Microeconometrics - e-Notes: Practice guide using R

Jaime MONTANA

Updated the 2019-10-07

Contents

Chapter 1

Introduction

1.1 General Info

1.1.1 Contact information

- **Webpage:** [Under construction]
- **E-mail:** jaimem.montana@gmail.com
- **Slack:** Follow this link

Feel free to email at any time to ask questions about the methods covered in class, although I will prioritize communications over the Slack channel. In this way everyone can benefit from others' questions and answers. If anyone knows how to solve a problem, debug or fix the code in the Slack forum s/he can help.

There are some rules for making questions on the procedures. Before asking, it is **mandatory** that you consult the documentation of the function/package; also try to search the answer in a public forum (i.e. Stack Overflow). If after that you still have troubles, post the question taking the following into consideration:

- Be clear and concise, so everyone can understand you
- Be as specific as possible, being clear and straightforward
- Include sufficient information: your goal, the code, the data, everething in order to reproduce the error

Also, you can ask questions about the interpretation of the results, the theory behind, and the like.

1.2 Objective

This class notes are an interactive e-material for the Microeconometrics course in the master APE in Paris School of Economics. The aim of this notes is to provide an e-learning material to apply the theorical concepts of the class. The notes are in **open review**: comments, corrections or contributions that you can make to this part of the course and the material provided are more than welcome.

This part of the course does not cover the theory, and assumes you already had it covered and understood. We will depart from the theory with direct application of the methods.

1.3 Prerequisites

Please install R and Rstudio in your laptop. Here is a video to install R and Rstudio in windows and mac. If you have questions or you could not manage to install it, bring your laptop next session. I will help out for the installation.

- windows-os: For Windows click this link
- mac-os: For Mac-OS click this link

Why **R**? R is a **free** software programming language and a software environment for statistical computing and graphics. The R language is widely used among statisticians, economist, in finance and academics circles.

- R is a **free** software, easy to install and runs in multiple OS.
- A lot of documentation and forums. Excellent documentation on packages.
- Very active community which allow to use other people codes and projects.
- **Great** visualization tools thanks to *ggplot* and *plotly* packages.
- If you understand the logic behind R you will get into every statistical software very easily.
- Everything seems hard at the beginning. Just try and ask.

A prior knowledge on the use of R is required. We don‘t have much time to cover the basics. For an introduction to R you can check the following material:

- Introduction to Econometrics with R, Chapter 1 - 6, by Christoph Hanck et al.
- Introduction to Econometrics with R by Florian Oswald, Jean-Marc Robin and Vincent Viers

1.4 Course structure

- We will have only 3 sessions (2 hours each)
- Bring your laptop with R installed on it
- The material for each session will be online just before each session. In that way you can follow from the e-notes and do the exercises with me during class
- What to expect from each session:

```
1. Brief explanation on the method (how it works)
2. Replication of a published paper that applies the method (downloading data, cleaning data
3. Discussion on the interpretation of the results
4. Q/A
```

- There will be *suggested exercises*. These are **not mandatory**, but remember that if you want to master something, you need to practice. I will be happy to give some feedback on the suggested exercises if you want.

Chapter 2

Session I - Quantile regression

2.1 Objective

This first class is to introduce your to using R for implementing quantile regressions. Therefore, we are going to:

1. Understand the mathematical procedure behind the QR estmation and its computation (both for the estimates and for the the standard errors). This will help understanding the interpretation of the results obtained when applying the procedure in other contexts.
2. Reproduce a paper's tables and results using quantile regression

   ```
   a) Import the data
   b) Clean the data
   c) Reproduce the summary statistics table
   d) reproduce the regressions tables
   ```

3. Interpret the results
4. Ways to communicate the results (plotting in R)

In the last part of the lecture I will just mention and make reference to other classes of QR estimators so you can investigate more on them;

2.2 Quantile Regression

For a summary on what is the intuition and objective of quantile regression check the article "Quantile Regression" (Koenker and Hallock, 2001).

QR is a method that allows you to analyse the relation between x and y across the y distribution. It is useful when the researcher thinks there are *heterogeneous effects* at different values of the indipendent variable. It is important to remark that the heterogeneity is on the **outcome** y. Also, it is widely used in presence of outliers and extreme events (infinite variance), for OLS is inconsistent in such cases while the median is always defined and consistent. For quantiles other than $\tau = 0.5$ the estimation is robust, too.

