Masterclass Advanced Trial-based Economic Evaluations in R

Ben Wijnen, Ph.D.
Trimbos Institute
Centre for Economic Evaluation and Machine Learning
bwijnen@trimbos.nl

Trimbos instituut
Netherlands Institute of Mental Health and Addiction

Centre for Economic Evaluation and Machine Learning

prof. dr. Filip Smit

drs. Anne Kleijburg

dr. Joran Lokkerbol

dr. Ben Wijnen

prof. dr. Silvia Evers

Aim of today

- See how an economic evaluation can be done using R
- Learn from the example code
- Understand what happens at each of the steps

Be aware:

- The level of experience with R varies a lot within this group
- Exercises are designed in such a way that they don’t require coding yourself
- It’s not the goal to be able to code everything yourself from scratch afterwards

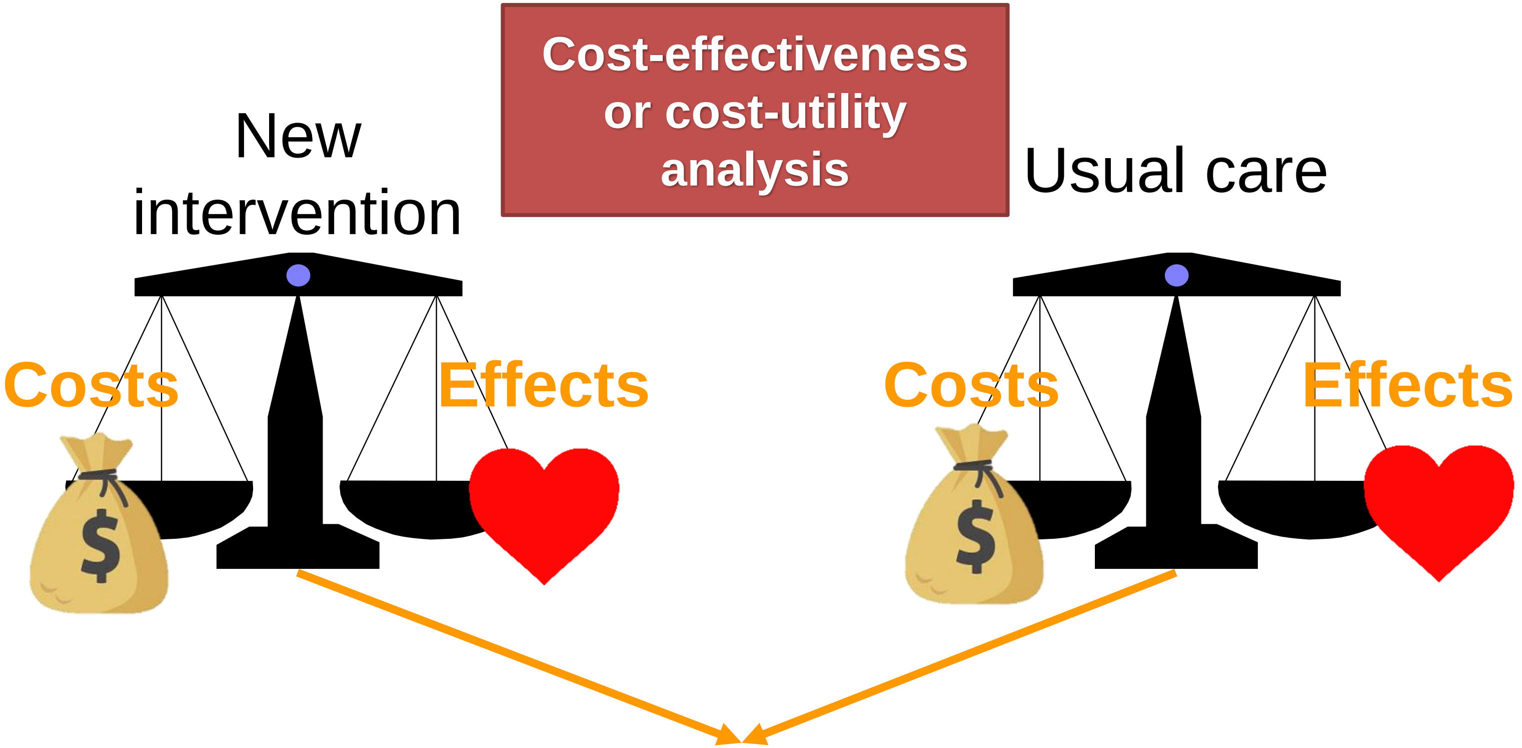
Disclaimer

- I work with R daily, but I feel I am in no means an expert user
- My code does the trick (hopefully) and is designed to be intuitive
  - That being said: things can be coded quite a bit more efficient (e.g., parallel processing, ggplot codes)
- I'm not a statistician (but process developed in close collaboration with one)
- Results will be (roughly) similar due to the use of seeds (but can still vary a little in the 3rd exercise due to variation in package versions)
- Most importantly: I learn everyday, so tips/tricks are always welcome

