Guidelines for revenue cap calculation in R

The Norwegian Water Resources and Energy Directorate

Contents

1 Introduction

The Norwegian Water Resources and Energy Directorate (NVE) regulates the network companies by using an incentive-based revenue cap (RC) model. We annually award each network company a RC, which is largely based on the company's own cost efficiency. The RCs are then included in each company's allowed revenue, which constitutes the basis for tariff calculations.

NVE calculates the RCs in the programming language, R. To ensure transparency in the economic regulation of network companies, we make the script publicly available. The script does all the calculations that are necessary to calculate the RCs. However, it may not be apparent to everyone what is actually happening in the script. Therefore, we provide you with these guidelines. Here we will give a systematic explanation of the content of the script.

In this first chapter, we will briefly introduce the RC regulation[1], and in chapter 2 we explain how to get started in R. We move on to actually describing the script in chapter 3. Here we focus on how the initial data set is imported and altered. Finally, chapter 4 explains the part of the script that does the actual RC calculation.

1.1 Fundamental principles of the economic regulation

The companies' RCs shall cover operating costs and depreciations, and give a reasonable return on invested capital over time, given efficient operation, utilization and development of the network grid. By comparing the companies' input and outputs, the RC calculation in R ensures the precondition of "given efficient …" above. The result is that companies that are more efficient gain a higher return on their invested capital than less efficient companies do.

1.2 Revenue Cap model

The cost base for the RCs consists of operation and maintenance costs, costs of energy not supplied (CENS), cost of network losses, depreciation, and return on invested capital. There is a two-year lag in the cost data, which means that the RC in year t is based on cost information from year t-2.

The annual RCs are based on a formula of 40 percent cost base and 60 percent cost norm. This means that the companies receive cost coverage for 40 percent of their actual costs. The cost norm, however, results from a benchmarking model based on data envelopment analysis (DEA). Here, we compare the companies to each other in order to award them with an efficiency score. The efficiency scores are later adjusted for differences in operating environments, as the companies may experience different geographical challenges.

Companies with a high efficiency score will receive a larger RC relative to their costs than companies with lower efficiency scores. This largely captures the essence of the economic regulation as the network companies are given incentives to operate, utilize and develop the network grid efficiently.

1.3 Practical information

The script consists of one main script and several underlying scripts, where the main script calls for the other scripts to run. The folder containing all folders with relevant scripts and data is called IRiR, possibly with some postfix if the file is downloaded from GitHub. The main script is called NVE_IRiR. A brief description of the file and folder organization is provided in the README-file.

[1] You can find more information about the revenue regulation on our web page:
https://www.nve.no/energy-market-and-regulation/revenue-regulation/ (ENG)
https://www.nve.no/elmarkedstilsynet-marked-og-monopol/okonomisk-regulering-av-nettselskap/reguleringsmodellen/ (NOR)

If the goal is to calculate RCs, it is sufficient to run the main script. However, if you want to make any changes in the calculation, or understand exactly what the calculations does, we need to look into the underlying scripts. In these guidelines, we will focus on what is happening in the underlying scripts.

`We show elements from the script, which includes variable names, commands, etc., in this font.`

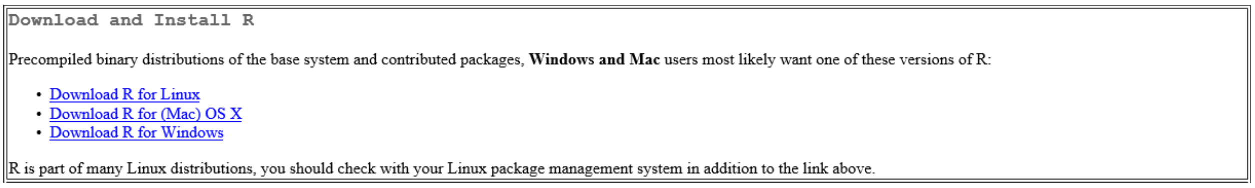
Throughout the script, there are some references to web pages, reports, etc. The ones written in Norwegian are labeled `NOR`, and the ones written in English are labeled `ENG`.

In the following chapters, you will see that we often leave out the first part of the variable name, which usually defines which grid level the variable applies to (local distribution = `ld_`, regional distribution = `rd_` and transmission = `t_`). This leaves us with `_variable.name`, which saves us for unnecessary repetitions.

2 Getting started in R

2.1 Installing R & RStudio

R is a free software for statistical computing and graphics[2]. Please download the R version suitable for your operating system (OS) from https://cran.r-project.org/.



Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

It is recommended to use the Precompiled binary distributions of R (if you are interested in the Source Code for R, you will not need to read the «Getting started» part of this guide). On a windows installation please download and run the .exe-file from this[3] link. Follow the installation instructions. If you are installing the software on a corporate computer, you might need permissions and aid from your IT-department to do so.

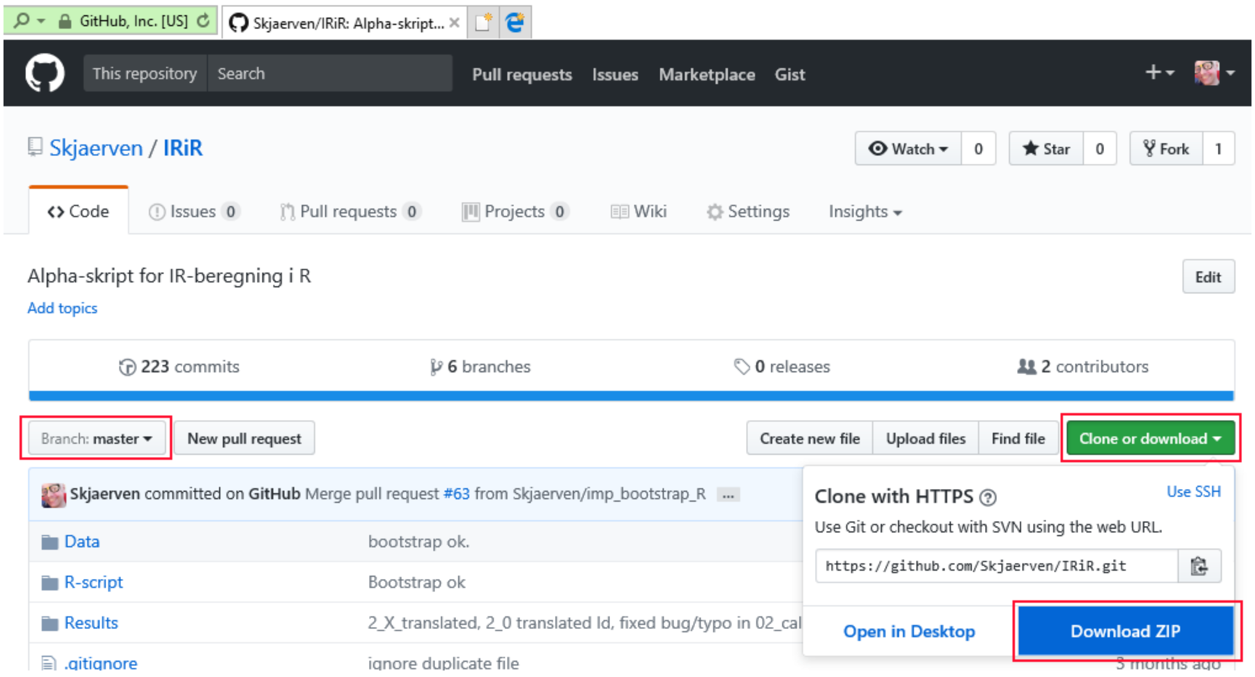
Please note that the «IRiR»-script is developed using the Windows R platform i386-w64-mingw32/i386 (32-bit). It has not been tested on neither Linux nor OS X devices.

To get a similar graphical user interface (GUI) as depicted in this guide, please download and install RStudio. RStudio is also recommended if you have limited experience working with computer scripts in R. The latest versions of RStudio should be available here[4]. As for the R installation, run the exe-file and follow installation instructions. Ask your IT-department for assistance if you are installing the software on a corporate computer.

2.2 Retrieve the latest version of the IRiR script

NVE uses GitHub for version control and as a hub for storage of the scripts. The latest version of the script is available for download at https://github.com/NVE/IRiR. The following screenshots show how to download the correct scripts to your computer.

[2] https://www.r-project.org/
[3] https://cran.r-project.org/bin/windows/base/release.htm
[4] https://www.rstudio.com/products/rstudio/download/



Alpha-skript for IR-beregning i R

Skjaerven committed on GitHub Merge pull request #63 from Skjaerven/imp_bootstrap_R

Data	bootstrap ok.
R-script	Bootstrap ok
Results	2_X_translated, 2_0 translated Id, fixed bug/typo in 02_cal
.gitignore	ignore duplicate file

