots, X_n)$ be an estimator of an unknown parameter θ.

- $\hat{\theta}_n$ is called "weakly consistent" if
$$\hat{\theta}_n \to_P \theta \quad \text{as} \quad n \to \infty$$
- $\hat{\theta}_n$ is called "strongly consistent" if
$$\hat{\theta}_n \to_{a.s.} \theta \quad \text{as} \quad n \to \infty$$

Remark: For most statistical estimation problems it is usually possible to define many different estimators. The real problem is to find a good estimator which approximates the true parameter θ with the maximal possible accuracy. Consistency is generally seen as a necessary condition which has to be satisfied by any reasonable estimator. In econometric practice usually only weak consistency is derived which generally follows from weak **laws of large numbers**.

Example: Assume again an i.i.d. sample $X_1, \ldots, X_n$ with mean $\mu = E(X_i)$ and variance $\sigma^2 = \text{var}(X_i) < \infty$. As stated above we then have
$$E\left((\bar{X} - \mu)^2\right) = \text{var}(\bar{X}) = \sigma^2/n \to 0 \quad \text{as } n \to \infty.$$
Therefore, $\bar{X} \to_{q.m.} \mu$. The latter implies that $\bar{X} \to_P \mu$, i.e. $\bar{X}$ is a (weakly) consistent estimator of μ.

3.3 Rates of Convergence

Rates of convergence quantify the (stochastic) order of magnitude of an estimation error in dependence of the sample size n. This order of magnitude is usually represented using the symbols: O_P and o_P.

Let $\{Z_n\}_{n=1,2,3,...}$ be a sequence of random variables, and let $\{c_n\}_{n=1,2,3,...}$ be a sequence of positive numbers (c_n may de a deterministic sequence of real numbers or it may be a sequence of random variables).

- We will write $Z_n = O_p(c_n)$ if for any $\epsilon > 0$ there exist numbers $0 < M < \infty$ and m such that

$$P(|Z_n| \geq M \cdot c_n) \leq \epsilon \quad \text{for all} \quad n \geq m.$$

- We will write $Z_n = o_p(c_n)$ if

$$\lim_{n\to\infty} P(|Z_n| \geq \epsilon \cdot c_n) = 0 \quad \text{for all} \quad \epsilon > 0.$$

- With $c_n = 1$ for all n, $Z_n = O_p(1)$ means that the sequence $\{Z_n\}$ is **stochastically bounded**. I.e., for any $\epsilon > 0$ there exist number $0 < M < \infty$ and m such that

$$P(|Z_n| \geq M) \leq \epsilon \quad \text{for all} \quad n \geq m.$$

- With $c_n = 1$ for all n, $Z_n = o_P(1)$ is equivalent to $Z_n \to_P 0$, i.e., Z_n **converges in probability to zero**.

Note that:

- $Z_n = O_p(c_n)$ is equivalent to $Z_n/c_n = O_p(1)$
- $Z_n = o_p(c_n)$ is equivalent to $Z_n/c_n = o_p(1)$

Definition: An estimator $\hat{\theta} \equiv \hat{\theta}_n$ of a parameter θ possesses the **rate of convergence** n^{-r} if and only if r is the *largest positive number* with the property that

$$|\hat{\theta}_n - \theta| = O_P(n^{-r}).$$

The rate of convergence quantifies how fast the estimation error decreases when increasing the sample size n.

Unbiased estimators: Let $\hat{\theta}_n$ be an *unbiased* estimator of an unknown parameter θ satisfying $\text{var}(\hat{\theta}_n) = Cn^{-1}$ for some $0 < C < \infty$. Then $\hat{\theta}_n$ possesses the rate of convergence $n^{-1/2}$. This is a consequence of the Chebyshev inequality.

Chebyshev inequality: If Z denotes a random variable with mean μ and variance σ^2, then

$$P\left(|X - \mu| > \sigma \cdot m\right) \leq \frac{1}{m^2} \quad \text{for all} \quad m > 0$$

$$\Rightarrow P\left(|\hat{\theta}_n - \theta| > n^{-1/2}\sqrt{C} \cdot \frac{1}{\sqrt{\epsilon}}\right) \leq \epsilon \quad \text{for all} \quad \epsilon > 0$$

Generalization: Let $\hat{\theta}_n$ be a *not necessarily unbiased* estimator of an unknown parameter θ. If $E\left((\hat{\theta}_n - \theta)^2\right) = Cn^{-2r}$ for some $0 < C < \infty$, then $|\hat{\theta}_n - \theta| = O_P(n^{-r})$ and n^{-r} is the rate of convergence of $\hat{\theta}_n$.

Example: Assume an i.i.d. sample $X_1, \ldots, X_n$ with mean $\mu = E(X_i)$ and variance $\sigma^2 = \text{var}(X_i) < \infty$. The sample mean $\bar{X}$ $(\equiv \bar{X}_n)$ is an unbiased estimator of μ with variance $\text{var}(\bar{X}) = \sigma^2/n$. For large n we have by the central limit theorem that approximately $\sqrt{n}(\bar{X} - \mu) \sim N(0, \sigma^2)$. Therefore, for example:

- with $\epsilon = 0.05$ we obtain
$$P\left(|\bar{X}_n - \mu| \geq 1.96\sigma \cdot n^{-1/2}\right) = 0.05$$
- with $\epsilon = 0.01$ we obtain
$$P\left(|\bar{X}_n - \mu| \geq 2.64\sigma \cdot n^{-1/2}\right) = 0.01.$$

Generalizing this argument for all possible $\epsilon > 0$ we can conclude that $\bar{X} - \mu = O_P(n^{-1/2})$. On the other hand for any $r > 1/2$ we have $n^{-r}/n^{-1/2} \to 0$ as $n \to \infty$. Hence, for any constant $c > 0$

$$P\left(|\bar{X}_n - \mu| \geq c\sigma \cdot n^{-r}\right) =$$
$$= P\left(|\bar{X}_n - \mu| \geq (c\sigma \cdot n^{-1/2}) \cdot \frac{n^{-r}}{n^{-1/2}}\right) \to 1 \quad \text{as} \quad n \to \infty.$$

Therefore $n^{-1/2}$ is the **rate of convergence** of $\bar{X}$.