Content

9:00 – 9:15	Introduction
9:15 – 9:45	Exercise 1 - First (simple) bootstrap
9:45 – 10:00	Short intro to exercise 2
10:00 – 10:15	Break
10:15 – 10:50	Exercise 2 – Baseline correction
10:50 – 11:00	Short intro to exercise 3
11:00 – 11:15	Break
11:15 – 11:45	Exercise 3 – Handling missing data
11:45 – 12:00	Final remarks/questions

Recap economic evaluation



Incremental cost-effectiveness ratio:

$$(C_n - C_c)\ /\ (E_n - E_c)$$

= Costs per 

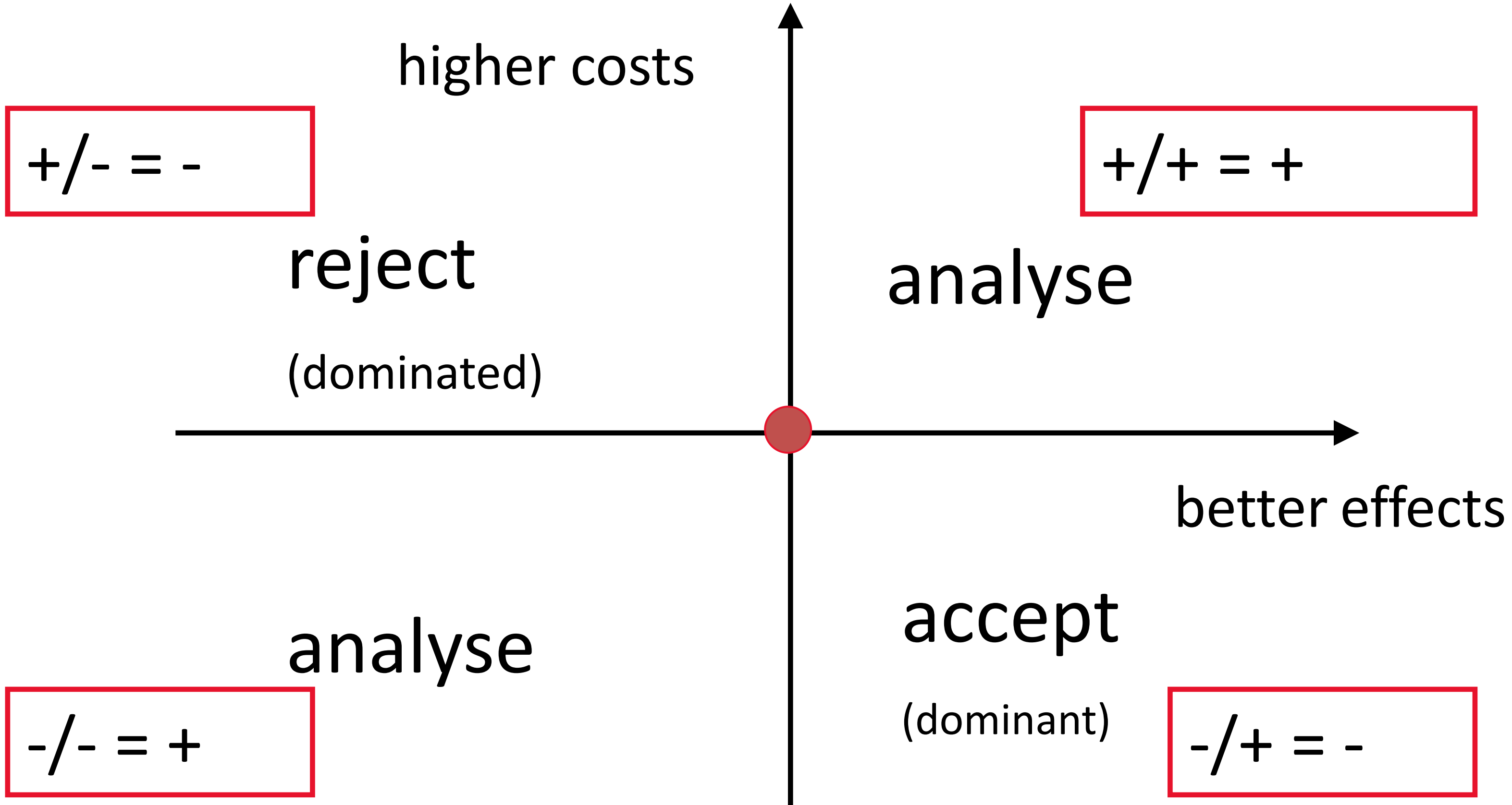
Calculating the ICER

	Intervention group		Control group	
Observation	Cost (€)	QALYs	Cost (€)	QALYs
1	215	0.630	415	0.310
2	500	0.456	350	0.425
3	460	0.870	630	0.370
4	150	0.458	200	0.250
5	600	0.650	315	0.425
6	750	0.560	260	0.510