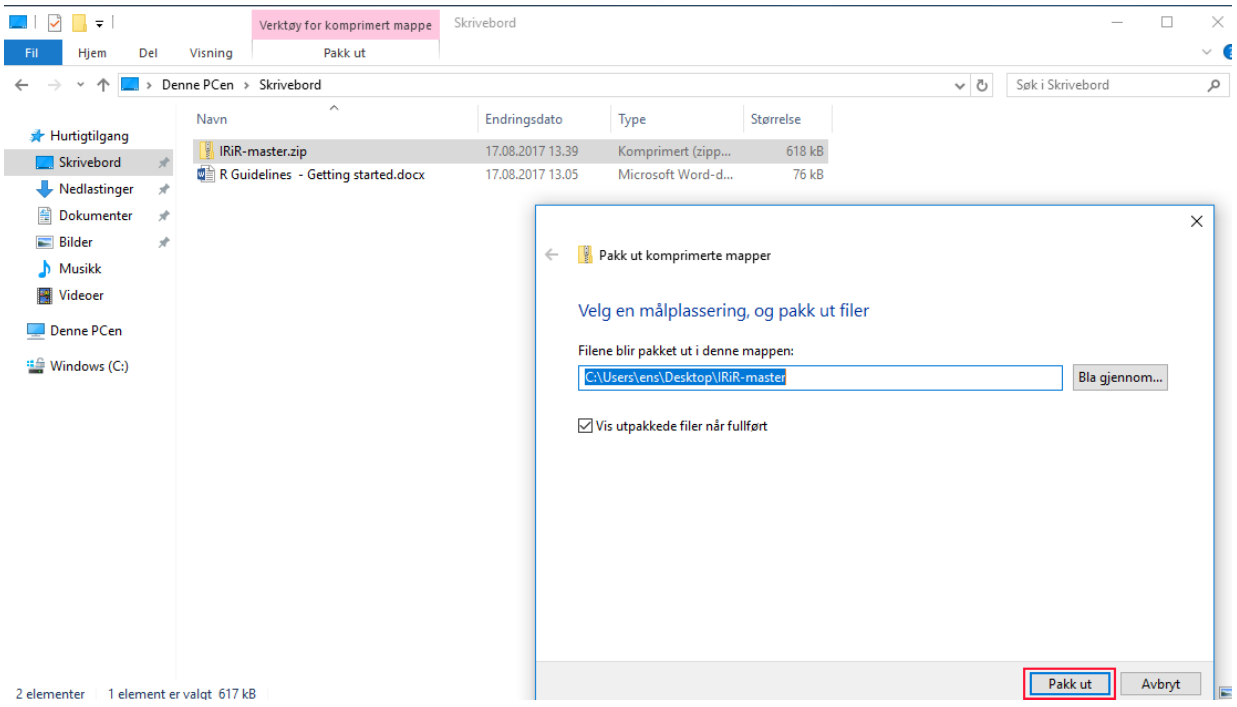
Use Git or checkout with SVN using the web URL.

https://github.com/Skjaerven/IRiR.git

Make sure that you have chosen branch "master", which is the default branch. Then choose "Clone or download", and "Download ZIP".

If you are using the GitHub desktop application, you can also use the "Open in Desktop" option. However, we have not included a description of this here.

Next, save the ZIP-folder in a desired location on your computer, and extract the files from the zipped folder.



Navn	Endringsdato	Type	Størrelse
IRiR-master.zip	17.08.2017 13.39	Komprimert (zipp...	618 kB
R Guidelines - Getting started.docx	17.08.2017 13.05	Microsoft Word-d...	76 kB

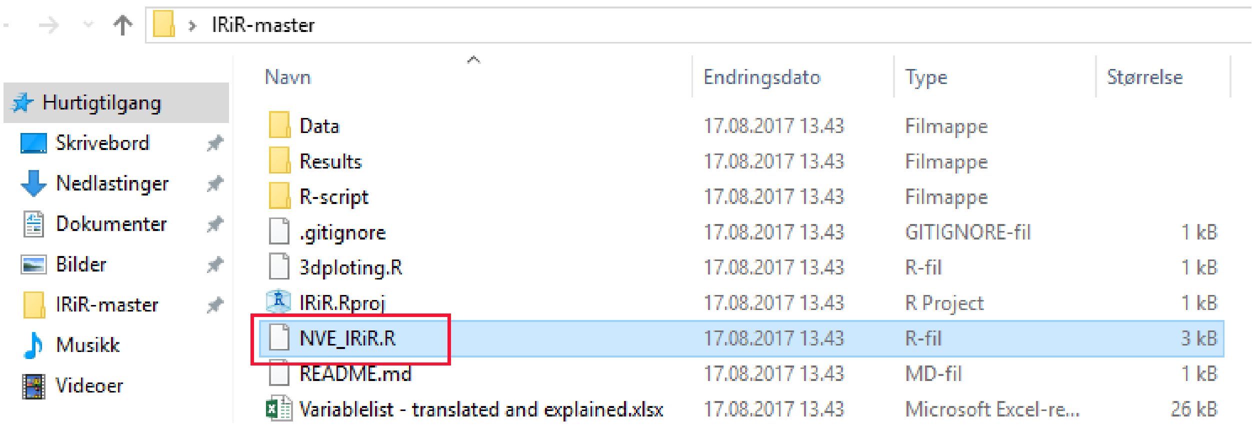
Velg en målplassering, og pakk ut filer

Filene blir pakket ut i denne mappen:

C:\Users\ens\Desktop\IRiR-master

☑ Vis utpakkede filer når fullført

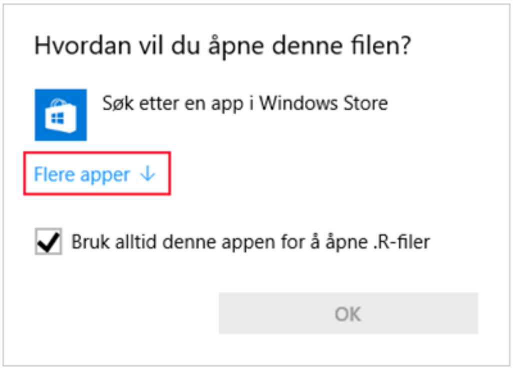
You will now have a folder named "IRiR-master". In this folder, open the file "NVE_IRiR.R".



IRiR-master

Navn	Endringsdato	Type	Størrelse
Data	17.08.2017 13.43	Filmappe	
Results	17.08.2017 13.43	Filmappe	
R-script	17.08.2017 13.43	Filmappe	
.gitignore	17.08.2017 13.43	GITIGNORE-fil	1 kB
3dploting.R	17.08.2017 13.43	R-fil	1 kB
IRiR.Rproj	17.08.2017 13.43	R Project	1 kB
NVE_IRiR.R	17.08.2017 13.43	R-fil	3 kB
README.md	17.08.2017 13.43	MD-fil	1 kB
Variablelist - translated and explained.xlsx	17.08.2017 13.43	Microsoft Excel-re...	26 kB

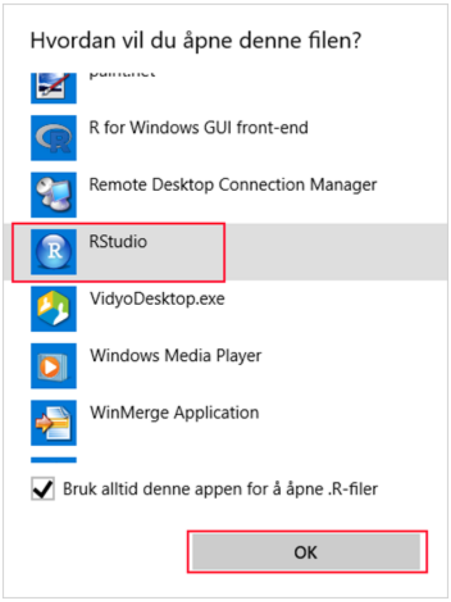
Choose to open these files with RStudio.



Hvordan vil du åpne denne filen?

Søk etter en app i Windows Store

✓ Bruk alltid denne appen for å åpne .R-filer

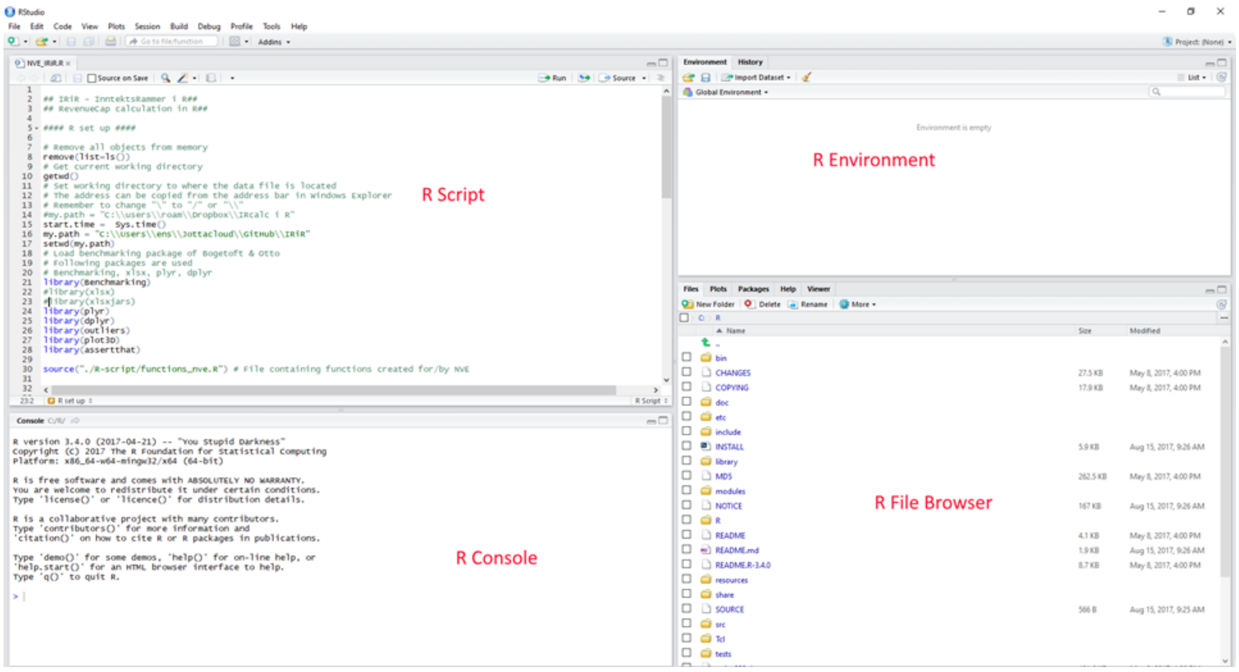


Hvordan vil du åpne denne filen?

✓ Bruk alltid denne appen for å åpne .R-filer

2.3 Set the work directory and load packages

The R script will by default display the loaded script. In this R script window, you must find the script line which says "my.path = …". Here you change the path so that it points to the folder where you installed the script on your computer.