Note that:

- Maximum-likelihood estimators of an unknown parameter usually possess the rate of convergence $n^{-1/2}$ (there are exceptions!).
- The situation is different, for instance, in nonparametric curve estimation problems. For example kernel estimators (of a density or regression function) only achieve the rate of convergence $n^{-2/5}$.
- The rate of convergence is an important criterion for selecting the best possible estimator for a given problem. For most parametric problems it is well known that the optimal (i.e. fastest possible) convergence rate is $n^{-1/2}$. In nonparametric regression or density estimation the optimal convergence rate is only $n^{-2/5}$, if the underlying function is twice continuously differentiable.

O_P-rules:

- We have
$$Z_n \to_P Z \qquad \text{if and only if} \qquad Z_n = Z + o_p(1)$$
This follows from $Z_n = Z + (Z_n - Z)$ and $Z_n - Z \to_P 0$.
- If $Z_n = O_P(n^{-\delta})$ for some $\delta > 0$, then $Z_n = o_P(1)$
- If $Z_n = O_P(r_n)$, then $Z_n^\delta = O_P(r_n^\delta)$ for any $\delta > 0$. Similarly, $Z_n = o_P(r_n)$ implies $Z_n^\delta = o_P(r_n^\delta)$ for any $\delta > 0$.
- If $Z_n = O_P(r_n)$ and $V_n = O_P(s_n)$, then
$$Z_n + V_n = O_P(\max\{r_n, s_n\})$$
$$Z_n V_n = O_P(r_n s_n)$$
- If $Z_n = o_P(r_n)$ and $V_n = O_P(s_n)$, then $Z_n V_n = o_P(r_n s_n)$
- If $E(|Z_n|^k) = O(r_n)$, then $Z_n = O_p(|r_n|^{1/k})$ for $k = 1, 2, 3, \ldots$

3.4 Asymptotic Distributions

The practically most important version of stochastic convergence is convergence in distribution. Knowledge about the "asymptotic distribution" of an estimator allows to construct confidence intervals and tests.

Definition: Let Z_n be a sequence of random variables with corresponding distribution functions G_n. Then Z_n **converges in distribution** to a random variable Z with distribution function G, if

$$G_n(x) \to G(x) \quad \text{as} \quad n \to \infty$$

at all continuity points x of G (abbreviated: $Z_n \to_L Z$ or $Z_n \to_L G$ or "$\to_D$" instead of "$\to_L$").

In a vast majority of practically important situation the limiting distribution is the normal distribution. One then speaks of **asymptotic normality**. Asymptotic normality is usually a consequence of central limit theorems. The simplest result in this direction is the central limit theorem of Lindeberg-Levy.

Theorem (Lindeberg-Levy) Let $Z_1, Z_2, \ldots$ be a sequence of i.i.d. random variables with finite mean μ and variance $\sigma^2 < \infty$. Then

$$\sqrt{n}\left(\frac{1}{n}\sum_{i=1}^{n} Z_i - \mu\right) \to_L N(0, \sigma^2).$$

Example: Let $X_1, \ldots, X_n$ be independent random variables with $E(X_i) = \mu$, $Var(X_i) = \sigma^2$. Then the central limit theorem of Lindeberg-Levy implies that

$$\sqrt{n}(\bar{X} - \mu) \to_L N(0, \sigma^2) \quad \text{or equivalently} \quad \frac{\sqrt{n}(\bar{X} - \mu)}{\sigma} \to_L N(0, 1).$$

We can conclude that $\bar{X}$ is an "asymptotically normal estimator" of μ. If n is sufficiently large, then $\bar{X}$ is approximatively normal with mean μ and variance σ^2/n. Frequently used notations:

- $\bar{X} \sim AN(\mu, \sigma^2/n)$
- $\bar{X} \overset{a}{\sim} N(\mu, \sigma^2/n)$

Most estimators $\hat{\theta}_n$ used in parametric and nonparametric statistics are asymptotically normal. In parametric problems (with rate of convergence $n^{-1/2}$) one usually obtains

$$\sqrt{n}(\hat{\theta}_n - \theta) \to_L N(0, v^2),$$

where v^2 is the asymptotic variance of the estimator (often, but not necessarily, $v^2 = \lim_{n\to\infty} n \cdot \mathrm{var}(\hat{\theta}_n)$).

Multivariate generalization: The above concepts are easily generalized to estimators $\hat{\theta}_n$ of a multivariate parameter vector $\theta \in \mathbb{R}^p$. Consistency and rates of convergence then have to be derived separately for each element of the vector. Convergence in distribution is defined via convergence of the multivariate distribution functions. For standard estimators (e.g., maximum likelihood) in parametric problems one usually obtains

$$\sqrt{n}(\hat{\theta}_n - \theta) \to_L N_p(0, V),$$

where V is the asymptotic covariance matrix (usually, $V = \lim_{n\to\infty} n \cdot \mathrm{Cov}(\hat{\theta}_n)$).

Multivariate normality holds if and only if for any vector $c = (c_1, \ldots, c_p)' \in \mathbb{R}^p$ with $\sum_{j=1}^{p} c_j^2 = \|c\|_2^2 = 1$

$$\sqrt{n}\left(\sum_{j=1}^{p} c_j(\hat{\theta}_{jn} - \theta_j)\right) = \sqrt{n}\left(c'\hat{\theta}_n - c'\theta\right) \to_L N\left(0, v_c^2\right),$$

where

$$v_c^2 = c'Vc = \sum_{j=1}^{p}\sum_{k=1}^{p} c_j c_k V_{jk},$$

and where V_{jk} are the elements of the asymptotic covariance matrix V.

This condition is frequently called **"Cramer-Wold device"**. Using one-dimensional central limit theorems it can be verified for any vector c.