7	873	0.750	325	0.310
8	357	0.358	415	0.335
9	413	0.560	325	0.410
10	475	0.370	275	0.415
Group mean	*479*	*0.566*	*351*	*0.376*

ICER = (479 – 351) / (0.566 – 0.376)
= €674 per QALY gained

Trimbos instituut

Cost-effectiveness plane



$$\text{Incremental CE ratio} = \frac{\Delta\ \text{costs}}{\Delta\ \text{effects}}$$

Bootstrapping

0

Trial data: **Ketogenic diet for children with epilepsy**

Control

Number of patients	**10**
Average effectiveness	**0.76**
Average cost	**522.00**

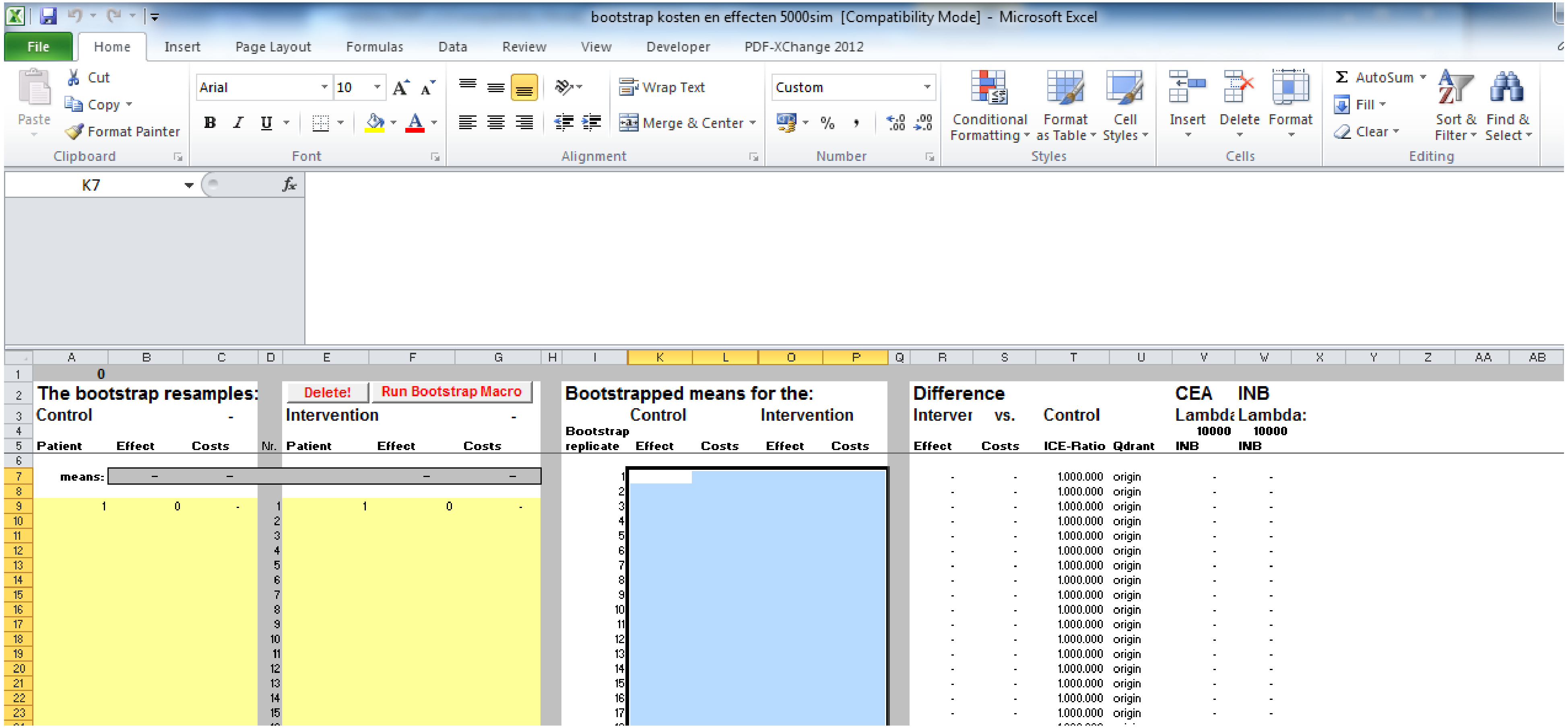
Patientnumbe	Effect	Costs
1	0.7065	550
2	0.8205	400
3	0.802	600
4	0.707	380
5	0.7205	590
6	0.678	550
7	0.802	700
8	0.761	650
9	0.8405	300
10	0.758	500
11		
12		
13		
14		
15		
16		
17		
18		