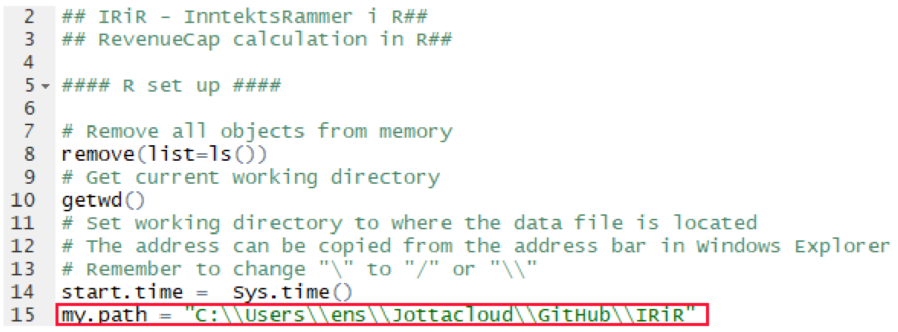


```
## IRiR - InntektsRammer i R##
## RevenueCap calculation in R##

#### R set up ####

# Remove all objects from memory
remove(list=ls())
# Get current working directory
getwd()
# Set working directory to where the data file is located
# The address can be copied from the address bar in Windows Explorer
# Remember to change "\" to "/" or "\\"
#my.path = "C:\\users\\roam\\Dropbox\\IRcalc i R"
start.time =  Sys.time()
my.path = "C:\\Users\\ens\\Jottacloud\\GitHub\\IRiR"
setwd(my.path)
# Load benchmarking package of Bogetoft & Otto
# Following packages are used
# Benchmarking, xlsx, plyr, dplyr
library(Benchmarking)
#library(xlsx)
#library(xlsxjars)
library(plyr)
library(dplyr)
library(outliers)
library(plot3D)
library(assertthat)

source("./R-script/functions_nve.R") # File containing functions created for/by NVE
```

```
R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> 
```

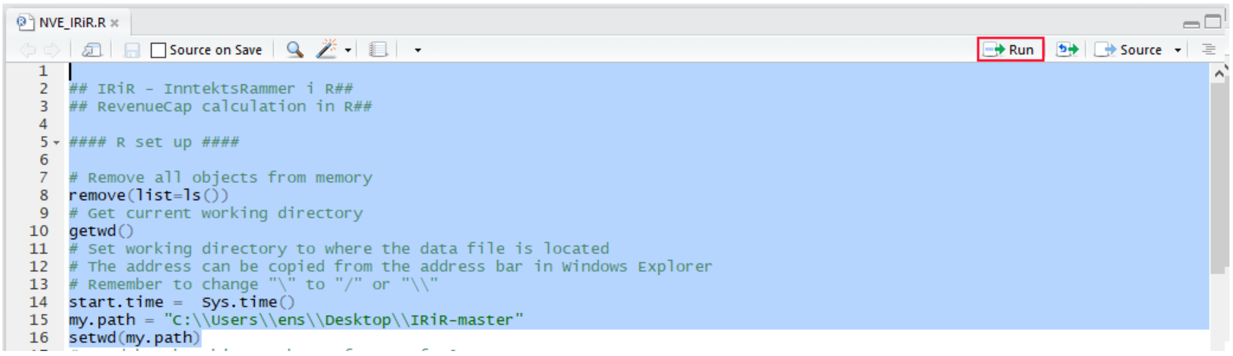
Name	Size	Modified
..		
bin		
CHANGES	27.5 KB	May 8, 2017, 4:00 PM
COPYING	17.9 KB	May 8, 2017, 4:00 PM
doc		
etc		
include		
INSTALL	5.9 KB	Aug 15, 2017, 9:26 AM
library		
MD5	262.5 KB	May 8, 2017, 4:00 PM
modules		
NOTICE	167 KB	Aug 15, 2017, 9:26 AM
R		
README	4.1 KB	May 8, 2017, 4:00 PM
README.md	1.9 KB	Aug 15, 2017, 9:26 AM
README.R-3.4.0	8.7 KB	May 8, 2017, 4:00 PM
resources		
share		
SOURCE	566 B	Aug 15, 2017, 9:25 AM
src		
Tcl		
tests		

See screen shot below regarding defining work directory, change the file path indicated in the red box.



```
## IRiR - InntektsRammer i R##
## RevenueCap calculation in R##

#### R set up ####

# Remove all objects from memory
remove(list=ls())
# Get current working directory
getwd()
# Set working directory to where the data file is located
# The address can be copied from the address bar in Windows Explorer
# Remember to change "\" to "/" or "\\"
start.time =  Sys.time()
my.path = "C:\\Users\\ens\\Jottacloud\\GitHub\\IRiR"
```

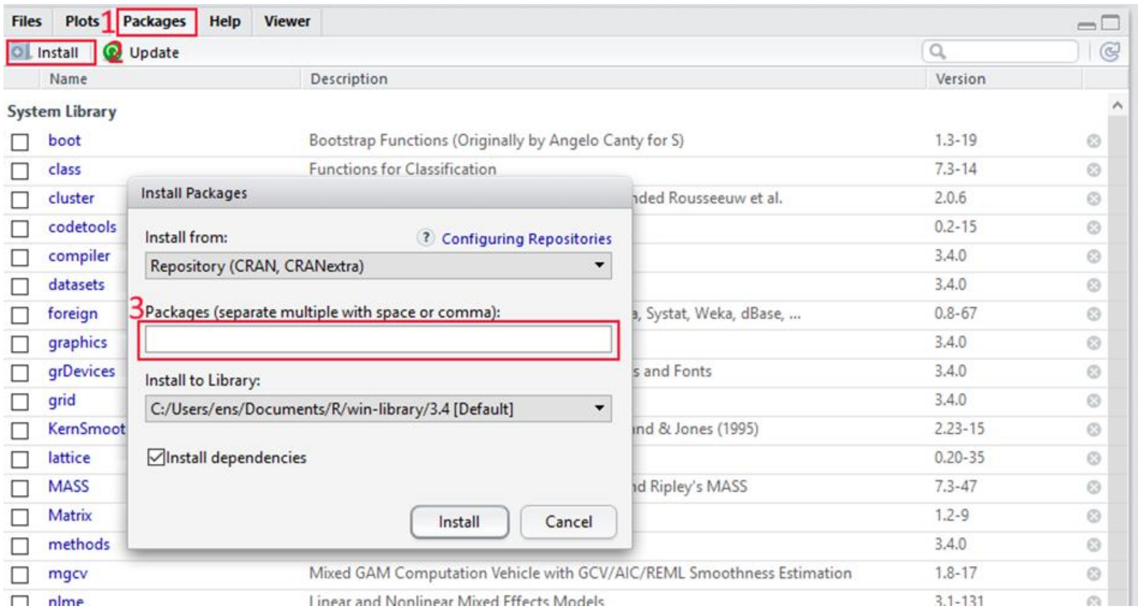
After changing the "my.path" value, save the script (CTRL+S). Mark the lines from top to the code line "setwd(my.path)", and click run in the top right corner of the Script window (or use CTRL+R).



```

## IRiR - InntektsRammer i R##
## RevenueCap calculation in R##

#### R set up ####

# Remove all objects from memory
remove(list=ls())
# Get current working directory
getwd()
# Set working directory to where the data file is located
# The address can be copied from the address bar in Windows Explorer
# Remember to change "\" to "/" or "\\"
start.time =  Sys.time()
my.path = "C:\\Users\\ens\\Desktop\\IRiR-master"
setwd(my.path)
```

Notice that the environment has changed, and now displays the working directory.

Now you need to install the packages that are used in the script. As of 17th August 2017 these packages are: Benchmarking, plyr, dplyr, outliers, plot3D and assertthat. "assertthat" should install by only running the line "library(assertthat)".



Name	Description	Version
System Library		
boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-19
class	Functions for Classification	7.3-14
cluster	nded Rousseeuw et al.	2.0.6
codetools		0.2-15
compiler		3.4.0
datasets		3.4.0
foreign	, Systat, Weka, dBase, ...	0.8-67
graphics		3.4.0
grDevices	s and Fonts	3.4.0
grid		3.4.0
KernSmoot	nd & Jones (1995)	2.23-15
lattice		0.20-35
MASS	d Ripley's MASS	7.3-47
Matrix		1.2-9
methods		3.4.0
mgcv	Mixed GAM Computation Vehicle with GCV/AIC/REML Smoothness Estimation	1.8-17
nlme	Linear and Nonlinear Mixed Effects Models	3.1-131

Install Packages

Install from: Configuring Repositories

Repository (CRAN, CRANextra)

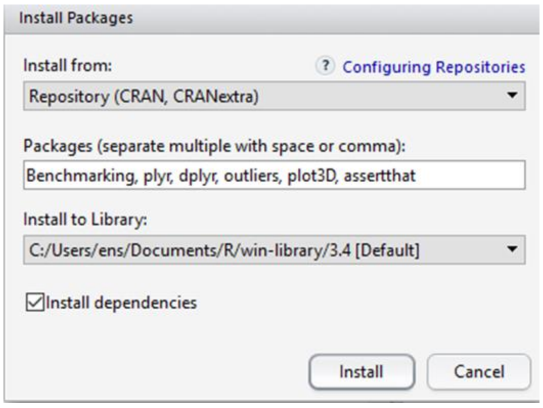
3 Packages (separate multiple with space or comma):

Install to Library:

C:/Users/ens/Documents/R/win-library/3.4 [Default]

☑ Install dependencies

You can install the remaining packages by choosing “Packages” in the R file browser, then “Install” (see screenshot above). Type the package names in the “Install packages” window separated by commas and click install (see screenshot below).