Example: Let $X_1 = (X_{11}, X_{12})', \ldots, X_n = (X_{n1}, X_{n2})'$ be i.i.d. two-dimensional random vectors with $E(X_i) = \mu = (\mu_1, \mu_2)'$ and $Cov(X_i) = \Sigma$. The Cramer-Wold device and Lindeberg-Levy's central limit theorem then imply that

$$\sqrt{n}\left(\bar{X} - \mu\right) \to_L N_2\left(0, \Sigma\right).$$

Note that asymptotic normality usually also holds for nonparametric curve estimators with convergence rates slower than $n^{-1/2}$.

3.5 Asymptotic Theory

Many estimation procedures in modern statistics rely on fairly general assumptions. For a given sample size n it is then often impossible to derive the exact distribution of θ_n. Necessary calculations are too complex, and finite sample distributions usually depend on unknown characteristics of the distribution of the underlying data.

The goal of asymptotic theory then is to derive reasonable approximations. For **large samples** such approximations are of course very accurate, for **small samples** there may exist a considerable approximation error. Therefore, for small samples the approximation quality of asymptotic approximations is usually studied by Monte-Carlo approximations.

Asymptotic theory is used in order to select an appropriate estimation procedure in complex situations. The idea is to determine the estimator which, at least for large sample sizes, provides the smallest possible estimation error. This leads to the concept of "asymptotically efficient" estimators.

Properties of an asymptotically efficient estimator θ_n:

- For the estimation problem to be considered θ_n is consistent and adopts the fastest possible rate of convergence (generally:$n^{-1/2}$ in parametric statistics, $n^{-2/5}$ in nonparametric curve estimation problems).
- In most regular situations one is additionally interest in a "best asymptotically normal" (BAN) estimator. Assume that $\sqrt{n}(\theta_n - \theta) \sim N(0, v^2)$. Then θ_n is a BAN-estimator if any alternative estimator $\tilde{\theta}_n$ with $\sqrt{n}(\tilde{\theta}_n - \theta) \sim N(0, \tilde{v}^2)$ possesses a larger asymptotic variance, i.e. $\tilde{v}^2 \geq v^2$.
- **Multivariate generalization:** An estimator θ_n with $\sqrt{n}(\theta_n - \theta) \sim N_p(0, V)$ is best asymptotically normal if
$$c'\tilde{V}c \geq c'Vc \quad \text{for all} \quad c \in \mathbb{R}^p, \|c\|_2^2 = 1$$
for any other estimator $\tilde{\theta}_n$ satisfying $\sqrt{n}(\tilde{\theta}_n - \theta) \sim N_p(0, \tilde{V})$.

For most estimation problems in parametric statistics maximum-likelihood estimators are best asymptotically normal.

3.6 Mathematical tools

3.6.1 Taylor expansions

Taylors' theorem: Let f be a real-valued function which is $k+1$ continuously differentiable in the interior of an interval $[a, b]$. Consider a point $x_0 \in (a, b)$. For any other value $x \in (a, b)$ there exists some $\psi \in [x_0, x]$ such that
$$f(x) = f(x_0) + \sum_{r=1}^{k} \frac{1}{r!} f^{(r)}(x_0) \cdot (x - x_0)^r + \frac{1}{(k+1)!} f^{(k+1)}(\psi) \cdot (x - x_0)^{k+1}$$

Qualitative version of Taylors' formula:
$$f(x) = f(x_0) + \sum_{r=1}^{k} \frac{1}{r!} f^{(r)}(x_0) \cdot (x - x_0)^r + O((x - x_0)^{k+1})$$

Example: Let $f(x) = ln(x)$ und $x_0 = 1 \Rightarrow f'(x_0) = 1$, $f''(x_0) = -1$.

First order Taylor approximation: $f(x) = \tilde{f}(x) + O((x - x_0)^2)$, where $\tilde{f}(x) = x - x_0$

- $x = 1.05 \Rightarrow f(x) = 0.04879$, $\tilde{f}(x) = 0.05$ and $|f(x) - \tilde{f}(x)| = 0.00121$
- $x = 1.1 \Rightarrow f(x) = 0.09531$, $\tilde{f}(x) = 0.1$ and $|f(x) - \tilde{f}(x)| = 0.00469$

- $x = 1.5 \Rightarrow f(x) = 0.40546,\ \tilde{f}(x) = 0.5$ and $|f(x) - \tilde{f}(x)| = 0.09454$
- $x = 2 \Rightarrow f(x) = 0.69315,\ \tilde{f}(x) = 1$ and $|f(x) - \tilde{f}(x)| = 0.30685$

Second order Taylor approximation: $f(x) = \tilde{f}(x) + O((x - x_0)^3)$, where $\tilde{f}(x) = x - x_0 - \frac{1}{2}(x - x_0)^2$

- $x = 1.05 \Rightarrow f(x) = 0.04879,\ \tilde{f}(x) = 0.04875$ and $|f(x) - \tilde{f}(x)| = 0.00004$
- $x = 1.1 \Rightarrow f(x) = 0.09531,\ \tilde{f}(x) = 0.95$ and $|f(x) - \tilde{f}(x)| = 0.00031$
- $x = 1.5 \Rightarrow f(x) = 0.40546,\ \tilde{f}(x) = 0.375$ and $|f(x) - \tilde{f}(x)| = 0.03046$
- $x = 2 \Rightarrow f(x) = 0.69315,\ \tilde{f}(x) = 0.5$ and $|f(x) - \tilde{f}(x)| = 0.19315$

Multivariate generalization: $x_0, x \in \mathbb{R}^p$, $f'(x_0) \in \mathbb{R}^p$, $f''(x_0)$ a $p \times p$ Matrix.