Intervention

Number of patients	**10**
Average effectiveness	**0.784750**
Average cost	**616**

Patientnumbe	Effect	Costs
1	0.8555	440
2	0.7415	520
3	0.721	660
4	0.8125	540
5	0.707	420
6	0.7685	540
7	0.796	620
8	0.79	680
9	0.816	1020
10	0.8395	720
11		
12		
13		
14		
15		
16		
17		
18		

Bootstrapping



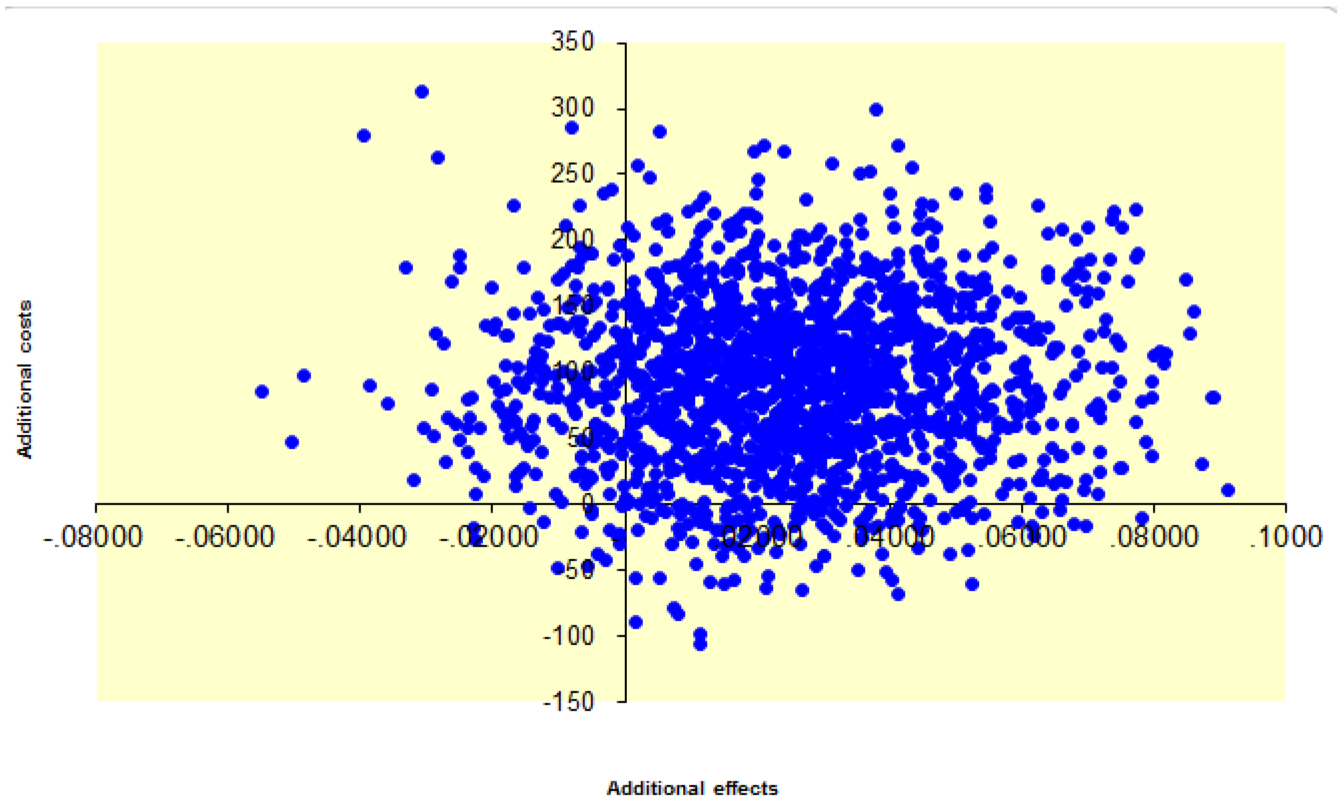
Bootstrapping

Original Data		
	Cost	Effect
1	1000	1
2	1500	1
3	12000	0
4	900	0
5	1000	1
A	2000	1
B	1500	0
C	14000	1
D	5000	1
E	800	1
	6900/0.2	

Bootstrap 1		
	Cost	Effect
5	1000	1
3	12000	0
3	12000	0
2	1500	1
1	1000	1
A	2000	1
A	2000	1
E	800	1
E	800	1
D	5000	1
	-16900/0.4	Dom.