Install Packages

Install from: Configuring Repositories

Repository (CRAN, CRANextra)

Packages (separate multiple with space or comma):

Benchmarking, plyr, dplyr, outliers, plot3D, assertthat

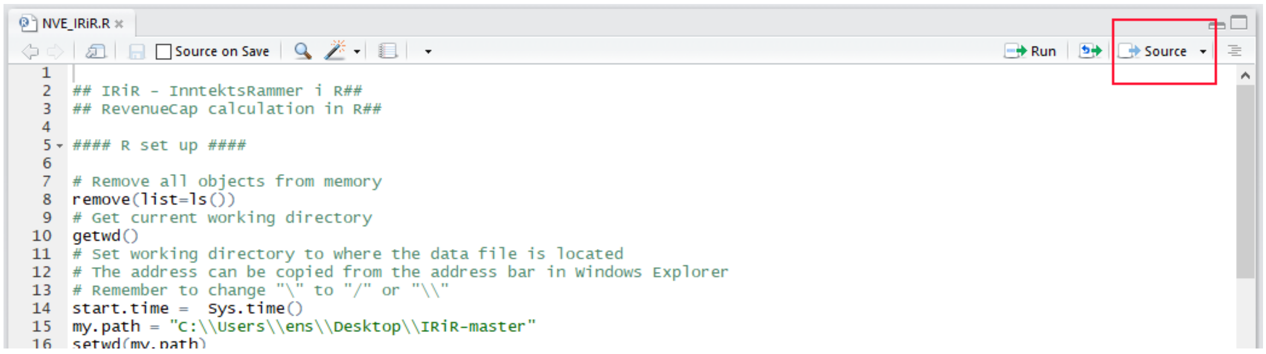
Install to Library:

C:/Users/ens/Documents/R/win-library/3.4 [Default]

☑ Install dependencies

2.4 Calculating Revenue Caps

Click “Source” in the top right corner of the script window to run the entire code and calculate revenue caps.



```

## IRiR - InntektsRammer i R##
## RevenueCap calculation in R##

#### R set up ####

# Remove all objects from memory
remove(list=ls())
# Get current working directory
getwd()
# Set working directory to where the data file is located
# The address can be copied from the address bar in Windows Explorer
# Remember to change "\" to "/" or "\\"
start.time =  Sys.time()
my.path = "C:\\Users\\ens\\Desktop\\IRiR-master"
setwd(my.path)
```

You can also do this by marking selected parts of the script, and then select “Run”. For detailed descriptions of the code in the scripts, see the following chapters.

2.4.1 Bootstrap settings

The purpose of bootstraping in the model is described in chapter 3.3. Please be aware that these settings are defined in the NVE_IRiR-file as described below.

- Bootstrap settings:
  - `BS.new`: a dummy variable that determines whether to calculate new bootstrap estimates (1) or reuse the latest calculations (0). Bootstrapping is used in stage two of the RC model to correct for selection bias in the DEA scores from stage one. You can read more about this procedure in chapter 4.2. It is important to remember that we cannot recreate any bootstrap results since they are based on some random draws. If we want to make some small changes and observe the effects, it may be better to reuse the latest calculations.
  - `BS.ite`: the number of iterations in the bootstrap calculation. A higher number gives more reliable results. We believe 2 000 is sufficient for regular calculations. For final notice or decision, we will be using 100 000 iterations. Please notice that a bootstrap calculation with more iterations will require more calculation time.

3 Set-up

3.1 Configuration, assumptions and data import (0_1_Config_Assumptions_Data.R)

This script defines key parameters, imports the data basis, and assigns companies to different evaluation groups. It also includes some smaller adjustments done to the imported data basis.

The RC is calculated based on economic and technical data from two years prior. In the following script, the year for which the RC applies will be referred to as the RC year (t, denoted `y.rc`). The cost base year, which is two years prior to the RC year, will be referred to as the CB year (t-2, denoted `y.cb`).

The RC calculation in R can be run in two different modes: notice or decision. Normally, the notice is calculated during November in the year prior to the revenue cap year (t-1). Here the price of network losses, the interest rate and the consumer price index are estimates. The decision is normally calculated in February the year after the revenue cap year (t+1). In principle, the decision shall only differ from the notice if the estimates for the parameters are wrong.

3.1.1 Defining key parameters

- Parameters:
    - `pnl.rc`: the price of network losses (NOK/MWh) for the RC year. We later multiply it with the network loss in MWh for local distribution grids.
        - For the notice, this price is estimated based on quarterly forward contracts for the RC year. The contract prices are weighted with quarterly gross consumption, where an average of quarterly gross consumption from 2002 to the CB year is used. Finally, a premium of 11 NOK is added.
        - For the decision, this price is based on actual volume-weighted monthly prices for the RC year. Monthly prices are daily averages of the actual local area prices, collected from Nord Pool Spot[5]. The weights are constructed based on monthly gross consumption of general supply for the RC year, collected from Statistics Norway[6]. Prices of network losses are calculated for each of the five pricing areas in Norway, and then a premium of 11 NOK is added.
    - `y.avg`: these are the years included in the calculation of average values. It contain five years: the CB year, and the four years prior.
    - `y.cb`: the cost base year (t-2), two years prior to the RC year. The RC is calculated based on economic and technical data from this year.
    - `y.rc`: the revenue cap year (t), or the year for which the revenue cap is calculated.
    - `y.hist.pen`: the years included in the calculation of average pension costs. You can find a more thorough explanation of the treatment of pension costs in chapter 3.2.3.
- NVE interest rates:

[5] http://www.nordpoolspot.com/Market-data1/#/nordic/table
[6] Kilde

    - `NVE.ir`: this is a list of the annual regulatory rates of return, decided by a weighted average cost of capital (WACC) model. The rate of return is also referred to as the NVE interest rate. Collectively the industry will receive this interest rate on its invested capital[7].

- Decision and notice mode:
    - `decision`: a dummy variable that decides whether the scripts calculates the RC for a decision (1) or a notice (0).
    - Notice mode:
        - `NVE.ir.t_2`: the actual interest rate for the CB year.
        - `NVE.ir.est`: the estimated interest rate for the RC year.
        - `sysp.t_2`: the system price in the CB year, which is used in the DEA-model for both notice and decision modes. It is based on monthly average system prices collected from Nord Pool Spot for the CB year. These prices are weighted with monthly gross consumption of general supply, collected from Statistics Norway, to form a weighted average system price for the cost base year. Finally, a premium of 11 NOK is added.
    - Decision mode:
        - `NVE.ir.t`: the actual interest rate of the RC year.
    - We add a loop to the script, which helps us choose the correct interest rates and prices depending on whether we run the script in decision or notice mode.
        - `ir.dea`: equals the interest rate from the CB year (`NVE.ir.t_2`) for both decision and notice mode.
        - `NVE.ir.RC`: equals the interest rate from the RC year (`NVE.ir.t`) for decision mode, and the estimated interest rate for the RC year (`NVE.ir.est`) for notice mode.
        - `pnl.dea`: equals the system price for the CB year (`sysp.t_2`) for both decision and notice mode.
- Economic assumptions:
    - `wpc`: the working capital premium. This premium is added to the book values to account for working capital.
    - `rho`: the norm cost share in the revenue cap model. This decides how much weight is put on the cost norm.
    - `gci.p`: the price of grid components in interface between regional and local distribution grids.
- Estimated costs for CB year:
    - `lrt_RC_dec.y.cb`: the costs that were estimated in the RC calculation two years prior, based on actual costs from four years prior.

[7] https://www.nve.no/energy-market-and-regulation/revenue-regulation/the-wacc-model/ (ENG)
https://www.nve.no/elmarkedstilsynet-marked-og-monopol/okonomisk-regulering-av-nettselskap/reguleringsmodellen/referanserenten/ (NOR)

3.1.2 Data import

We import data sets from csv-files into R:

- `dat`: this data set includes detailed accounting information from each company, as well as technical information about the network grid.
- `id`: the companies' IDs are also imported, and then merged into `dat`.
- `ap.t_2`: the data set contains the correct area prices for each company from the CB year. These are later used to calculate each company's actual cost of network losses in the CB year. We merge the data set into `dat`.
    - Since the rest of the script shows values in thousand NOK, we divide the `ap.t_2` by 1000.

3.1.3 Configuration / Selecting companies for different evaluation types

In principal, all network companies shall be included in the DEA model, either for the regional distribution (RD) grids or the local distribution (LD) grids. However, some companies may be subject to special treatment. The special companies are sorted into different groups below. The companies that are not put into a group will be included in the normal DEA model.

- For the LD grid, we separate the companies into the following groups:
    - `ld_OOTO`: companies that are out of the ordinary (OOTO). These companies have subscribers < 500 or km high voltage grid < 100. Their performances are measured against their own historical performance.
    - `ld_av.eff`: companies with structural breaks in their data. These are set to average efficiency.
    - `ld_sep.eval`: companies that are included in the DEA model, but are only allowed to be their own peer.
    - `ld_no.rc`: companies excluded from the revenue cap calculation.
- For the RD grid, we separate the companies into the following groups:
    - `rd_OOTO`: companies that are out of the ordinary (OOTO). These companies have defined outputs < 4000 (outputs are defined in chapter 4.1.1) or zero km overhead lines. Their performances are measured against their own historical performance.
    - `rd_av.eff`: companies with structural breaks in their data. These are set to average efficiency.
    - `rd_sep.eval`: companies with total costs < 15 mill. NOK. These are included in the DEA model, but are only allowed to be their own peer.
    - `rd_no.rc`: companies excluded from the revenue cap calculation.

3.1.4 Consumer price index

- `cpi`: a list of the Norwegian consumer price index from 2004 to the RC year. Collected from Statistics Norway.
- `cpi.l`: a list of a price index for the service sector, which is dominated by the growth in annual earnings. Collected from Statistics Norway.

- `y.cb.cpi.l.factor`: an inflation factor, later used to inflate the values from the CB year into RC year values.
- `y.cb.cpi.factor`: an inflation factor, later used to inflate the values from the CB year into RC year values.

3.2 Calculating input values for DEA (0_2_Calculated_Input_Values.R)

This script calculates the input values for the DEA, including price index adjustments, calculating average values and composing the companies' cost bases.

3.2.1 Variable for costs associated with grid components in interface between LD and RD grid

Some grid components are located in interface between the LD and the RD grid. These components are registered as part of the local distribution grid, but we need to manually create a separate variable that displays the costs for these components.

- `ld_gci.dummy`: a dummy variable for costs associated with grid components in interface. The variable will be 1 for companies in the groups `ld_av.eff` and `ld_ooto`, and 0 for all other companies.
- `ld_gci.cost`: for companies with grid components in interface, the number of components (`ld_gci`) is multiplied with the price per grid component (`p.gci`). For other companies the variable `ld_gci.cost` will be assigned a value of zero.

3.2.2 CPI and CPI.L adjustment

In order to make all costs fixed in the price of the RC year, we adjust some variables with CPI or CPI.L. CPI is the Norwegian consumer price index, and the CPI.L is a price index for the service sector, dominated by the growth in annual earnings. Both collected from Statistics Norway[8].