First order Taylor approximation:

$$f(x) = f(x_0) + f'(x_0) \cdot (x - x_0) + O(\|x - x_0\|_2^2)$$

Second order Taylor approximation:

$$f(x) = f(x_0) + f'(x_0) \cdot (x - x_0) + \frac{1}{2}(x - x_0)^T f''(x_0)(x - x_0) + O(\|x - x_0\|_2^3)$$

3.6.2 Tools for deriving asymptotic distributions

Let $\{W_n\}$, $\{Z_n\}$ be sequences of random variables, then:

- $Z_n = W_n + o_P(1) \quad \Leftrightarrow \quad Z_n - W_n \to_P 0$. If additionally $W_n \to_L N(0, v^2)$ then $Z_n \to_L N(0, v^2)$.
- For any fixed constant $c \neq 0$: If $Z_n \to_P c$ and $W_n \to_L N(0, v^2)$, then

  $$cW_n \to_L N(0, c^2 v^2) \quad \text{as well as} \quad V_n := Z_n \cdot W_n \to_L N(0, c^2 v^2).$$

  Furthermore, If Z_n and c are positive (with probability 1) then also

  $$W_n / c \to_L N(0, v^2/c^2) \quad \text{as well as} \quad V_n := W_n / Z_n \to_L N(0, v^2/c^2).$$

- Multivariate generalization (C, Z_n $p \times p$ matrices; W_n p-dimensional random vectors): If $Z_n \to_P C$ as well as $W_n \to_L N_p(0, V)$, then

  $$CW_n \to_L N_p(0, CVC') \quad \text{as well as}$$
  $$V_n := Z_n \cdot W_n \to_L N_p(0, CVC')$$

3.6.3 The Delta-Method

A further tool which is frequently used in asymptotic statistics is the so-called delta-method.

Delta-Method: Let $\widehat{\theta}_n$ be a sequence of estimators of a one-dimensional parameter θ satisfying $n^r(\widehat{\theta}_n - \theta) \to_L N(0, v^2)$, and let $g(.)$ be a real-valued function which is continuously differentiable at θ and satisfies $g'(\theta) \neq 0$. Then

$$n^r \left(g(\widehat{\theta}_n) - g(\theta) \right) \to_L N\left(0, g'(\theta)^2 v^2\right).$$

Example: Assume an i.i.d. sample $X_1, \ldots, X_n$ from an exponential distribution, i.e., the underlying density of X_i is given by $f(x|\theta) = \theta \exp(-\theta x)$. We then have $\mu := E(X_i) = 1/\theta$ as well as $\sigma_X^2 := \text{var}(X_i) = 1/\theta^2$. The underlying parameter $\theta > 0$ is unknown and has to be estimated from the data.

The maximum-likelihood estimator of θ is $\hat{\theta} = 1/\bar{X}$.

We know that $\sqrt{n}(\bar{X} - \frac{1}{\theta}) \to_L N(0, \frac{1}{\theta^2})$, but what's about the distribution of $1/\bar{X}$? For this purpose the delta-method can be applied with $g(x) = 1/x$. Then $g'(x) = -1/x^2$, $g'(1/\theta) = -\theta^2$, and consequently

$$n^{1/2} \left(\frac{1}{\bar{X}} - \theta \right) = n^{1/2} \left(g\left(\bar{X}\right) - g\left(\frac{1}{\theta}\right) \right) \to_L N\left(0, \theta^2\right).$$

Chapter 4

Linear Regression

The lecture script of this chapter can be found HERE.

Chapter 5

Monte-Carlo Simulations

5.1 Checking Test Statistics

5.1.1 Simple Example: Gauss Test

First, we repeat some of the things we already know about the Gauss test statistic from Chapter.

Let us consider the simple case of the *one-sided* Gauss (or Z) test statistic under the following following setup:

- $X_1, \ldots, X_n$ i.i.d. random sample with $X_i \sim N(\mu_X, \sigma_X^2)$ and $\sigma_X^2 = 1$
- $\alpha = 0.05$ (Significance level)
- $n \in \{15, 30, 50\}$ (Different sample sizes)
- $\mu_{X,0} = 0$, i.e., $\Omega_0 = \{0\}$ and $\Omega_1 =]0, \infty[$.

Under the above setup, we know the **theoretical power function:**

$$\begin{aligned} \beta^Z_{n,\alpha}(\mu_X) &= \mathbb{P}(Z \geq z_{1-\alpha}) \\ &= 1 - \mathbb{P}(Z < z_{1-\alpha}) \\ &= 1 - \Phi_{\mu_Z, \sigma_Z^2}(z_{1-\alpha}), \end{aligned}$$

where Φ_{μ_Z, σ_Z^2} denotes the distribution function of a Gaussian distribution with mean and variance:

$$\begin{aligned} \mu_Z &= \frac{\sqrt{n}(\mu_X - \mu_{X,0})}{\sigma_X} = \sqrt{n}(\mu_X - 0) \\ \sigma_Z^2 &= 1. \end{aligned}$$

Furthermore, $z_{1-\alpha} = z_{0.95}$ is the 95% quantile of a standard normal distribution.

Computation in R: This is how you can use R in order to compute $\beta^Z_{n,\alpha}(\mu_X) = 1 - \Phi_{\mu_Z,1}(z_{1-\alpha})$:

```
Gauss.beta <- function(n,              # sample size
                       alpha=0.05,     # significance level
                       mu.X.true=0,    # The true mean of X_i
                       mu.X.null=0,    # The null-hypothesis mean of X_i
                       var.X    =1     # The assumed known var of X_i
                       ){
  ## Critical value:
  z_crit <- qnorm(1-alpha, mean=0, sd=1)
  ## Power:
  Phi <- pnorm(q    = z_crit,
               mean = sqrt(n)*(mu.X.true - mu.X.null)/sqrt(var.X),
```

```
                sd   = 1)
  power <- 1- Phi
  ## Return result:
  return(power)
}
```