Bootstrap X		
	Cost	Effect
2	1500	1
2	1500	1
4	900	0
5	1000	1
5	1000	1
C	14000	1
D	5000	1
A	2000	1
D	5000	1
E	800	1
	20900/0.2	

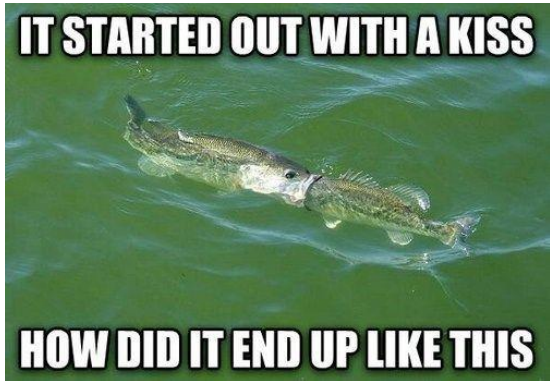
Bootstrapping



Advantages of R

- No need to switch between programs (e.g., SPSS and Excel)
- Run multiple (sensitivity) analyses simultaneously

- Combine baseline adjustment with imputations in one step
- **But: STATA (or Python) can also be used to do all the things we are doing Today**
- Watchout:



Trimbos instituut

First exercise

Simple
bootstrapping in R

Intro exercise 1

In total, there are three exercises today. They all consist of similar files:

- HTML-file with code & additional explanation
- .R file wich you can use to build the syntax yourself

Use copy/paste! It is more important to know what is happening, than to code everything yourself

Intro exercise 1

Assignment_1_replicate.R

Source on Save | Run | Sour

```
# Non-parametric bootstrapping of costs and effects

# dr. Ben Wijnen
# Netherlands Institute of Mental Health and Addiction, Center for Economic Evaluations (Trimbos Institute)
# bwijnen@trimbos.nl

# SETUP
rm(list = ls())                # Clear the R environment
options(scipen=999)            # Turn off scientific notation in Console
options(max.print=1000000)     # Turn off scientific notation in Console
set.seed(1234)            # Set seed to keep random results the same everytime you run the analysis

# INSTALL PACKAGES (this only need to be done once)
install.packages("openxlsx")
install.packages("BCEA")
install.packages("summarytools")
install.packages("eq5d")

## Load packages
library(summarytools) # Package to easily create summary statistics of your dataset
library(openxlsx)     # To open Excel files
library(eq5d)         # Package to easily calculate utility values (not used in this example)
library(BCEA)         # Package to easily create plots based on EE

## Set working directory
setwd("C:/Users/wijnbe/OneDrive - Trimbos-Instituut/Documents/Trimbos/Cursus_EE") ##                                   PUTERS

# Load dataset
dataset <- read.xlsx("Data_trial_based_EE.xlsx", sheet=1) ## PLEASE MAKE SURE YOU DC
```

This is standard setup code

Run this code once (you can delete this afterwards)

Make sure all packages are loaded correctly

Set the WD correctly and make sure the dataset is in the same folder as the WD

Trimbos instituut

Intro exercise 1

The following steps are taken:

- Step 1: Inspect data (most important: distributions, missing values, outliers)
- Step 2: Data cleaning / feature engineering
- Step 3: Calculate QALYs / costs
- Step 4: Look at baseline imbalances
- Step 5: Look at missing values
- Step 6: Perform analyses
- Step 7: Perform sensitivity analyses
- Step 8: Inspect results
- Step 9: Visualize results

Step 1: Inspect data

- Try to use the package *Summarytools* (provides easy overview of both contineous and categorial data)

- Alternatively, the Desctools package could be used to obtain quick summary statistics (e.g. using the *describeFast* function)

Step 3: Calculate QALYs / costs

For EQ-5D:

- Very useful R-package (eq5d) which makes it easy to calculate index scores (utilities) for either the three-level (EQ-5D-3L) or five-level (EQ-5D-5L) version

Step 6: Perform analyses

The analysis step consists of three steps:

- Define the data.frame in which you want to store the results
- Create a loop in which the data is bootstrapped
- Within each bootstrap: calculate incremental costs and effects