We adjust the variables for cost of energy not supplied (CENS) with CPI, and the variables for OPEX, wages and pensions with CPI.L. We use loops to multiply the yearly values with the correct CPI factor, and then add "`fp_`" in front of the new variables to indicate that the new variables show costs in fixed prices.

- CPI adjustment: we adjust the same variables for all grid levels.
    - `v_cpi`: a vector including the variables that will be adjusted with CPI. An explanaition of these variables will follow in chapter 3.2.4.
- CPI.L adjustment: we adjust the same variables for all grid levels.
    - `v_cpi.l`: a vector including the variables that will be adjusted with CPI.L. An explanaition of these variables will follow in chapters 3.2.3 and 3.2.4.

3.2.3 Pension costs

Pension costs fluctuate between years. This leads to unnecessary and unwanted variations in the RC between years. From the RC calculation for 2016, we operate with smoothed pension costs in order to reduce the fluctuations. To achieve this we use average costs over the past five years, which includes CPI adjusted pension costs from the CB year and the four years prior.

As a transition to the new approach described above, the years prior to 2013 will include CPI adjusted average pension costs for the fixed period 2007-2013. This transition period will be over after the RC calculation for 2019.

- Historical averages, 2007-2013: we use a loop to calculate the average pension costs (`_pens`, `_pens.eq`, `_impl`) for the period 2007-2013, `y.hist_pen`. The loop then assigns the average

[8] ssb.no, tables 03013 (CPI) and 11118 (CPI.L)

costs to the years prior to 2013. We paste "`hist_`" in front of these new variables to state that they are historical averages.

- Five-year historical averages: we use a loop to calculate the average pension costs (`_pens`, `_pens.eq`, `_impl`) for the historical period, `y.avg`. The loop then assigns the average costs to the five-year historical period. We paste "`av_`" in front of these new variables to state that they are averages.
- The pension costs base: this is the total of the average pension costs for each network level.
    - `_pcb`: the pension cost base which is later included in the `OPEX`. It consists of:
        - `_pens`: pension costs.
        - `_pens.eq`: pension costs recognized in equity. This does not include costs associated with implementation of other pension regimes.
        - `_impl`: implemented pension costs. These are costs associated with implementation of other pension regimes. Since companies rarely switch to a new pension regime, this is normally a one-time effect.

3.2.4 Cost base

We calculate the cost base for each company and network level.

- `TOTXDEA`: a company's total expenditures, which are later used as an input in DEA. The calculation is similar for all network levels, and consists of:
    - `_OPEX:` a company's total operating expenditures. This consists of:
        - `_OPEXxS`: operation and management costs. This does not include salaries or pension costs.
        - `_sal`: salaries and other personnel costs. This does not include pensions.
        - `_sal.cap`: salaries and other personnel costs recognized in equity. These are labor costs connected to investment projects, which are capitalized.
        - `_pcb`: the pension cost base, which includes pension costs, pension costs recognized in equity and costs associated with implementation of new pension regimes.
        - (-) `_391`: other operating revenues which are specified in accounting item 391 in eRapp. We subtract these revenues from the OPEX.
        - (-) `_cga`: costs associated with grid assessments in the LD and RD grid. We subtract these costs from the OPEX for the LD and RD grid (this does not exist for the transmission grid).
    - `_RAB * ir.dea`: the regulatory asset base multiplied with the interest rate returns a company's calculated return on invested capital. The `_RAB` consists of:
        - `_rab.sf`: self-funded book values, including a 1 % working capital premium.
        - `_rab.gf`: grants-funded book values, including a 1 % working capital premium.
    - `_DEP`: the depreciation of the network components which includes:

        - `_dep.sf`: depreciation associated with self-funded assets.
        - `_dep.gf`: depreciation associated with grants-funded assets.
    - `_cens`: the cost of energy not supplied (CENS). This is a measure of the calculated value of lost load for the customers. The CENS arrangement also provides an incentive for network operators to maintain their assets properly, and to ensure necessary investments in order to avoid power outages at a socioeconomic efficient level.
    - `ld_nl.NOK`: the cost of network losses. This is only included in OPEX for the LD grid, and consists of the network losses in MWh (`ld_nl`) multiplied by the price of network losses (`pnl.dea`).
    - `ld_gci.cost`: the cost of grid components in interface LD and RD grid. This is only included in OPEX for the LD grid.

3.2.5 Five-year historical averages of input and output variables

As a preparation for the DEA calculation, we calculate five-year historical averages for all input and output variables. A loop helps us obtain these values. We paste "`fha_`" in front of the name of the new variables, indicating that they include five-year historical averages. The inputs and outputs are more thoroughly explained in chapter 4.1.1.

3.3 Company selection (0_3_Company_Selection.R)

This script creates new data frames for the companies that will be included in the normal evaluation. To do this, we use the already defined evaluation groups from chapter 3.1.3.

- Local distribution:
    - `ld_EVAL`: a dummy variable for LD companies. If the value is 1, the company will be part of the normal evaluation. The first condition is that the company must have total expenditures (`TOTXDEA`) greater than zero. The second condition is that the company's ID is not included in any of the groups `ld_av.eff`, `ld_ooto`. The companies in these groups will be part of a separate evaluation. Companies in the group `y.avg` are set equal to 1. Companies in the `ld_no.rc` are included in the dataset but shall not have a revenue cap for relevant year.
    - `ld_EVAL`: a data frame containing observations from the past five years for all companies included in normal evaluation, and companies only allowed to be their own peers (`ld_sep.eval`).
    - `ld_DEA_id`: this is a vector containing IDs of all companies in DEA. It is later used for datawrangling.
- Regional distribution: the exact same process is performed for companies with RD grids.

4 Revenue cap calculation

4.1 Stage 1: calculating DEA-scores (1_0_DEA.R)

In stage one of the RC calculation we use benchmarking to compare one company's performance to the performance of other companies. The benchmarking approach used in the Norwegian regulation is the Data envelopment analysis (DEA). This is a recognized method for measuring a company's productivity or efficiency. The DEA assumes that companies can be compared to one another by looking at the relationship between inputs and outputs.

Figure 1 illustrates the best practice production frontier, which is the line between the most efficient companies (observations N1, N2 and N3). It can be described as the performance standard or the technology. We refer to the companies located on the production frontier as peer units.

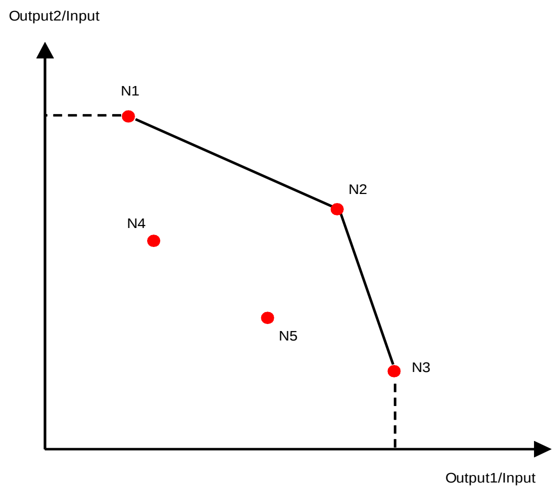


Figure 1: Illustration of the DEA production frontier

The companies are given an efficiency score based on their performance relative to the production frontier. To assign efficiency scores we first need to create target units for all companies. In Figure 2 we illustrate the creation of target units N4* and N5*. The target units are placed on the production frontier, and are constructed by directly comparing each company to a combination of peer units. I.e. the peer units of company N4 are N1 and N2 since the target unit N4* is located between these two peers on the production frontier. We can consider the target units as each company's opportunity for improvement.

The relationship between the distance from origin to N4* and the distance between origin and N4 decides the company's DEA score. The score will be between 0 % and 100 %, where higher scores represents higher efficiency.

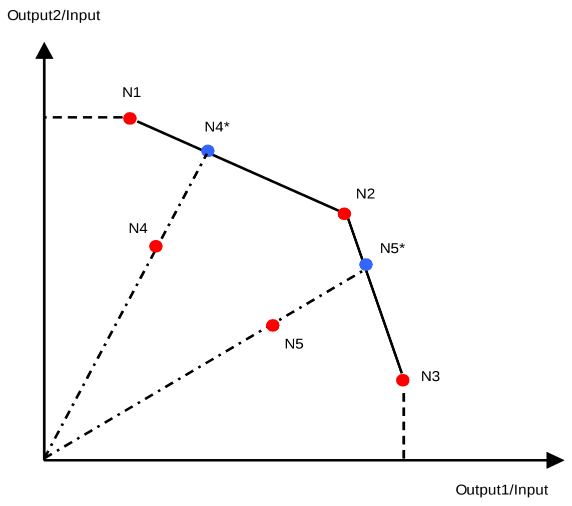


Figure 2: Illustration of the DEA target units

The production frontier in the Norwegian regulation is constructed by five-year historical averages. This ensures that all companies are compared to the same frontier, and gives a more symmetrical treatment for the most efficient companies. Another reason for using five-year historical averages is to stabilize the production frontier, and we consider five years to be sufficient to smooth out any irregular cost variations. Yearly company data are compared to the average production frontier. If a company becomes more cost efficient compared to its historical average, it will quickly receive a better DEA score. A company that is a peer unit, based on historical averages, can outperform itself and obtain a DEA score above 100 %[9].

Our DEA model assumes constant returns to scale, which favors neither the largest nor the smallest company, but give them incentives to adjust to an optimal scale. We run separate analyses for the local and regional distribution grid. Both have one input, which is total expenditures, and several outputs, which are explained more thoroughly below.

4.1.1 The production frontier – five-year historical averages

In the DEA, we use five-year historical averages of input and outputs to create the efficient frontier. This includes observations from the CB year, as well as the four years prior. In preparation for the DEA, we group five-year historical averages of input and outputs separately for both local and regional distribution grid.