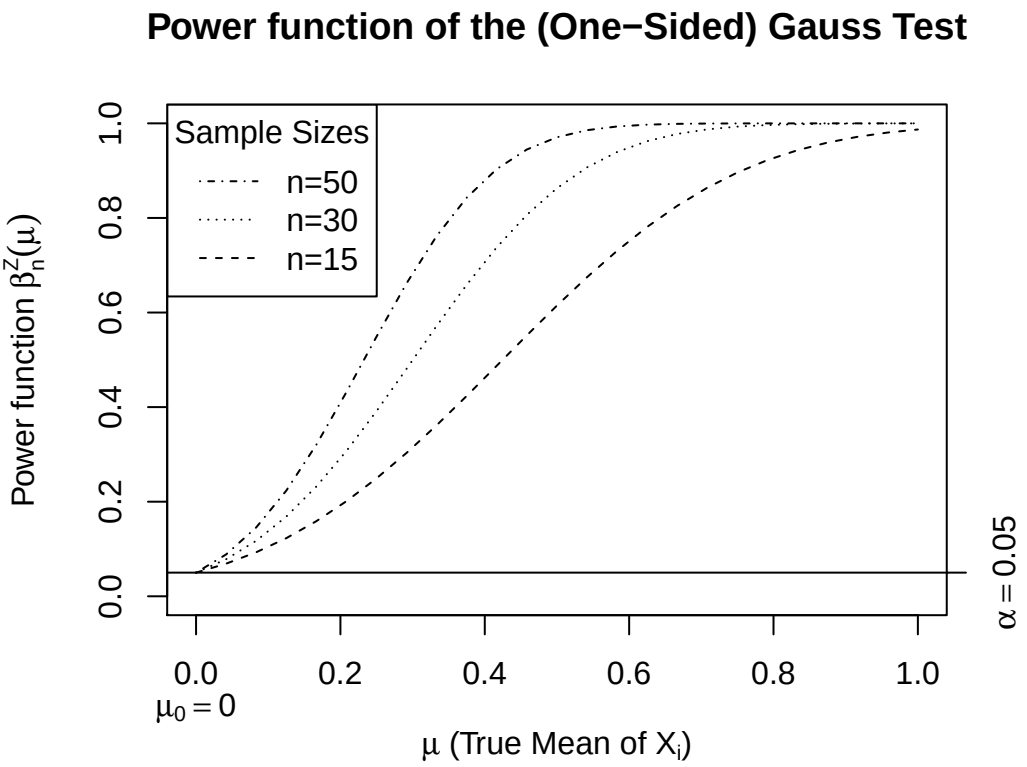
For our purposes, it is convenient to vectorize the `Gauss.beta()` function with respect to its argument `mu.X.true`:

```
## Vectorization with respect to the argument `mu.X.true`:
Gauss.beta <- Vectorize(FUN=Gauss.beta, vectorize.args = "mu.X.true")
```

Plot: The function `Gauss.beta()` allows us now to easily produce a plot of the trajectories of the power function $\beta^Z_{n,0.05}(\mu_X)$ for $\mu_X \in \Omega_0 \cup \Omega_1$ and for the different sample sizes $n \in \{15, 30, 50\}$.

Here is the R-Code to do this:

```
## Sequence of different mu_X values (here from 0 to 1):
mu.X.true.seq   <- seq(0,1,len=25)

## Trajectories of the power function for different sample sizes:
## n=15
beta.n.15       <- Gauss.beta(n=15, mu.X.true=mu.X.true.seq)
## n=30
beta.n.30       <- Gauss.beta(n=30, mu.X.true=mu.X.true.seq)
## n=50
beta.n.50       <- Gauss.beta(n=50, mu.X.true=mu.X.true.seq)

## Plot
par(mar=c(5.1,4.1+1,4.1,2.1))
plot(y=0, x=0, type="n",
     ylim=c(0,1),
     xlim=range(mu.X.true.seq),
     xlab=expression(paste(mu," (True Mean of ", X[i],")")),
     ## Labels:
     ylab=expression(paste("Power function ",beta[n]^Z,(mu))),
     main="Power function of the (One-Sided) Gauss Test")
## Null-hypothesis mean:
mtext(text = expression(mu[0]==0), side = 1, line = 2, at = 0)
## Trajectories:
lines(y=beta.n.15, x=mu.X.true.seq, lty=2)
lines(y=beta.n.30, x=mu.X.true.seq, lty=3)
lines(y=beta.n.50, x=mu.X.true.seq, lty=4)
## Significance Level:
axis(4, at=0.05, labels = expression(alpha==0.05))
abline(h=0.05, lty=1)
## Legend:
legend("topleft", title = "Sample Sizes",
       legend = c("n=50","n=30","n=15"),
       lty=c(4:2))
```



5.1.2 Simulated Power Function

The power function is best suited to compare several test statistics with each other. Very often, however, it is impossible to compute the power function $\beta_{n,\alpha}(\theta)$ analytically. (The Gauss test is a rare exception.) The reason for this is that we often know only the distribution of a test statistic under the null hypothesis, but not under the alternative hypothesis. In fact, things can be even worse: Very often, we only know the **asymptotic distribution** of a test statistic under the null hypothesis. That is, the null distribution is only known for the limiting case of $n \to \infty$.

Solution: Use **Monte-Carlo Simulations** in order approximate the power function.

For the sake of simplicity let's approximate the power function $\beta_{n,\alpha}^Z(\theta)$ of the one-sided Gauss-Test. This has the (didactic) advantage that we can compare our MC-approximated power function with the theoretical power function.

General Idea: MC-Simulations make use of the **Law of Large Numbers**. For instance, by the Strong Law of Large Numbers we know that the empirical mean

$$\bar{X}_m \to_{(a.s)} \mu_X$$

converges almost surely (a.s.) to the desired limit $\mathbb{E}(X) = \mu_X$ as $m \to \infty$. The only prerequisites are that X has finite first moments, i.e., $\mathbb{E}(X) = \mu_X < \infty$, and that $\bar{X}_m$ is constructed from an i.i.d. sample $X_1, \ldots, X_m$. That is, MC-Simulations use averages in order to approximate mean values.