Step 6: Perform analyses

```r
BS <- 2500                                            # set the nr of bootstrap replications
ints <- c("COVID vaccination","No vaccination") # Set names of treatment arms

bootstraps <- data.frame(sim_nr = 0, incr_costs = 0, incr_QALYs=0, incr_responder=0, QALYs_control=0, QALYs_interven=0, response_control=0,
                         response_interven=0, costs_control=0, costs_interven=0) # create dataframe for bootstrap results
```

Results in:

	sim_nr	incr_costs	incr_QALYs	incr_responder	QALYs_control	QALYs_interven	response_control	response_interven	costs_control	costs_interven
1	0	0	0	0	0	0	0	0	0	0

Step 6: Perform analyses

Bootstrap loop:

```
for(i in 1:BS){
  t <- sample(c(1:as.numeric(nrow(dataset))),as.numeric(nrow(dataset)),replace=T) # Sample rows random with replacement
  datasetx <- dataset[t,]                                                          # Create a new dataset based on the resampled rows

  QALYs_control<- mean(datasetx[datasetx$Trial_arm==0, "QALY"])                    # Calculate mean QALYs for control condition
  QALYs_interven<- mean(datasetx[datasetx$Trial_arm==1, "QALY"])                   # Calculate mean QALYs for intervention condition

  response_control <- mean(datasetx[datasetx$Trial_arm==0, "Responder"])           # Calculate mean response rate for control condition
  response_interven <- mean(datasetx[datasetx$Trial_arm==1, "Responder"])          # Calculate mean response rate for intervention condition

  costs_control<- mean(datasetx[datasetx$Trial_arm==0, "Total.cost"])              # Calculate mean costs for control condition
  costs_interven<- mean(datasetx[datasetx$Trial_arm==1, "Total.cost"])             # Calculate mean costs for intervention condition

  incr_costs <- costs_interven - costs_control                                     # Calculate incremental costs
  incr_QALYs <- QALYs_interven -  QALYs_control                                    # Calculate incremental QALYs
  incr_responder <- response_interven - response_control                           # Calculate incremental response rate

  scores <- c(sim_nr = paste(i), incr_costs, incr_QALYs, incr_responder, QALYs_control, QALYs_interven, response_control,
            response_interven, costs_control, costs_interven)                      # Add all created vars to a dataframe
  bootstraps[i,] <- scores                                                         # Add the dataframe to each "row" of the bootstrap data.frame (each "i")
} # this ends the bootstrap loop
```

Trimbos instituut

Step 8: Inspect results

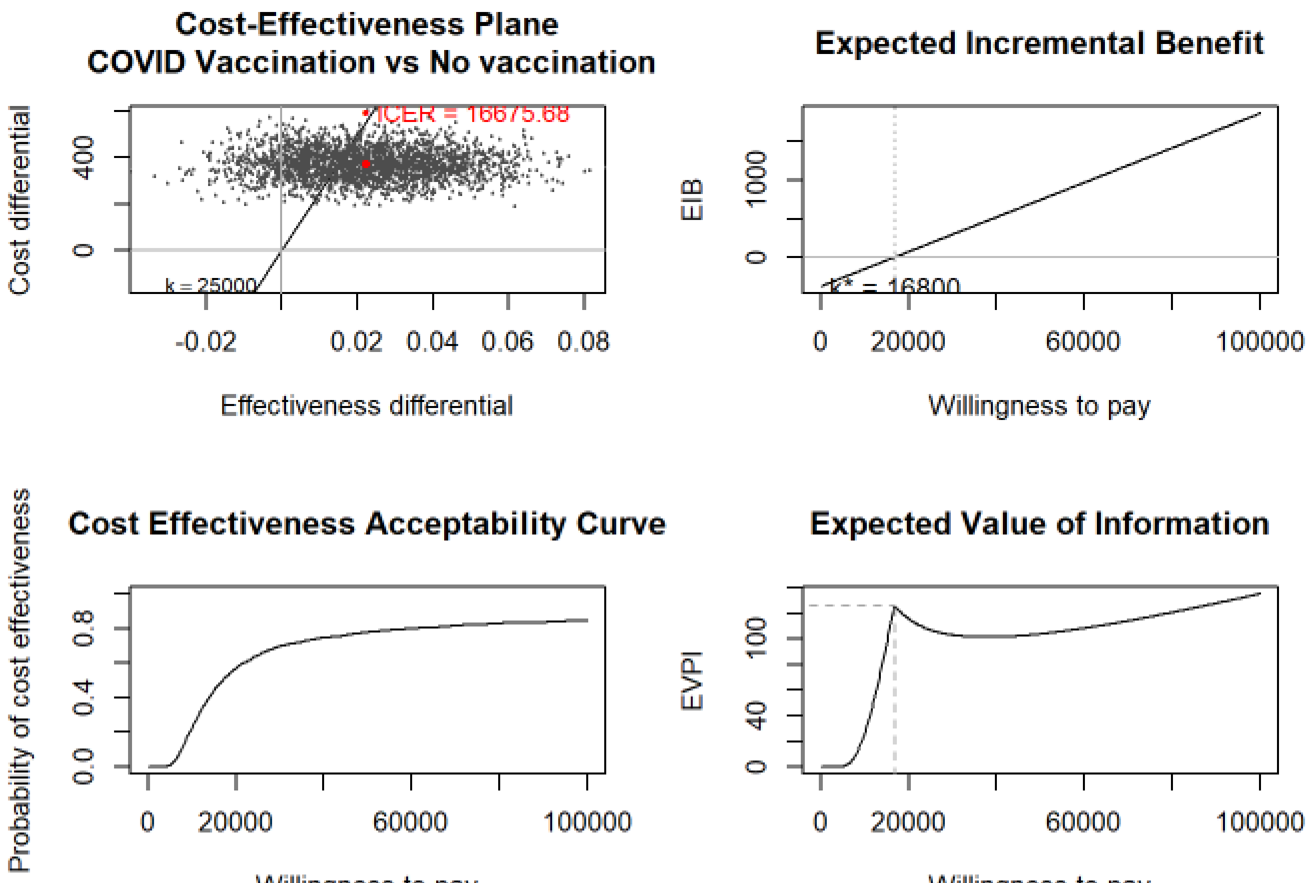
- All bootstrap results are now stored in the data.frame “bootstraps”
- You can use a simple “summary” command to get means and distributions of each variable