- Local distribution grid:
    - `X.avg.ld`: a group containing the five-year historical input used in DEA.
        - `fha_ld_TOTXDEA`: five-year historical average of a company's total expenditures/costs. This includes OPEX, CENS, depreciations, return on invested capital, and network losses. A more detailed review of the components included in this variable can be found in chapter 3.2.4.
    - `Y.avg.ld`: a group containing the five-year historical outputs used in DEA. The elemental task of the local distribution grid is to transport electricity from the input points of production to the output points of consumption. Each company with a local distribution network grid has its own distinct license area. Within this area, the network company is obliged to supply all

[9] NVE-report 2/2012: http://publikasjoner.nve.no/hoeringsdokument/2012/hoeringsdokument2012_02.pdf (NOR)

possible subscribers with access to the grid. Together, the outputs aim to capture the companies' overall task.

        - `fha_ld_sub`: five-year historical average of the number of subscribers supplied by each company. This output reflects the demand facing each company.
        - `fha_ld_hv`: five-year historical average of total length (in kilometers) of the high voltage grid, including overhead lines, underground cables and subsea cables. This output reflects the transportation of electricity, as it accounts for the geographical extent of the grid.
        - `fha_ld_ss`: five-year historical average of the number of substations. This output reflects the distribution of the demand within the company's license area, and may tell us whether a company transports electricity in rural or urban areas.

- Regional distribution grid:
    - `X.avg.rd`: a group containing the five-year historical input used in DEA.
        - `fha_rd_TOTXDEA`: five-year historical average of a company's total expenditures/costs. This includes OPEX, CENS, depreciations and return on invested capital. A more detailed review of the components included in this variable can be found in chapter 3.2.4.
    - `Y.avg.rd`: a group containing the five-year historical outputs used in DEA. The elemental task of the regional distribution grid is the transportation and transformation of electricity. To quantify this task we give each network component a weighted value based on its technical characteristics. These weighted values represent standard costs associated with the network components, including annual capital costs and OPEX[10]. Together, the outputs aim to capture the companies' overall task.
        - `fha_rd_wv.ol`: five-year historical average of weighted value of overhead lines. Relevant properties of overhead lines are voltage, mast element, diameter, type, system, top line, marked aviation obstacles, and number of lines.
        - `fha_rd_wv.uc`: five-year historical average of weighted value of underground cables. Relevant properties of underground cables are voltage, diameter, type, system, and location.
        - `fha_rd_wv.sc`: five-year historical average of weighted value of subsea cables. Relevant properties of subsea cables are voltage, diameter, type and system.
        - `fha_rd_wv.ss`: five-year historical average of weighted value of station components. The station component is a collection of four different components: stations, transformers, circuit ends, and compensation systems. These have two or three relevant properties each.

4.1.2 Observations from the cost base year

The actual data is observations of input and outputs from the CB year. In the DEA-model, we compare these observations to the production frontier, which is based on average observations. In preparation for the DEA-calculation, we group the actual input and outputs separately for both local and regional

[10] NVE-report 110/2015: http://publikasjoner.nve.no/rapport/2015/rapport2015_110.pdf (NOR)

distribution grid. The explanations of inputs and outputs for the five-year historical averages in chapter 4.1.1 also applies to the observations from the CB year.

- Local distribution grid:
    - `X.cb.ld`: a group containing observations of input from the CB year.
        - `ld_TOTXDEA`: a company's total expenditures/costs.
    - `Y.cb.ld`: a group containing observations of outputs from the CB year.
        - `ld_sub`: the number of subscribers supplied by each company.
        - `ld_hv`: total length (in kilometers) of the high voltage grid, including overhear lines, underground cables and subsea cables.
        - `ld_ss`: the number of network stations.
- Regional distribution grid:
    - `X.cb.rd`: a group containing observations of input from the CB year.
        - `rd_TOTXDEA`: a company's total expenditures/costs.
    - `Y.cb.rd`: a group containing observations of outputs from the CB year.
        - `rd_wv.ol`: weighted value of overhead lines.
        - `rd_wv.uc`: weighted value of underground cables.
        - `rd_wv.sc`: weighted value of subsea cables.
        - `rd_wv.ss`: weighted value of station components.

4.1.3 Rounding data used in DEA

All input and output variables used in DEA are rounded to the closest integer, i.e. thousand NOK.

4.1.4 DEA calculation

To calculate the actual efficiency scores we use the command `dea` from the "`Benchmarking`" package[11]. We are interested in both efficiency scores and lambdas, which emerges from the DEA. We perform the DEA four times for both the LD and the RD grid. The calculations are identical for the two grids; therefore, we only describe the DEA for the LD grid here.

- Regular DEA calculation for companies in the `ld_EVAL` group. These DEA calculations provide us with the companies' efficiency scores.
    - `dea.cb.avg.ld`: DEA calculation of actual observations compared to a front constructed by average observations.
        - `X/Y`: actual input and outputs from the CB year.
        - `XREF/YREF`: average input and outputs from the five-year historical period. The front only includes companies in the `ld_eval` group.

[11] Created by P. Bogetoft and L. Otto: https://cran.r-project.org/web/packages/Benchmarking/Benchmarking.pdf (ENG)

        - `ld_EVAL`: the efficiency scores are stored in this data frame. These are now called `ld_eff.s1.cb`.
    - `dea.avg.avg.ld`: DEA calculation of average observations compared to a front constructed by average observations.
        - `X/Y`: average input and outputs from the five-year historical average.
        - `XREF/YREF`: average input and outputs from the five-year historical period. The front only includes companies in the `ld_eval` group.
        - `ld_EVAL`: the efficiency scores are stored in this data frame. These are now called `ld_eff.s1.avg`.
- Adapted DEA calculation for companies in the `ld_sep.eval` group. These companies can only be their own peers, and were not included in the regular DEA calculations above. We already have the efficiency scores for all companies, but we are missing the lambdas for the `ld_sep.eval` group.
    - `ld_lambda`: a new data frame containing the lambdas from the first calculation, `dea.cb.avg.ld$lambda`. We also want to include the lambdas from the adapted calculation. Therefore, we add a loop with DEA that collects these:
        - `dea.sep.cb.avg.ld`: adapted DEA calculation of actual observations (`X` and `Y`) compared to a front constructed by average observations (`XREF` and `YREF`).
    - `ld_lambda.avg`: a new data frame containing the lambdas from the second calculation, `dea.avg.avg.ld$lambda`. We also want to include the lambdas from the adapted calculation. Therefore, we add a loop with DEA that collect these:
        - `dea.sep.avg.avg.ld`: adapted DEA calculation of average observations (`X` and `Y`) compared to a front constructed by average observations (`XREF` and `YREF`).

4.2 Stage 2: correcting for environmental factors (2_0_Stage2_GeoCorrection.R)

The network companies in Norway are faced with different environmental factors, also known as z-variables. Some companies may operate in areas with dense forests and steep terrain, while others may operate in exposed areas with strong winds, or very cold temperatures. This gives rise to cost differences between companies since it can be more costly to operate under certain environmental factors. Therefore, we must correct for the different environmental factors when we calculate the companies' efficiency scores.

In stage 2 of the RC calculation, the DEA scores from stage 1 is increased if the company has tougher environmental factors than its target unit, and decreased if its environmental factors are easier than the target unit's. The target unit's environmental variables are constructed by multiplying the peer units' environmental variables with their reference weight (the lambdas which were collected in chapter 4.1.4):

$$z_{i,j}^T = \left(w^{P1} * z_j^{P1}\right) + \left(w^{P2} * z_j^{P2}\right) \tag{1}$$

This second stage of the RC calculation uses a regression with DEA scores from the first stage as dependent variable and environmental factors as independent variables. The DEA scores from the first stage is bias corrected using bootstrapping. The independent variables are the difference between the values of the environmental variables for each DSO and its corresponding target unit. The regression estimates the coefficients (β_j) for all environmental variables:

$$E_i = \alpha + \sum_j \beta_j (z_{i,j} - z_{i,j}^T) + u_i \tag{2}$$

The total adjustment value for company *i* is the sum of all *j* coefficients multiplied with the difference between the values of all *j* environmental variables for company *i,* and its corresponding target unit:

$$ADJ_i = \sum_j \beta_j (z_{i,j} - z_{i,j}^T) \tag{3}$$

Lastly, we subtract the adjustment value from the DEA score from stage 1 for all companies. This leaves us with the second stage DEA scores, which are now corrected for environmental factors.

4.2.1 Bootstrapping

In stage two of our DEA model, we are going to perform a regression. The regression will provide coefficients that helps us correct the DEA scores for environmental factors. The dependent variable in this regression are the DEA scores from stage one. Since these are all set based on the same peer units, we may have a problem with serial correlation. This means that there is a covariance between the dependent variables for the different companies. The problem of serial correlation is especially evident since the DEA is used on a limited number of companies. One way to solve this problem is by correcting for this bias. Selection bias may arise in DEA for companies located in areas of the production frontier where there are none or few other observations. Selection bias results in too high DEA scores for these companies.

In DEA, we correct for this selection bias by using bootstrapping. Bootstrapping uses different statistical elements to calculate a corrected DEA result. The use of bootstrapping will reduce the problem of serial correlation, which implies better estimates of how environmental factors affect the companies' costs. We only use the bias-corrected, bootstrapped DEA scores for the stage two regression.

- `ld_bs`: if the `BS.new` dummy from chapter 2.4.1 is set to 1, we will run a new Bootstrap on the DEA results. If it is set to 0, the last Bootstrapped results are used. Every time a new Bootstrap is run, the new results will overwrite the previous results. If the `ld_bs` data frame is empty, meaning no bootstrapped results are stored, you will receive an error message.

- `ld_eff.bs.is2.cZ`: the variable name for the bootstrapped efficiency scores. These are merged into the data frame `ld_EVAL`.

4.2.2 Constructing Z-variables for local distribution

The map of Norway is divided into map grid squares of 100*100 meters. For each square, we have information about all relevant environmental factors, including inclination, vegetation, climate etc. We also know in which squares each network company has grid components. We use this information when we determine the different geographical factors.