Approximating a Power Function (Theory):

Remember that

$$\beta_{n,\alpha}^Z(\mu_X) \quad = \quad \mathbb{P}(Z \geq z_{1-\alpha}).$$

Let us rewrite this probability using the following binary random variable:

$$V = 1_{(Z \geq z_{1-\alpha})},$$

where $1_{(\text{TRUE})} = 1$ and $1_{(\text{FALSE})} = 0$. Then we have that

$$\beta^Z_{n,\alpha}(\mu_X) \quad = \quad \mathbb{P}(Z \geq z_{1-\alpha}) \; = \; \mathbb{E}(V),$$

since

$$\mathbb{E}(V) \quad = \quad \underbrace{\mathbb{P}(V=1)}_{\mathbb{P}(Z \geq z_{1-\alpha}) \cdot 1} + \underbrace{\mathbb{P}(V=0) \cdot 0}_{=0}.$$

Now we have an expression for the power function $\beta^Z_{n,\alpha}(\mu_X)$ in terms of the population mean $\mathbb{E}(V)$.

By the **Law of Large Numbers** we know that we can use averages of i.i.d. random variables in order to approximate their population mean. That is,

$$\frac{1}{m}\sum_{j=1}^{m} V_j \to_{(a.s)} \mathbb{E}(V) \quad \text{as} \quad m \to \infty,$$

where $(V_1, \ldots, V_m)$ is an iid random sample with $V_j \sim V = 1_{(Z \geq z_{1-\alpha})}$. This approximation can be made **arbitrarily accurate** as $m \to \infty$.

The MC-Simulation proceeds as following:

1. Choose a large number m, for instance, $m = 50,000$.
2. Generate realizations
$$(v_1, \ldots, v_m)$$
from the MC random sample
$$(V_1, \ldots, V_m) = (1_{(Z_1 \geq z_{1-\alpha})}, \ldots, 1_{(Z_m \geq z_{1-\alpha})})$$
3. Approximate $\mathbb{E}(V) = \beta^Z_{n,\alpha}(\mu_X)$ using the empirical means.
$$\frac{1}{m}\sum_{j=1}^{m} v_j.$$

Approximating a Power Function (Practice):

Let's start with approximating only the following value of the power function for the one-sided Gauss-Test:

$$\beta^Z_{n=15,\alpha=0.05}(0.5) = 0.6147.$$

First, we set the Monte-Carlo sample size to $m = 50,000$.

Second, we need a realization from the random sample

$$(1_{(Z_1 \geq z_{1-\alpha})}, \ldots, 1_{(Z_m \geq z_{1-\alpha})}).$$

In R you can do this as following:

```
set.seed(1009)
## Setup:
n         <- 15    # Sample Size
alpha     <-  0.05 # Significance Level
mu.X.true <-  0.5  # The (usually unknown) true mean of X_i
mu.X.null <-  0    # The null-hypothesis mean of X_i
var.X     <-  1    # The (assumed known)  var of X_i

## Critical value:
z_crit <- qnorm(1-alpha, mean=0, sd=1)
```

```
## Number of Monte-Carlo Repetitions:
m         <- 50000

## Container for Z-realizations:
Z         <- rep(NA, m)

## MC-Experiments:
for(j in 1:m){
  ## Generate X-sample:
  X.sample <- rnorm(n=n, mean=mu.X.true, sd=sqrt(var.X))
  ## Compute jth realization of Z:
  Z[j]     <- sqrt(n)*(mean(X.sample) - mu.X.null)/sqrt(var.X)
}

## Check Z>=z_crit:
head(Z >= z_crit)
```

```
## [1]  TRUE  TRUE  TRUE FALSE FALSE  TRUE
```

```
head(as.numeric(Z >= z_crit))
```

```
## [1] 1 1 1 0 0 1
```

Third, we need to compute the average

$$\frac{1}{m}\sum_{j=1}^{m} 1_{(Z_j \geq z_{1-\alpha})}$$

with respect to the simulated realizations $1_{(Z_1 \geq z_{1-\alpha})}, \ldots, 1_{(Z_m \geq z_{1-\alpha})}$.

In R you can do this as following:

```
## MC-Approximated Power:
MC_power_n15_mu0.5 <- mean(Z >= z_crit)
MC_power_n15_mu0.5
```

```
## [1] 0.6139
```

Observe that this approximation is really close to the true value: $\beta^{Z}_{n=15,\alpha=0.05}(0.5) = 0.6147 =$ `Gauss.beta(n=15, mu.X.true=0.5)`.

We can write all this as a practical R function `Gauss.MC.beta()`:

```
Gauss.MC.beta <- function(
    n         = 15,    # Sample Size
    alpha     =  0.05, # Significance Level
    mu.X.true =  0.5,  # The (usually unknown) true mean of X_i
    mu.X.null =  0,    # The null-hypothesis mean of X_i
    var.X     =  1,    # The (assumed known)  var of X_i
    ##
    m         = 50000  # Number of Monte-Carlo Repetitions:
    ){

  ## Critical value:
  z_crit <- qnorm(1-alpha, mean=0, sd=1)
  ## Container for Z-realizations:
  Z         <- rep(NA, m)
```

```
  ## MC-Experiments:
  for(j in 1:m){
    ## Generate X-sample:
    X.sample <- rnorm(n=n, mean=mu.X.true, sd=sqrt(var.X))
    ## Compute jth realization of Z:
    Z[j]     <- sqrt(n)*(mean(X.sample) - mu.X.null)/sqrt(var.X)
  }
  ## MC-Approx Power
  MC.power <- mean(c(as.numeric(Z >= z_crit)))
  ##
  return(MC.power)
}

## Vectorization:
Gauss.MC.beta <- Vectorize(FUN=Gauss.MC.beta, vectorize.args = "mu.X.true")
```