Trimbos instituut

Step 9: Visualize results

- In this example, instead of create our own graphs with packages such as ggplot2, we use the ‘BCEA’ package to easily create the CE-plane and CEAC
- Downside: results need to be saved in a matrix, so it requires an additional step
- Advantage: Once in a matrix, graphs are very easy to create (but hard to customize yourself)

Step 9: Visualize results



Start with the first exercise

Trimbos instituut
Netherlands Institute of
Mental Health and Addiction

Intro exercise 2

The following steps are taken:

- Step 1: Inspect data (most important: distributions, missing values, outliers)
- Step 2: Data cleaning / feature engineering
- Step 3: Calculate QALYs / costs
- **Step 4: Look at baseline imbalances**
- Step 5: Look at missing values
- Step 6: Perform analyses
- Step 7: Perform sensitivity analyses
- Step 8: Inspect results
- Step 9: Visualize results

Trimbos instituut

Intro exercise 2

- The data is synthesized, but roughly based on a previously published trial-based economic evaluation looking at Complaint-Directed Mini-Interventions for Depressive Symptoms

- We're using complete cases only

- We are going to perform a baseline adjustment (we're not going into detail whether this should be done in this case or not)

- Structure is roughly similar to part I (i.e. we will create a bootstrap loop in which we are now going to use regression functions to adjust baseline differences)

Baseline adjustment

- Previously, we’ve looked at the mean incremental costs and effect in each bootstrap simulation
- Now: To simultaneously evaluate both costs and outcomes, seemingly unrelated regression equations (SURE) models will be used
  - Just two regression equations run simultaneously
- When adding no covariates other than the “group” variable, this regression will still give the mean incremental differences
  - However, when adding covariates, the incremental effect will be accounted for baseline difference
- In line with e.g. Van Asselt, Antoinette DI, et al. "How to deal with cost differences at baseline." Pharmacoeconomics 27.6 (2009): 519-528.

Baseline adjustment

```r
## Initiate the SURE regression model
sureg <- function(dataset_comp){
  r1 <- QALYs~Condition
  r2 <- Total_costs_12m~Condition
  fitsur <- systemfit(list(eff = r1, costs = r2), data=dataset_comp)
  return(c(fitsur$coefficients[4],fitsur$coefficients[2])) # extract coefficient for difference in costs and QALYs
}
```

Custom function (that uses the "*systemfit* function to perform SURE) that gives:

- R1: Regression function for QALYs (note: not baseline corrected yet)
- R2: Regression function for costs (note: not baseline corrected yet)

- Returns: the coefficient for "condition" (i.e. which represents the incremental differences for the two groups)

- Next, this function is used in the bootstrap loop for each replication

Trimbos instituut

Baseline adjustment

- For the dataset in exercise 2:

  - Costs:

```
> mean(dataset_comp[dataset_comp$Condition==0, "Total_costs_12m"]) ## Total costs control group
[1] 3506.838
> mean(dataset_comp[dataset_comp$Condition==1, "Total_costs_12m"]) ## Total costs intervention group
[1] 2722.506
```