From the class we know the relationship between the **definition of the estimator**, the **risk function** used in the optimization (in the case of the lector the *'LAD function'*, when $\tau = 0.5$) and that we need to solve numerically the **optimization program** in order to identify the parameters of interest. Accordingly, we will explain how the algorithm works and we are going to perform the numerical optimization by hand from the simplest case to more complex problems.

2.2.1 Geometric interpretation

From (Koenker and Hallock, 2001):

> Quantiles seem inseparably linked to the operations of ordering and sorting the sample observations that are usually used to define them. So it comes as a mild surprise to observe that we can define **the quantiles** through a simple alternative expedient **as an optimization problem**. Just as we can define the sample mean as the solution to the problem of minimizing a sum of squared residuals, we can define the *median as the solution to the problem of minimizing a sum of absolute residuals*. The symmetry of the piecewise linear absolute value function implies that the minimization of the sum of absolute residuals must equate the number of positive and negative residuals, thus assuring that there are the same number of observations above and below the median. What about the other quantiles? Since the symmetry of the absolute value yields the median, perhaps minimizing a sum of asymmetrically weighted absolute residuals—simply giving differing weights to positive and negative residuals—would yield the quantiles. This is indeed the case.

The slope of the coefficient is dividing the error space in two parts according to the desired proportion. It is important to notice that we are considering the error space, we are referring to the conditioning quantile. The difference with the OLS is then clear since the two processes are not comparable. While OLS might provide causal linkages, this is prevented in QR precisely for this reason.

Let's generate some data to see how the line bisects the error space. Since we are generating random data the first thing is to set a seed so our example is reproducible. Then, we generate variance for our error term (not constant) and set an intercept and define the slope. We set everything in a 'data.frame' objecto to plot it using the package 'ggplot2'(Wickham et al., 2019).

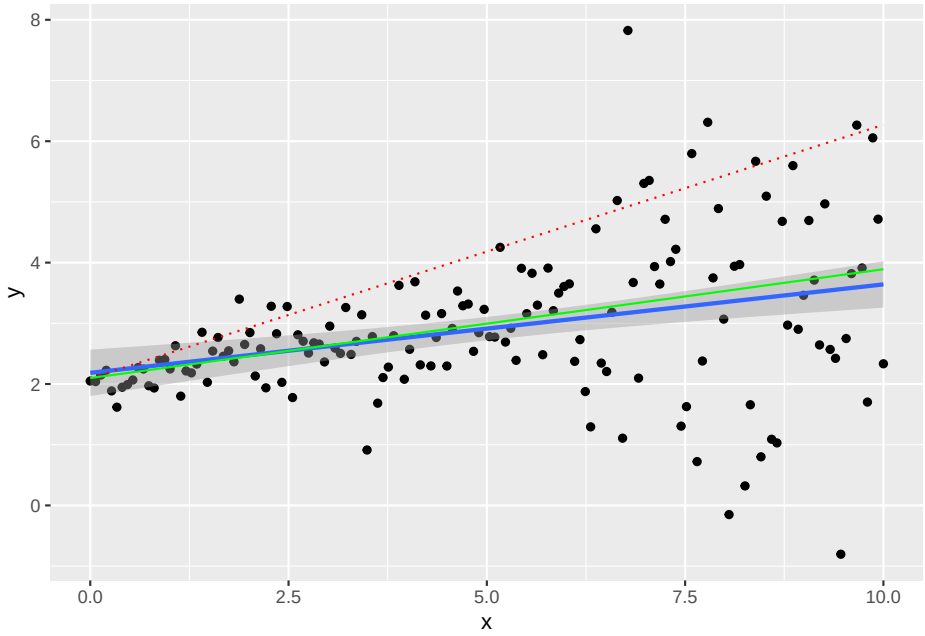
```
set.seed(464)
npoints <- 150
x <- seq(0,10,length.out = npoints)
sigma <- 0.1 + 0.25*sqrt(x) + ifelse(x>6,1,0)
intercept <- 2
slope <- 0.2

error <- rnorm(npoints,mean = 0, sd = sigma)
y <- intercept + slope*x + error
dat <- data.frame(x,y)
```

Let's plot our synthetic data. We are going to plot the line crossing the 90th percentile conditioning on X (dashed red line), the OLS curve (blue line with confidence intervals in grey) and the LAD regression (regression to the median, $\tau = 0.5$)

```
ggplot(dat, aes(x,y)) + geom_point() + geom_smooth(method="lm") +
        geom_quantile(quantiles = 0.9, colour = 'red', linetype="dotted") +
        geom_quantile(quantiles = 0.5, colour = 'green')
```

```
## Smoothing formula not specified. Using: y ~ x
## Smoothing formula not specified. Using: y ~ x
```



We can interpret the causal relationship in quantile regression only under rank invariant condition. This requires that individuals have always the same ranking in the distribution of $Y(X)$ no matter the X. This is difficult to verify or believe in actual applications, but feasible in theory.