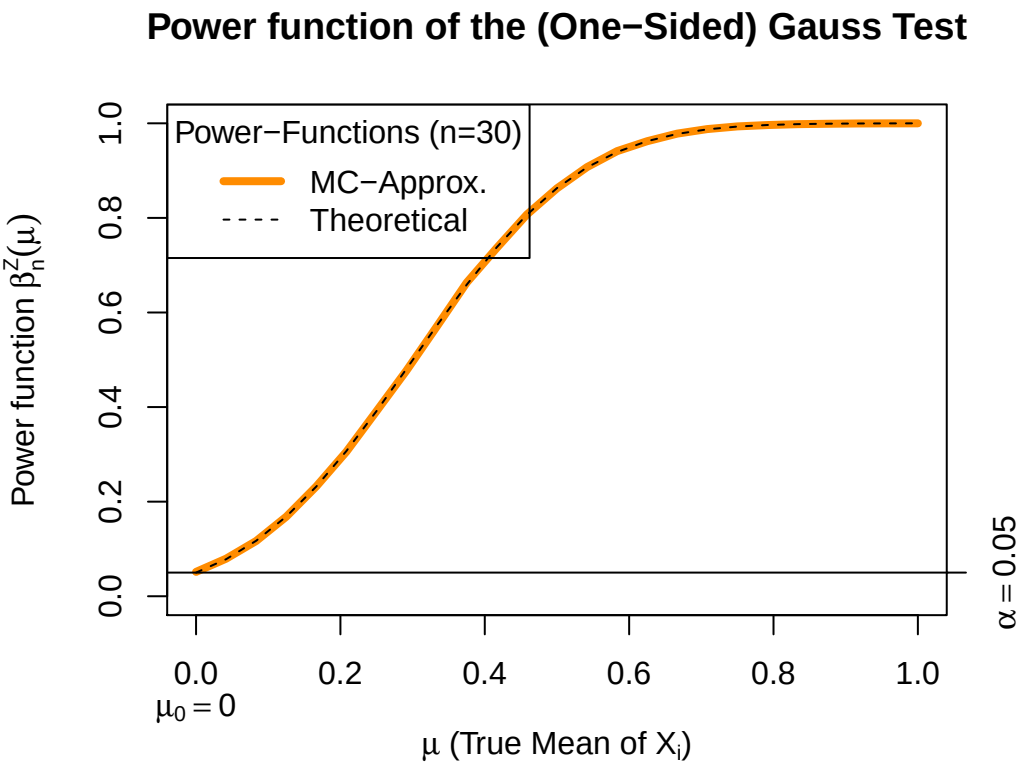
Plot:

The function `Gauss.MC.beta()` allows us now to compare the theoretical trajectory of the power function $\beta^Z_{n,0.05}(\mu_X)$ for $\mu_X \in \Omega_0 \cup \Omega_1$, e.g., for the sample size $n = 30$ with its simulated counterpart.

Here is the R-Code to do this:

```
## Sequence of different mu_X values (here from 0 to 1):
mu.X.true.seq   <- seq(0,1,len=25)

## Simulating the trajectory of the power function for n=30:
## Number of MC-Repetitions:
m <- 50000
## Try also:
## m <- 100

beta.MC.n.30       <- Gauss.MC.beta(n         = 30,
                                    mu.X.true = mu.X.true.seq,
                                    m         = m)


## Plot
par(mar=c(5.1,4.1+1,4.1,2.1))
plot(y=0, x=0, type="n",
     ylim=c(0,1),
     xlim=range(mu.X.true.seq),
     xlab=expression(paste(mu," (True Mean of ", X[i],")")),
     ## Labels:
     ylab=expression(paste("Power function ",beta[n]^Z,(mu))),
     main="Power function of the (One-Sided) Gauss Test")
## Null-hypothesis mean:
mtext(text = expression(mu[0]==0), side = 1, line = 2, at = 0)
## Trajectories:
lines(y=beta.MC.n.30, x=mu.X.true.seq, lty=1, lwd=4, col="darkorange")
lines(y=beta.n.30,    x=mu.X.true.seq, lty=2)
## Significance Level:
axis(4, at=0.05, labels = expression(alpha==0.05))
abline(h=0.05, lty=1)
## Legend:
legend("topleft", title = "Power-Functions (n=30)",
```

```
       legend = c("MC-Approx.","Theoretical"),
       lty=c(1:2), lwd=c(4,1), col=c("darkorange","black"))
```



5.2 Checking Parameter Estimators

For the following, we use our `myOLSFun()` function to compute OLS estimators.

```
myOLSFun <- function(y, x, add.intercept=FALSE){

  ## Number of Observations:
  n         <- length(y)

  ## Add an intercept to x:
  if(add.intercept){
    Intercept <- rep(1, n)
    x         <- cbind(Intercept, x)
  }

  ## Estimation of the slope-parameters:
  beta.hat.vec <- solve(t(x) %*% x) %*% t(x) %*% y

  ## Return the result:
  return(beta.hat.vec)
}
```

Let us consider the multiple regression model:

$$y_i = \beta_1 + \beta_2 x_{2i} + \beta_3 x_{3i} + \varepsilon_i, \quad i = 1, \ldots, n,$$

where ε_i is a heteroscedastic error term

$$\varepsilon_i \sim N(0, \sigma_i^2), \quad \sigma_i = x_{3i},$$

and where for all $i = 1, \ldots, n = 50$:

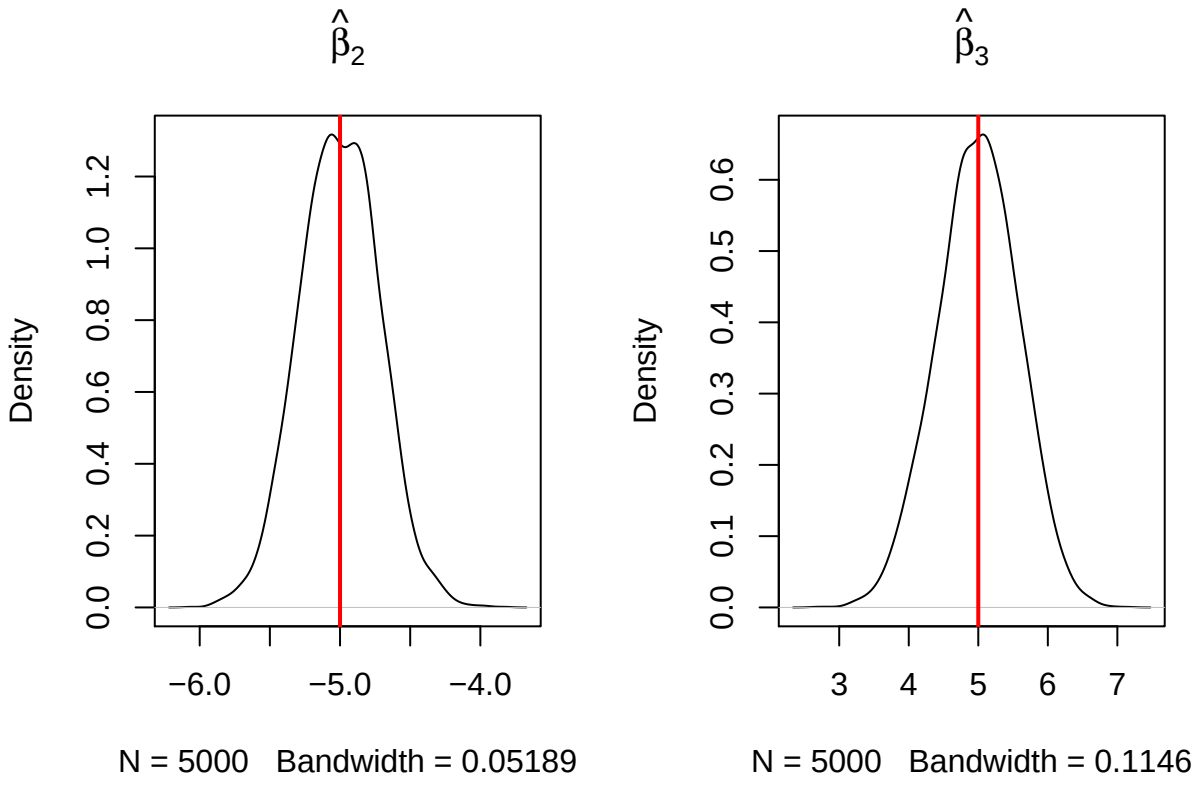
- $\beta_1 = 1$, $\beta_2 = -5$, and $\beta_3 = 5$
- $x_{2i} \sim N(10, 1.5^2)$
- x_{3i} comes from a t-distribution with 5 degrees of freedom and non-centrality parameter 2

This following code generates:

1. `m <- 5000` (pseudo) random samples from the above model.
2. Computes the OLS estimates $\hat{\beta}_{2j}$ and $\hat{\beta}_{3j}$ for each sample $j = 1, \ldots, m$
3. Stores the estimation results in the data-vectors `beta.2.sim` and `beta.3.sim`
4. Plots the distribution of the estimation results using non-parametric density plots

```
## Simulation parameters:
set.seed(109)            # Sets the "seed" of the random number generators
m           <- 5000      # Number of simulation runs
## Model parameters:
beta.vec    <- c(1,-5,5) # Slope coefficients
n           <- 50        # Number of observations
## Containers to save simulation results:
beta.2.sim  <- rep(NA,m)
beta.3.sim  <- rep(NA,m)

## Generate the regressors:
## Outside of the loop (i.e., 'conditional on X')
X.1 <- rep(1, n)
X.2 <- rnorm(n, mean=10, sd=1.5)    # Draw realizations form a normal distr.
X.3 <- rt(n, df=5, ncp=2)           # Draw realizations form a t-distr.
X   <- cbind(X.1, X.2, X.3)         # Save as a Nx3-dimensional matrix.

## Setup a progressbar
#pb <- txtProgressBar(min = 0, max = m, style = 3)

for(rpt in 1:m){
  eps <- (X.3)*rnorm(n, mean=0, sd=1) # heteroscadastic error term
  y   <- X %*% beta.vec + eps         # Dependent variable
  ## Estimation
  beta.hat <- myOLSFun(y=y,x=X)
  ## Save results
  beta.2.sim[rpt] <- beta.hat[2]
  beta.3.sim[rpt] <- beta.hat[3]
  ## Progress bar
  #setTxtProgressBar(pb, rpt)
}
#close(pb)# Close progressbar

## Plot results
par(mfrow=c(1,2))
plot(density(beta.2.sim, bw="SJ"), main=expression(hat(beta)[2]))
abline(v=beta.vec[2], col="red", lwd=2)
##
```

```
plot(density(beta.3.sim, bw="SJ"), main=expression(hat(beta)[3]))
abline(v=beta.vec[3], col="red", lwd=2)
```



Bibliography

Baltagi, B. (2008). *Econometric Analysis of Panel Data*. John Wiley & Sons.

F. Bretz, T. Hothorn, P. W. (2010). *Multiple Comparisons Using R*. Chapman and Hall/CRC.

Fan, J. and Gijbels, I. (1996). *Local Polynomial Modelling and its Applications*, volume 66 of *Monographs on Statistics and Applied Probability*. Chapman & Hall/CRC, 1. edition.

Gałecki, A. and Burzykowski, T. (2013). *Linear Mixed-Effects Models Using R: A Step-by-Step Approach*. Springer.

Gelman, A. and Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.

Greene, W. (2003). *Econometric Analysis*. Pearson.

Hastie, T., Tibshirani, R., and Wainwright, M. (2015). *Statistical Learning with Sparsity: The Lasso and Generalizations*. CRC press.

Hsiao, C. (2014). *Analysis of Panel Data*. Cambridge university press.

Li, Q. and Racine, J. (2007). *Nonparametric Econometrics: Theory and Practice*. Princeton University Press.

Romano, J. and Wolf, M. (2005). Exact and approximate stepdown methods for multiple hypothesis testing. *Journal of the American Statistical Association*, 100(469):94–108.

Verbeke, G. and Molenberghs, G. (2000). *Linear Mixed Models for Longitudinal Data*. Springer.

Wand, M. and Jones, M. (1994). *Kernel Smoothing*, volume 60. Chapman & Hall/CRC.

White, H. (2014). *Asymptotic Theory for Cconometricians*. Academic press.

Y. Hochberg, A. T. (1987). *Multiple Comparison Procedures*. Wiley Series in Probability and Statistics.