There should not be too much correlation between the different environmental factors, since they are independent variables in the second stage regression. If the variables are correlated, the coefficients estimated from the regression may be wrong. To avoid this problem we use principal component analysis (PCA), which aims to create an index variable that captures the variations in the correlated variables[12]. We create four index variables, also referred to as Geo variables. Three of these are used in the LD model and one is used in the RD model. The environmental variables are first sorted into new data frames, before they are subject to the PCA. We use the command `z.est` to conduct the PCA in R. This function is more thoroughly explained in chapter 4.5.

- `Geo1.comp`: groups the variables included in the Z-variable, `ldz_Geo1` ("Fjellbekk"). This variable adjusts for mountain environments and rugged terrain. It includes:
    - `ldz_inc.av:` average inclination in a network company's map grid squares.
    - `ldz_f7:` the share of a company's network components located in deciduous forest.
    - `ldz_cmpp.sz`: installed capacity of micro power plants. We scale the value based on norm cost to make it size independent.
- `ldz_Geo1` ("Fjellbekk"): a Z-variable is constructed by using PCA on the group `Geo1.comp`. We use the command `z.est`, which is more thoroughly explained in chapter 4.5. The new variable is included in the data frame `ld_EVAL`.
- `Geo2.comp`: groups the variables included in the Z-variable, `ldz_Geo2` ("Øyvind"). This variable adjusts for costal environments. It includes:
    - `ldz_wind2_cod`: average wind (squared) divided by average distance to coast.
    - `ldz_isl.sz`: number of supplied islands located more than 1 kilometer from mainland or other supplied islands. We scale the value based on norm cost to make it size independent.
    - `ldz_hvsc.s`: a company's share of high voltage sea cables.
- `ldz_Geo2` ("Øyvind"): a Z-variable is constructed by using PCA on the group `Geo2.comp`. We use the command `z.est`, which is more thoroughly explained in chapter 4.5. The new variable is included in the data frame `ld_EVAL`.

[12] The PCA is not an independent method of analysis, but a collective term for various multivariate statistical methods that analyzes the relationships between a large number of variables, before explaining their common underlying dimensions.

- `Geo3.comp`: groups the variables included in the Z-variable, `ldz_Geo3` ("Frost"): This variable adjusts for cold environments. It includes:
    - `ldz_snow`: average daily snow depth over a 30-year period.
    - `ldz_lat.av`: average latitude of placement of company's grid components. This variable is left censored to 65.9 degrees, meaning that companies with latitude below the threshold is given a value of 65.9.
    - `ldz_ice.av`: average number of hours where the ice load exceeds a given threshold during a year.
    - `ldz_tempneg`: average daily temperature over a 30-year period, multiplied by -1.
- `ldz_Geo3` ("Frost"): a Z-variable is constructed by using PCA on the group `Geo3.comp`. We use the command `z.est`, which is more thoroughly explained in chapter 4.5. The new variable is included in the data frame `ld_EVAL`.

4.2.3 Adjusting efficiency scores from stage one for local distribution

We use a regression based on bootstrapped DEA scores to obtain the Z-variable coefficients. These coefficients help us correct the efficiency scores from stage one for environmental factors.

- `ld_eff.bs`: we store the bootstrapped efficiency scores as a new vector. This ensure that the correct companies are included in the following regression. We omit the companies in the `ld_sep.eval` group, and companies identified as outliers by the BACON-test, from the regression. The BACON-test is described in chapter 4.5
- Calculating Z-variable coefficients for the LD grid:
    - `Geovar.ldz`: groups all Z-variables:
        - `ldz_hvug.s`: a company's share of underground cables
        - `ldz_f4`: the share of a company's network components located in coniferous forest
        - `ldz_Geo1`: adjustment for mountain environments and rugged terrain
        - `ldz_Geo2`: adjustment for coastal environments
        - `ldz_Geo3`: adjustment for cold environments
    - `ldz.coeff`: these are the z-variable coefficients. They are obtained by using the function `Zvar1`, which also includes a regression. This function is more thoroughly explained in chapter 4.5.
        - `x`: equals the companies' average expenditures, `X.avg.ld.`
        - `z`: equals the Z-variables in `Geovar.ldz.`
        - `eff`: equals the bootstrapped efficiency scores in `ld_eff.bs.`
        - `lambda`: equals the lambdas, or reference weights (`ld_lambda.avg`), from the DEA calculation that only used average data.
        - `id`: equals the companies in `X.avg.ld.`

        - `id.out`: equals the coefficients in `ld_sep.eval`. All companies in `id` are included in the regression, except those that are on the `ld_sep.eval` model.

- Z-variable adjustment: here we correct the efficiency scores for environmental factors, using the difference in Z-value relative to the target unit's Z-value.
    - `ld_eff.s2.cb`: the `Zvar2` function corrects the efficiency scores from stage 1 for environmental factors. The efficiency scores after stage 2 are returned in this variable. The function is more thoroughly explained in chapter 4.5.
        - `x`: equals the companies' average expenditures over last five years, `X.avg.ld`.
        - `eff`: equals the efficiency scores from the stage 1 DEA, `eff.cb.avg.ld`.
        - `id`: equals the companies in `X.avg.ld`.
        - `lambda`: equals the lambdas, or reference weights (`ld_lambda`), from the DEA calculation with observations from the CB year compared to average observations.
        - `coeff`: these are the Z-variable coefficients, `ldz.coeff`.
        - `z`: equals the Z-variables in `Geovar.ldz`.

4.2.4 Estimate the Z-variables for regional distribution

- `GeoR.comp`: groups the variables included in the Z-variable, `rdz_Geo1` (Helskog). This variable adjusts for inclination and dense forests. It includes:
    - `rdz_f12`: the share of a company's network components located in forest.
    - `rdz_inc.av`: average inclination in a network company's map grid squares.
- `rdz_Geo1` (Helskog): a Z-variable is constructed by using PCA on the group `GeoR.comp`. The new variable is included in the data frame `rd_EVAL`.

4.2.5 Adjusting efficiency scores from stage one for regional distribution

We conduct the same procedure for the RD grid as we did above for the LD grid.

- `rd_eff.bs`: the bootstrapped efficiency scores.
- `rdz.coeff`: the Z-variable coefficients.
- Z-variable adjustment: here we correct the efficiency scores for environmental factors, using the difference in Z-value relative to the target unit's Z-value.
    - `rd_eff.s2.cb`: the adjusted stage two efficiency scores

4.3 Stage 3: calibration of the cost norm (3_0_Stage3_Calibration.R)

Stage 3 ensures that the industry in total recovers all costs, and that the averagely efficient firm receives a return on their invested capital equal to the NVE-interest rate. In this stage, we distribute the difference between the total cost norm and the total cost base for the RC back to the industry. The difference is distributed based on each company's share of the regulatory asset base of self-funded assets.

- `ld_cb`: the calculated cost base for local distribution. It consists of costs from the CB year, which we inflate to RC year prices.
    - `fp_ld_OPEX` * `y.cb.cpi.l.factor`: inflated operation and management costs.
    - `ld_rab.sf` * `NVE.ir.RC`: return on invested capital. Regulatory asset base from CB year multiplied by the interest rate for the RC year.
    - `ld_dep.sf`: depreciation of self-funded assets in CB year.
    - `ld_cens` * `y.cb.cpi.factor`: inflated costs of energy not supplied.
    - `ld.nl` * `pnl.rc`: cost of network losses. Network losses from CB year multiplied by the price of network losses for the RC year.
    - `ld_gci.cost` * `y.cb.cpi.fatcor`: inflated costs of grid components.
- `ld_calib`: a new data frame that contains information associated with the calibration. The command `NVE_cal` calculates the cost norm for each company after calibration.
- `rd_cb`: the calculated cost base for regional distribution. It consists of costs from the CB year, which we inflate to RC year prices.
    - `fp_rd_OPEX` * `y.cb.cpi.l.factor`: inflated operation and management costs.
    - `rd_rab.sf` * `NVE.ir.RC`: return on invested capital. Regulatory asset base from CB year multiplied by the interest rate for the RC year.
    - `rd_dep.sf`: depreciation of self-funded assets in CB year.
    - `rd_cens` * `y.cb.cpi.factor`: inflated costs of energy not supplied.
- `rd_calib`: a new data frame that contains information associated with the calibration. The command `NVE_cal` calculates the cost norm for each company after calibration.

4.4 Revenue cap calculation (4_0_Revenue_Cap_Calculation.R)

This script calculates the final revenue caps. The revenue caps consists of actual costs, weighted at 40 % (1-rho) and norm costs, weighted 60 % (rho). Therefore, we need to calculate each company's actual costs and cost norm for the RC year. The actual costs in the RC year is actually costs from the CB year, inflated into RC year cost levels. These estimated costs will differ from the actual costs in the RC year, but this is adjusted for in the re-calibration in the future (t+2). In this year's RC calculation, you will see that we adjust for the difference between the estimated and actual costs for the CB year. This is what we call the re-calibration.

- We calculate OPEX like we did for the years prior to 2012. This is to ensure consistency in our calculations. This is done equally for all network levels (`ld_opex_2012`, `rd_opex_2012` and `t_opex_2012`). It equals OPEX with salaries excluded + salaries - capitalized salaries + pension costs + pension costs recognized in equity.
- `RevCap`: a new data frame that only includes observations from the CB year for selected variables.
    - o `ld_cn.cal.RAB`: each company's cost norm after calibration for LD grids. These were stored in the `ld_calib` data frame in the script for stage 3.
    - o `rd_cn.cal.RAB`: each company's cost norm after calibration for RD grids. These were stored in the `rd_calib` data frame in the script for stage 3.
- Calculating total costs:
    - o `ld_sum.cost`: includes inflated OPEX costs, inflated CENS costs, depreciations and costs of network losses.
    - o `rd_sum.cost`: includes inflated OPEX costs, inflated costs for grid assessments, inflated CENS costs, depreciations and costs of network losses.
    - o `t_sum.cost`: includes inflated OPEX costs, inflated CENS costs and depreciations.
    - o `lrt_sum.cost`: the sum of all relevant costs for all grid levels.
- Calculating cost base used in RC calculation. In addition to the total costs calculated above, the cost base also includes the regulated revenues.
    - o `ld_cost.RC`: includes inflated OPEX costs, inflated CENS costs, depreciations, regulated revenues and costs of network losses.
    - o `rd_cost.RC`: includes inflated OPEX costs, inflated costs for grid assessments, inflated CENS costs, depreciations, regulated revenues and costs of network losses.
    - o `t_cost.RC`: includes inflated OPEX costs, inflated CENS costs, depreciations and regulated revenues.
    - o `lrt_cost.RC`: the sum of the cost bases for all grid levels.
- Calculating the actual inflicted costs in the RC year for all grid levels, excluding network losses and grid assessment costs in the regional distribution grid: `lrt_cost.RC.ex.nlR.og.cgaR`.
- Calculating the actual inflicted costs in the CB year for all grid levels:
    - o `lrt_cost.cby`: includes OPEX for all grid levels, CENS for all grid levels, and costs of network losses for LD and RD, which is network losses in MWh for the CB year multiplied

with the CB year area prices. It also includes costs associated with grid assesments in the LD and RD grid (`_cga`).