Under the rank invariant condition β_τ can be interpreted as the effect on Y of an increase of one unit of X among entities at rank τ in the distribution $Y|X = x$.

2.3 Replication of a paper using quantile regression

We are going to replicate the quantile regression procedure by (Abrevaya, 2002). In the paper the author investigates the impact of different demographic characteristics and maternal behaviour on the weight at birth in the United States in 1992 and 1996. Why is this relevant:

- There is a correlation between health problems after birth for underweight chidren
- There might be a relation with labor market participation and educational attaintment later in life
- There are incentives to create specific programs to deal with underweight children; it is important to understand such behaviours

2.3.0.1 Data:

In order to get the data we access the following link. Here you can download the 'NCHS' Vital Statistics Natality Birth Data', which is the data used in the paper. We are using only the 1992 and 1996 waves. To download the data follow this link for the zip CSV file of 1992. One important thing to do when analizing the data is understand your data before the actual analysis. Before you start you take a minute or two to consider:

- What is the data?
- Where does it come from? What is the universe, the population and what is your sample.
- What is the shape, format of your data? Do I have access to a data dictionary?

In this case many of this questions are available in the documentation that is provided [at the following link] (https://www.nber.org/natality/1992/natl1992.pdf) detailing every variable in the dataset. From the documentation we can see the kind of information (a glimpse of the amount of variables) and the data counts ($4'069'428$ observations). An indicator of the size of the data is the size of the file: the zip file is *156 Mb* and the uncompressed version of the CSV *2.07 GB*! Even if this does not seem much, consider that all this data is stored in the cache of your RAM memory, and it can easily slow down even recent machines. For this reason it is better to read in only the columns that we are interested in. This requires reading the manual and a prior inspection of a subset of the data, which allows you to know the structure, column types and other properties. The most efficient function to open plain text files is *'fread'* from the *'data.table'* package (Dowle and Srinivasan, 2019). First let's investigate the data. The first thing to do is to load the required libraries int the current session:

- *'data.table'* to open the data
- *'quantreg'* to perform the quantile regression estimation
- *'stargazer'* to export the results in latex
- *'dplyr'* to manipulate the data to create the summary statistics

```
library(data.table)
library(quantreg)
library(stargazer)
library(dplyr)
library(kableExtra)
```

```
path_source <- "YOUR PATH GOES HERE"
# for macOS and linux: use / in your path to data
# for Win: rembember to use \\ instead of /
```

```
data <- fread(path_source, nrows = c(100))
head(data[,c(35:41)])
```

```
##    mage12 mage8 ormoth orracem mraceimp mrace mrace3
## 1:      9     4      0       7       NA     2      3
## 2:      9     4      0       7       NA     2      3
## 3:      8     3      0       7       NA     2      3
## 4:      8     3      0       6       NA     1      1
## 5:      5     2      0       6       NA     1      1
## 6:      9     4      0       6       NA     1      1
```

```
type_of_data <- data %>% summarise_all(typeof)
type_of_data[35:41]
```

```
##    mage12   mage8  ormoth orracem mraceimp   mrace  mrace3
## 1 integer integer integer integer  logical integer integer
```

Now that we have had a look at the content of the database, let's import the data and clean it. To import only the relevant variables of the paper we create a list containing all the relevant variables for the estimation. I used the codebook to construct this list. Then we use this list within 'fread()' to import solely the desired columns.

```
desired <- c("birmon","mrace3","dmage","dbirwt","dplural","stnatexp",
             "mraceimp","dmarimp","mageimp","cseximp","dmar","meduc6",
             "wtgain", "mpre5", "tobacco","cigar", "csex", "plurimp", "restatus")

data <- fread(path_source, select = desired)
```

Following the indications of the paper:

> To cut down he sample size, we have decided to use only births occurring in June […] There is no evidence that suggest that the June sampledifffers in any meaningful way to the full sample. The sample was further limited to singleton births and mothers who were either white or black, between ages 18 and 45, and residents of the United States. Observations for which there was missing information on any relevant variable were also dropped. Unfortunately, all births occurring in California, Indiana, New York, and South Dakota had to be dropped from the sample since these states either did not asked a question about smoking during pregnancy or did not ask it in a form compatible with NHCS standards…

We need to apply the following filters:

- Remove all obs. in months different than the sixth (June)
- Remove all non white or non black mothers
- Remove all obs. which age is not in [18,45] group
- Remove the non stated weights
- Remove all the obs. from California, New York, Indiana and South Dakota

```
data <- data[which(data$restatus!=4),]
data <- data[which(data$birmon==6),]
data <- data[which(data$mrace3!=2),]
data <- data[which(data$dmage>17),]
data <- data[which(data$dmage<46),]
data <- data[which(data$dbirwt!=9999),]
data <- data[which(data$dplural==1),]
data <- data[which(!(data$stnatexp %in% c("05","33","34","15","43"))),]
```

Moreover, we remove the missing observations. One particularity of many of the variables of the database is that the missing values are coded, meaning that are not representes by 'NA' but by a code that changes in each variable. We are going to remove also the missing from those variables:

```
data <- data[which(!(data$plurimp %in% c(1))),]
data <- data[which(!(data$mraceimp %in% c(1))),]
data <- data[which(!(data$dmarimp %in% c(1))),]
data <- data[which(!(data$mageimp %in% c(1))),]
data <- data[which(!(data$cseximp %in% c(1))),]
toKeep <- c("dbirwt", "mrace3","dmar","dmage","meduc6", "wtgain", "mpre5", "tobacco","cigar", "cs
data <- as.data.frame(data)
data <- data[,toKeep]
data <- data[which(data$dbirwt!=9999),]
data <- data[which(data$wtgain!=99),]
data <- data[which(data$dmage!=99),]
data <- data[which(data$meduc6!=6),]
data <- data[which(data$mpre5!=5),]
data <- data[which(data$tobacco!=9),]
data <- data[which(data$cigar!=99),]
data <- data[which(data$wtgain !=99),]
```

For the categorical variables (i.e. education of the mother), we create dummy variables. We keep the contrast (levels of reference) according to the paper. In this way we will be able to compare the results.

```
data$black <- ifelse(data$mrace3==3,1,0)
data$married <- ifelse(data$dmar==1,1,0)
data$agesq <- (data$dmage)^2
data$hsgrad <- ifelse(data$meduc6==3,1,0)
data$somecoll <- ifelse(data$meduc6==4,1,0)
data$collgrad <- ifelse(data$meduc6==5,1,0)
data$natal2 <- ifelse(data$mpre5==2,1,0)
data$natal3 <- ifelse(data$mpre5==3,1,0)
data$novisit <- ifelse(data$mpre5==4,1,0)
data$nosmoke <- ifelse(data$tobacco==2,1,0)
data$boy <- ifelse(data$csex==1,1,0)
```

We also relabel variables to match those in the paper:

```
finalVars <- c("dbirwt","black", "married", "dmage", "agesq", "hsgrad", "somecoll", "c
data <- data[, finalVars]
names(data) <- c("birwt","black", "married", "age", "agesq", "hsgrad", "somecoll", "co
```

Now let's make a summary table. We are going to use the `dplyr` package (Wickham et al., 2018) for the data transformation and the `stargazer` package (Hlavac, 2018) to nicely export the results. We construct the columns for all selected variables. To store the results it is convenient to save tables in (LaTeX) format, so you can use them directly form that folder into your paper, or copy the result from them.

```
desc_stats <- round(as.data.frame(cbind(t(data %>% summarise_all(mean)),t(data %>% summ
names(desc_stats) <- c("Mean", "Standard Deviation")
```

```
stargazer::stargazer(desc_stats,
type=ifelse(knitr::is_latex_output(),"latex","html"),
label=knitr::opts_current$get("label"),
title="Summary Statistics 1992", summary = FALSE, out="./images/summary_table.tex")
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu % Date and time: Mon, Oct 07, 2019 - 18:10:06

Now let's calculate the quantile regression:

How does the command rq() work? An easy way to access the help file of a command, just put a question mark before it in the console, i.e. if you want to collect information on the *mean()* function you would type `?mean`. For our quantile regression, we are going to use the function `rq()` from the 'quantreg' package.

From the help file we can see that the principal inputs of the function are 'formula' (the relationship to evaluate), the 'tau' (the vector of quantiles), and

Table 2.1: Summary Statistics 1992

	Mean	Standard Deviation
birwt	3,388.683	571.511
black	0.163	0.370
married	0.752	0.432
age	26.960	5.434
agesq	756.389	303.877
hsgrad	0.394	0.489
somecoll	0.232	0.422
collgrad	0.211	0.408
wtgain	30.760	12.245
natal2	0.158	0.365
natal3	0.026	0.161
novisit	0.009	0.096
nosmoke	0.830	0.376
cigar	2.139	5.737
boy	0.513	0.500

the 'data', which is a dataframe containing the information. Regarding the data it requires a specific type of object, a *'data.frame'*, and also specifies that if we have factors among our variables, it is important to provide a vector with the contrast levels. We do not have to provide it since we already constructed our dummies for such purpose. We are only missing the formula and the quantile vector.

For the moment we are going to construct the formula:

```
Y <- "birwt"
X <- paste(names(data[,-1]),collapse = " + ")
formula_qr <- as.formula(paste(Y, " ~ ", X))
formula_qr
```

```
## birwt ~ black + married + age + agesq + hsgrad + somecoll + collgrad +
##     wtgain + natal2 + natal3 + novisit + nosmoke + cigar + boy
```

And then we construct the vector of quantiles:

```
quantiles_table <- c(0.1,0.25,0.5,0.75,0.9)
```

As in the main slides, if the number of observations is not large enough, we can use the simplex method, but given that we have more than a hundred thousand observations we are going to use the *'Frish-Newton' interior point*

method. We specify this option in the command options including the `method = "fn"` statement. After the calculation we will have an object of class 'rq' or 'rqs', depending on the number of quantiles specified; this might be relevant since some commands might behave differently if operated over this object. For example, the command `summary`, that is often used to get the summary statistics of a 'data.frame', when applied to a 'rq' or 'rqs' object returns the summary of the fit of the QR.

```
quant_reg_res1992 <- rq(formula_qr,tau = quantiles_table, data = data, method = 'fn')
qr_results <- summary.rqs(quant_reg_res1992)
qr_results
```

```
##
## Call: rq(formula = formula_qr, tau = quantiles_table, data = data,
##     method = "fn")
##
## tau: [1] 0.1
##
## Coefficients:
##             Value      Std. Error t value    Pr(>|t|)
## (Intercept) 1556.90509   58.45879   26.63252    0.00000
## black       -253.65558    7.91665  -32.04078    0.00000
## married       73.32313    6.55406   11.18744    0.00000
## age           45.17747    4.27348   10.57159    0.00000
## agesq         -0.78109    0.07630  -10.23749    0.00000
## hsgrad        28.57972    7.31354    3.90778    0.00009
## somecoll      49.49438    8.22498    6.01757    0.00000
## collgrad      82.64542    8.79916    9.39242    0.00000
## wtgain        11.77551    0.16692   70.54735    0.00000
## natal2         6.17644    6.55496    0.94225    0.34606
## natal3        39.43374   15.11504    2.60891    0.00908
## novisit     -388.58389   44.08937   -8.81355    0.00000
## nosmoke      170.75013   11.15951   15.30087    0.00000
## cigar         -3.80301    0.63954   -5.94644    0.00000
## boy           87.76342    4.50293   19.49028    0.00000
##
## Call: rq(formula = formula_qr, tau = quantiles_table, data = data,
##     method = "fn")
##
## tau: [1] 0.25
##
## Coefficients:
##             Value      Std. Error t value    Pr(>|t|)
## (Intercept) 2020.48850   37.98519   53.19147    0.00000
## black       -216.79384    4.64082  -46.71456    0.00000
```