  - QALYs

```
> mean(dataset_comp[dataset_comp$Condition==0, "QALYs"]) ## Total QALYs control group
[1] 0.5226339
> mean(dataset_comp[dataset_comp$Condition==1, "QALYs"]) ## Total QALYs intervention group
[1] 0.6030966
```

  - SURE model:

```
r1 <- QALYs~Condition
r2 <- Total_costs_12m~Condition
fitsur <- systemfit(list(eff = r1, cost
```

Results SURE:

```
systemfit results
method: OLS

Coefficients:
  eff_(Intercept)  eff_Condition costs_(Intercept)  costs_Condition
        0.5226339      0.0804627      3506.8378571     -784.3314935
```

Coefficient #2

Coefficient #4

Trimbos instituut

Baseline adjustment

```r
# Add variables you wish to correct for to SURE regression model
sureg_corr <- function(dataset_comp){
  r1 <- QALYs~Condition+Utility_bs ## NOTE: SURE MODELS DO NOT NEED TO BE THE SAME
  r2 <- Total_costs_12m~Condition  ## NOTE: SURE MODELS DO NOT NEED TO BE THE SAME
  fitsur <- systemfit(list(eff = r1, costs = r2), data=dataset_comp)
  return(c(fitsur$coefficients[5],fitsur$coefficients[2])) # extract coefficient for difference in costs and QALYs;
                                                            # MAKE SURE YOU EXTRACT THE CORRECT COEFFICIENTS (E.G. IN THIS CASE 5 & 2)
}
```

Easy extension to baseline adjustments:

- R1: Regression function for QALYs (note: this is **now corrected for baseline differences**)
- R2: Regression function for costs (note: not corrected)

- Make sure you are returning the correct coefficients from the “systemfit” function

Step 9: Visualize results

- In this exercise, results are now visualized using “ggplot2”
- Bit more cumbersome but give the freedom do adapt the graphs more easy
- Note: provided code can be made a bit more efficient

Start with the second exercise

Trimbos instituut
Netherlands Institute of Mental Health and Addiction

Intro exercise 3

The following steps are taken:

- Step 1: Inspect data (most important: distributions, missing values, outliers)
- Step 2: Data cleaning / feature engineering
- Step 3: Calculate QALYs / costs
- Step 4: Look at baseline imbalances
- **Step 5: Look at missing values / imputations**
- Step 6: Perform analyses
- Step 7: Perform sensitivity analyses
- Step 8: Inspect results
- Step 9: Visualize results

Look at missing values / imputations

- In this masterclass, we assume that data is missing completely at random (MCAR)

- In the past, it was challeging to combine bootstraps with (multiple) imputations
  - E.g. could be done but to a limited extend
  - Even more difficult when combing this with baseline adjustments

Trimbos instituut

Look at missing values / imputations

- Brand et al. 2019 looked at statistical efficiency of 10 candidate methods and applied these methods to a clinical data set
  - Brand, Jaap, et al. "Combining multiple imputation and bootstrap in the analysis of cost-effectiveness trial data." Statistics in Medicine 38.2 (2019): 210-220.
- Single imputation nested in the bootstrap percentile method emerged as the method with the best statistical properties
  - can require extensive computation time

Look at missing values / imputations

How to identify predictor variables imputation algorithm:

1. Identify predictors based on which variables are significantly associated with the clinical outcome
2. Variables which are significantly associated with missing values using a logistic regression with “missing yes/no” as binary outcome

- You can impute each timepoint but in the exercise we only have total cost
- I normally impute subtotals instead of individual cost items

Look at missing values / imputations

Next: create a prediction matrix in R

```
##                 Leeftijd geslacht Opleidingdi Condition Total_costs_12m QALYs
## Leeftijd               0        0           0         0               0     0
## geslacht               0        0           0         0               0     0
## Opleidingdi            0        0           0         0               0     0
## Condition              0        0           0         0               0     0
## Total_costs_12m        1        0           1         1               0     0
## QALYs                  1        1           1         1               0     0
```

We use predictive mean matching (PMM) to account for non-normality of the data, by imputing 'real' observed values from similar cases instead of imputing regression estimates

Look at missing values / imputations

The is now a double loop in the code:

1. The bootstrap replication → this command now replicates the dataset **with missings**
2. Within each bootstrap replication, we now have a **single imputation**

This results in 2,500 imputed bootstrap replications which can be used to construct CE-planes and CEACs etc.

Start with the third exercise

Trimbos instituut
Netherlands Institute of Mental Health and Addiction