- Cost norms: each company's cost norm after calibration are collected from the `ld_EVAL` and `rd_EVAL` data frames and merged into the `RevCap` data frame.
- In the data frame RevCap, we set cost norm equal to 0 for all companies with OPEX = 0 in the given grid level.
- Cost norms prior to re-calibration: we use the cost norms after calibration from stage 3. Here we add costs associated with grid comopnents in interface for LD grid, and costs of network losses in RD grid.

4.4.1 Re-Calibration

Here we calculate the companies final RCs. The RC is based on costs from the CB year, and will always differ from the actual costs from the RC year. The re-calibration corrects for the difference between the RC from two years back (based on costs from the CB year, t-4) and the actual costs in the RC year (t-2). This difference for the RC from two years back is distributed among the companies, and may be either positive or negative. It is distributed based on the companies' RAB.

- Cost norms prior to re-calibration. Here we use the cost norms after calibration, which we calculated in stage 3.
    - `lrt_TOTAL.cost.y.cb`: total costs inflicted in the CB year.
    - `lrt_TOTAL.RC.pre.recal`: the sum of all revenue caps prior to re-calibration.
    - `lrt_TOTAL.Cost.pre.recal`: the sum of all costs before re-calibration. These are costs from the CB year, which are inflated into RC year cost levels.
    - `lrt_RAB`: each company's sum of self-funded RAB for all grid levels.
    - `lrt_TOTRAB`: the sum of self-funded RAB for all companies and for all grid levels.
    - `lrt_recal.fact1`: the re-calibration factor 1. This is the difference between total revenue cap in the RC year and total costs in the CB year, inflated into RC year costs, divided by the total RAB. The difference between the RC and the costs are later distributed among the companies based on their share of total RAB.
    - `lrt_recal.norm`: the re-calibration norm. This is the difference between the total RC and the total costs from the CB year, inflates into RC year cost levels.
    - `lrt_recal.TOT`: the total sum for re-calibration, excluding interest rates. This is the difference between the estimated costs for the CB year (in the RC calculation two years prior) and the actual costs in the CB year.
    - `lrt_recal.fact2`: the re-calibration factor 2. This is the total sum for re-calibration (`lrt.recal.TOT`) including interest rates, divided by the total regulatory asset base. The difference between the estimated and actual costs for the CB year is later distributed among the companies based on their share of total RAB.
    - `lrt_recal.TOT.int`: the total sum for re-calibration (`lrt.recal.TOT`), including interest rates.

- Cost norms after re-calibration. Variables that applies to all companies are stored in the data frame `RevCap`, and the calculation of totals are stored as single parameters.
    - `lrt_cn.pos.recal`: the companies' cost norms after re-calibration. Here we adjust the cost norms after calibration in stage 3. The adjustment multiplies each company's total RAB with the sum of the re-calibration factors 1 and 2. This amount is then divided by rho, which is 0,6, and results in the amount each company's cost norm is to be adjusted for.
    - `lrt.RC.pos.recal`: the companies' revenue caps after calibration. Actual costs are given a weight of 1-rho (40 %) and cost norms are given a weight of rho (60 %).
    - `lrt_TOTAL.RC.pos.recal`: the total revenue caps for all companies after re-calibration
    - `ltr_EBIT.pos.recal`: EBIT after re-calibration
    - `lrt_RET.pos.recal`: "return" after re-calibration

4.5 Functions

`bacon`

This is a multivariate outlier test which detects outliers in our data. Outliers are observations of companies that are substantially different from the other companies. By including outliers in regressions, we may get unreliable results.[13]

`z.est`

This function helps us deal with the problem of too much correlation between the environmental variables. We use principal component analysis to convert correlated environmental variables into new Geo variables.

As inputs in this function, we need a matrix with the correlated environmental variables (`geovar.in`) and a matrix with restricted observations (`restricted.obs`).

We perform the following calculations in this function:

- The cost norm for each company:
    - `geovar.in`: a matrix providing the data for the principal component analysis. It includes the correlated environmental factors.
    - `pca`: the command `prcomp` performs a principal component analysis on the matrix, `geovar.in`. The variables are scaled to have unit variance before the analysis takes place (`scale. = TRUE`).
    - `geovar.ut`: this command predicts a single Geo variable by performing PCA on the restricted observations, which are the correlated environmental variables.

`Zvar1`

This function does the second stage regression, and returns the coefficients for the environmental factors (z-variables).

As inputs in this function we need a vector of input values (`x`), a matrix with values of environmental variables (`z`), the companies' bootstrapped efficiency scores (`eff`), a matrix of reference weights (`lambda`), the companies' IDs (`id`) and the ID's of the companies that are not included in the regression (`id.out`).

We perform the following calculations in this function:

- The cost norm for each company:
    - `x.norm`: each company's average costs (`x`, input value) are multiplied by the reference weights (the lambdas) for each of its peer units, and then added together. This returns a vector with the total cost norm for each company, equal to the costs of each company's target unit.
- The cost norm contribution of each peer unit:

[13] Billor, N., Hadi, A. S., and Velleman , P. F. (2000). BACON: Blocked Adaptive Computationally-Efficient Outlier Nominators; Computational Statistics and Data Analysis 34, 279–298.

  - `x.norm.contrib`: calculates the cost norm contribution of each peer unit. The sum of each company's cost norm contributions will equal the company's cost norm. The reference weights are multiplied by the diagonal input values (a matrix with values only on the diagonal, and zero elsewise).
- The weight for each peer unit:
  - `w.ref`: The weight for each peer unit, which is its norm cost contribution share. We divide each peer unit's norm cost contribution by each company's total norm cost. This returns a matrix with all reference weights.
- Differences in environmental factors:
  - `z.diff`: calculates the difference in environmental factors (z-values) between the company and its target unit. We use the differences as explanatory variables in the coming regression.
- Remove companies that cannot set the technology:
  - `tech`: this defines the companies that can set the technology. We use the command `setdiff(id, id.out)`, which includes the companies in `id` and exclude those in `id.out`. This operation includes all companies except those in the `sep.eval` group.
  - `z.diff`: this makes sure that the z differences are only included for companies in `tech`.
- Outlier test:
  - `outlier.X`: this is a list containing the ID's of outliers. We run the bacon outlier test to check if any of the companies should be defined as outliers, based on their DEA score and their z-values.
  - `id.out`: the IDs of the companies defined as outliers are put into this vector. Outliers are excluded from the following regression.
- The second stage regression:
  - `res.regr.NVE`: this performs the second stage regression with bootstrapped efficiency scores as the dependent variable and differences in environmental factors as independent variables. Only companies that we allow to set the technology are included in the regression.
  - `coeff`: here we store the coefficients for the z-variables from the second stage regression. The columns are named after the z-variables.

`Zvar2`

This function adjusts the DEA scores from stage one for environmental factors.

As inputs in this function we need a vector of input values (`x`), a matrix with values of environmental variables (`z`), the companies' DEA scores from stage 1 (`eff`), a matrix of reference weights (`lambda`), the companies' IDs (`id`) and the coefficients from the second stage regression in `Zvar1` (`coeff`). The function does many of the same calculations as the `Zvar1` function, but it is necessary since slightly different values for the lambdas are used.

We perform the following calculations in this function:

- Adjusting DEA scores:

- `eff.corr`: here we store the adjusted efficiency scores. We multiply the difference in each z-variable between the company and its target unit (`z.diff`) by the corresponding coefficient (`coeff`), and subtract this value from the initial DEA score.
- `res`: these are the results, which includes a list of the corrected efficiency scores (`eff.corr`) and the differences in z-values (`z.diff`).

`NVE_cal`

This function is used stage 3 of the RC model, and returns the companies' calibrated cost norms.

As inputs in this function, we need the companies' efficiency scores after stage 2 (`eff`), the inflated cost bases for the RC year (`cost_base`) and the regulatory assets bases (`RAB`).

We perform the following calculations in this function:

- The companies' cost norms:
  - `cost_norm`: calculating each company's cost norm by multiplying the efficiency score with the cost base.
- The indusrty's total cost base:
  - `tot.cost_base`: calculates the sum of the cost bases for all relevant companies.
- The industry's total cost norm:
  - `tot.cost_norm`: calculates the sum of the cost norms for all relevant companies.
- The industry's total regulatory asset base:
  - `tot.RAB`: calculates the total sum of the regulatory asset base for all relevant companies.
- The cost norms after calibration:
  - `cost_norm.calRAB`: calculates the cost norms after calibration. It takes each company's cost norm and adds (or deducts) some of the difference between the total cost base and the total cost norm. The share of this difference is decided by each company's share of the total regulatory asset base.
- The change in cost norms for each company:
  - `cost_norm_supp`: calculates the change in cost norms for each company. This is each company's share of the difference between the total cost base and the total cost norm. The share of this difference is decided by each company's share of the total regulatory asset base.
- The companies' efficiency scores after calibration:
  - `eff.cal`: calculates each company's efficiency score after calibration. This is done by dividing the new cost base (`cost_norm.calRAB`) by the total cost norm.
- The industry's average efficiency scores:
  - `ind.av.eff`: calculates the average efficiency scores. It divides the total cost norm by the total cost base. After stage 3, the average efficiency score should be 100 %.