```
## married       55.18092    4.23478   13.03042    0.00000
## age           36.72168    2.75656   13.32156    0.00000
## agesq         -0.59142    0.04893  -12.08786    0.00000
## hsgrad        25.07794    4.76923    5.25828    0.00000
## somecoll      40.54889    5.23706    7.74268    0.00000
## collgrad      57.60057    5.76602    9.98967    0.00000
## wtgain         9.90517    0.11880   83.37620    0.00000
## natal2         2.83300    4.17057    0.67928    0.49696
## natal3        12.67925    9.60040    1.32070    0.18660
## novisit     -196.45154   24.28562   -8.08921    0.00000
## nosmoke      171.83979    7.21094   23.83042    0.00000
## cigar         -3.51950    0.46625   -7.54859    0.00000
## boy          109.56824    2.93365   37.34874    0.00000
##
## Call: rq(formula = formula_qr, tau = quantiles_table, data = data,
##     method = "fn")
##
## tau: [1] 0.5
##
## Coefficients:
##             Value      Std. Error t value    Pr(>|t|)
## (Intercept) 2377.88346   31.58611   75.28256    0.00000
## black       -199.07093    3.82874  -51.99386    0.00000
## married       50.56335    3.57758   14.13338    0.00000
## age           34.63588    2.29095   15.11860    0.00000
## agesq         -0.52963    0.04064  -13.03315    0.00000
## hsgrad        17.54254    3.82369    4.58786    0.00000
## somecoll      31.69835    4.44505    7.13116    0.00000
## collgrad      36.83945    4.81609    7.64925    0.00000
## wtgain         9.13414    0.10254   89.08206    0.00000
## natal2        -4.47675    3.67308   -1.21880    0.22292
## natal3         3.96735    7.23390    0.54844    0.58339
## novisit     -147.23008   15.30005   -9.62285    0.00000
## nosmoke      158.82091    6.10852   25.99989    0.00000
## cigar         -3.90210    0.39009  -10.00299    0.00000
## boy          129.12756    2.55257   50.58719    0.00000
##
## Call: rq(formula = formula_qr, tau = quantiles_table, data = data,
##     method = "fn")
##
## tau: [1] 0.75
##
## Coefficients:
##             Value      Std. Error t value    Pr(>|t|)
## (Intercept) 2715.22798   34.34314   79.06174    0.00000
## black       -192.40481    4.26347  -45.12869    0.00000
```

```
## married       42.45607    4.08114   10.40300    0.00000
## age           31.90923    2.50037   12.76181    0.00000
## agesq         -0.43958    0.04427   -9.92928    0.00000
## hsgrad        15.02372    4.38866    3.42330    0.00062
## somecoll      26.97497    4.98902    5.40687    0.00000
## collgrad      16.26961    5.42208    3.00062    0.00269
## wtgain         8.83829    0.11772   75.08147    0.00000
## natal2        -0.54374    4.07482   -0.13344    0.89385
## natal3        -6.23893    8.86964   -0.70340    0.48181
## novisit     -126.64150   16.13168   -7.85049    0.00000
## nosmoke      153.22724    6.65373   23.02876    0.00000
## cigar         -4.46120    0.40831  -10.92596    0.00000
## boy          142.40192    2.86197   49.75669    0.00000
##
## Call: rq(formula = formula_qr, tau = quantiles_table, data = data,
##     method = "fn")
##
## tau: [1] 0.9
##
## Coefficients:
##             Value      Std. Error t value    Pr(>|t|)
## (Intercept) 2967.68273   47.25148   62.80613    0.00000
## black       -182.08652    5.68366  -32.03682    0.00000
## married       38.55994    5.42803    7.10386    0.00000
## age           33.78764    3.45080    9.79126    0.00000
## agesq         -0.43555    0.06158   -7.07260    0.00000
## hsgrad        13.35546    5.86831    2.27586    0.02286
## somecoll      18.63983    6.67219    2.79366    0.00521
## collgrad      -4.66343    7.47727   -0.62368    0.53284
## wtgain         8.57592    0.15653   54.78798    0.00000
## natal2         3.79637    5.63320    0.67393    0.50036
## natal3       -25.49765   10.73838   -2.37444    0.01758
## novisit     -101.65891   17.28236   -5.88223    0.00000
## nosmoke      150.21941    9.44534   15.90408    0.00000
## cigar         -5.16788    0.60668   -8.51835    0.00000
## boy          153.58224    3.83777   40.01857    0.00000
```

One important consideration is the kind of errors that the procedure is calculating when running the summary.

Nevertheless this kind of results are not easy to handle and manage, and is desireable to have a summary table like the one of the paper or to summarize the results in just one table. We have seen already that is possible to save the results in LaTeX, now we are going to save them in TXT format, which might be usefull in many cases. To this end, we select the first and second column of the results. If in this case the list contains only five items, is still acceptable

to do it line by line. If an operation has more elements and is used often it is better to write a function to save time and avoid copypaste mistakes.

```
tab_res <- as.data.frame(cbind(qr_results[[1]]$coefficients[,1],
                 qr_results[[2]]$coefficients[,1],
                 qr_results[[3]]$coefficients[,1],
                 qr_results[[4]]$coefficients[,1],
                 qr_results[[5]]$coefficients[,1]))
names(tab_res) <- paste0("Tau_",quantiles_table,"_beta")

tab_ES <- as.data.frame(cbind(qr_results[[1]]$coefficients[,2],
                 qr_results[[2]]$coefficients[,2],
                 qr_results[[3]]$coefficients[,2],
                 qr_results[[4]]$coefficients[,2],
                 qr_results[[5]]$coefficients[,2]))

names(tab_ES) <- paste0("Tau_",quantiles_table,"_SE")

results <- as.data.frame(cbind(tab_res,tab_ES))
results <- results[,sort(names(results))]
results <- results[-1,]
```

```
stargazer::stargazer(results, type='text', out="./images/qr_res1992.tex", summary = FALSE)
```

```
##
## ============================================================================================
##          Tau_0.1_beta Tau_0.1_SE Tau_0.25_beta Tau_0.25_SE Tau_0.5_beta Tau_0.5_SE Tau_0.75_be
## --------------------------------------------------------------------------------------------
## black      -253.656     7.917      -216.794       4.641      -199.071     3.829      -192.405
## married     73.323      6.554       55.181        4.235       50.563      3.578       42.456
## age         45.177      4.273       36.722        2.757       34.636      2.291       31.909
## agesq       -0.781      0.076       -0.591        0.049       -0.530      0.041       -0.440
## hsgrad      28.580      7.314       25.078        4.769       17.543      3.824       15.024
## somecoll    49.494      8.225       40.549        5.237       31.698      4.445       26.975
## collgrad    82.645      8.799       57.601        5.766       36.839      4.816       16.270
## wtgain      11.776      0.167        9.905        0.119        9.134      0.103        8.838
## natal2      6.176       6.555        2.833        4.171       -4.477      3.673       -0.544
## natal3      39.434      15.115       12.679        9.600        3.967      7.234       -6.239
## novisit    -388.584     44.089      -196.452      24.286      -147.230     15.300      -126.642
## nosmoke    170.750      11.160       171.840       7.211       158.821     6.109       153.227
## cigar       -3.803      0.640        -3.519        0.466       -3.902      0.390       -4.461
## boy         87.763      4.503        109.568       2.934       129.128     2.553       142.402
## --------------------------------------------------------------------------------------------
```

As aforementioned in the case of a vector of 20 or 50 quantiles it is better to use a **user written function**. We will use this opportunity to remember how

to define user written functions (even if in this example is not necessary). An example of a function is presented to highlight the important parts:

- Define the name
- Define in parenteses the inputs and parameters of the function - remember that default values and ordering are important
- Define in brackets the procedure of the function
- Return the output, results of the operation

```
table_rq_beta_sd <- function(qr_obj){
        # The function creates a summary table from the results of the command rq()

        len <- length(qr_obj)

        res_beta <- c()
        for (i in 1:len) {
                res <- qr_results[[i]]$coefficients[,1]
                res_beta <- cbind(res_beta,res)
        }

        res_se <- c()
        for (i in 1:len) {
                resse <- qr_results[[i]]$coefficients[,2]
                res_se <- cbind(res_se,resse)
        }

        tau <- c()
        for (i in 1:len) {
                tau <- c(tau,qr_results[[i]]$tau)
        }

        res_beta <- as.data.frame(res_beta)
        names(res_beta) <- paste0("Quant_",tau,"_beta")
        res_se <- as.data.frame(res_se)
        names(res_se) <- paste0("Quant_",tau,"_se")


        results <- cbind.data.frame(res_beta,res_se)
        results <- results[, order(names(results))]
        return(results)
}
```

Now we just have to call the function and introduce the object we created before:

```
table_rq_beta_sd(qr_results)
```

```
##             Quant_0.1_beta Quant_0.1_se Quant_0.25_beta Quant_0.25_se
## (Intercept)   1556.9050909  58.45879291    2020.4884992   37.98519460
## black         -253.6555790   7.91664698    -216.7938389    4.64081903
## married         73.3231343   6.55406020      55.1809157    4.23477730
## age             45.1774681   4.27347783      36.7216751    2.75655948
## agesq           -0.7810851   0.07629657      -0.5914186    0.04892666
## hsgrad          28.5797213   7.31354472      25.0779441    4.76923272
## somecoll        49.4943765   8.22498108      40.5488870    5.23705765
## collgrad        82.6454195   8.79916500      57.6005742    5.76601552
## wtgain          11.7755114   0.16691643       9.9051687    0.11880091
## natal2           6.1764400   6.55495922       2.8329988    4.17057029
## natal3          39.4337385  15.11503777      12.6792476    9.60040130
## novisit       -388.5838921  44.08936522    -196.4515370   24.28562273
## nosmoke        170.7501323  11.15950768     171.8397939    7.21094247
## cigar           -3.8030072   0.63954401      -3.5194985    0.46624584
## boy             87.7634154   4.50293347     109.5682422    2.93365299
##             Quant_0.5_beta Quant_0.5_se Quant_0.75_beta Quant_0.75_se
## (Intercept)   2377.8834596  31.58611134    2715.2279805   34.34313590
## black         -199.0709271   3.82873901    -192.4048070    4.26346977
## married         50.5633497   3.57758349      42.4560697    4.08113674
## age             34.6358786   2.29094506      31.9092314    2.50036832
## agesq           -0.5296273   0.04063692      -0.4395786    0.04427096
## hsgrad          17.5425396   3.82368815      15.0237215    4.38866496
## somecoll        31.6983508   4.44505051      26.9749700    4.98902204
## collgrad        36.8394529   4.81608945      16.2696096    5.42208252
## wtgain           9.1341375   0.10253622       8.8382909    0.11771601
## natal2          -4.4767476   3.67307972      -0.5437404    4.07482199
## natal3           3.9673501   7.23390304      -6.2389299    8.86964396
## novisit       -147.2300835  15.30004647    -126.6415003   16.13167721
## nosmoke        158.8209086   6.10852283     153.2272351    6.65373468
## cigar           -3.9020968   0.39009317      -4.4612046    0.40831223
## boy            129.1275630   2.55257411     142.4019151    2.86196534
##             Quant_0.9_beta Quant_0.9_se
## (Intercept)   2967.6827348  47.25148330
## black         -182.0865220   5.68366323
## married         38.5599376   5.42802626
## age             33.7876428   3.45079678
## agesq           -0.4355504   0.06158277
## hsgrad          13.3554587   5.86831170
## somecoll        18.6398317   6.67219287
## collgrad        -4.6634288   7.47727096
## wtgain           8.5759164   0.15652917
## natal2           3.7963741   5.63320458
```

```
## natal3         -25.4976502  10.73837729
## novisit       -101.6589089  17.28236339
## nosmoke        150.2194090   9.44533549
## cigar           -5.1678773   0.60667596
## boy            153.5822439   3.83777441
```

If instead we want to reproduce the table of the paper, we need to compute each of the QR in a separate object and calculate the OLS. Given that the QR regression results table has a different variable order than before, we use for comparison. We also redefine the formula to preserve such order.

```
order_qr <- c("birwt","black", "married", "boy", "nosmoke", "cigar", "age", "agesq", "
```

```
data <- data[,order_qr]
Y <- "birwt"
X <- paste(names(data[,-1]),collapse = " + ")
formula_qr <- as.formula(paste(Y, " ~ ", X))
formula_qr
```

```
## birwt ~ black + married + boy + nosmoke + cigar + age + agesq +
##     hsgrad + somecoll + collgrad + wtgain + natal2 + natal3 +
##     novisit
```

```
p10 <- rq(formula_qr,tau = c(0.1), data = data, method = 'fn')
p25 <- rq(formula_qr,tau = c(0.25), data = data, method = 'fn')
p50 <- rq(formula_qr,tau = c(0.5), data = data, method = 'fn')
p75 <- rq(formula_qr,tau = c(0.75), data = data, method = 'fn')
p90 <- rq(formula_qr,tau = c(0.9), data = data, method = 'fn')
ols <- lm(formula_qr, data = data)
```

Finally, we put the results in a paper format using LaTeX syntax and 'stargazer' functionality. The table presented shows the results for the QR estimation.

```
stargazer(p10,p25,p50,p75,p90,ols, title = "Quantile Regression Results",
          type=ifelse(knitr::is_latex_output(),"latex","html"), out ="./images/qr_rep_
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu % Date and time: Mon, Oct 07, 2019 - 18:11:30

2.3.1 Quantile Regression visualization

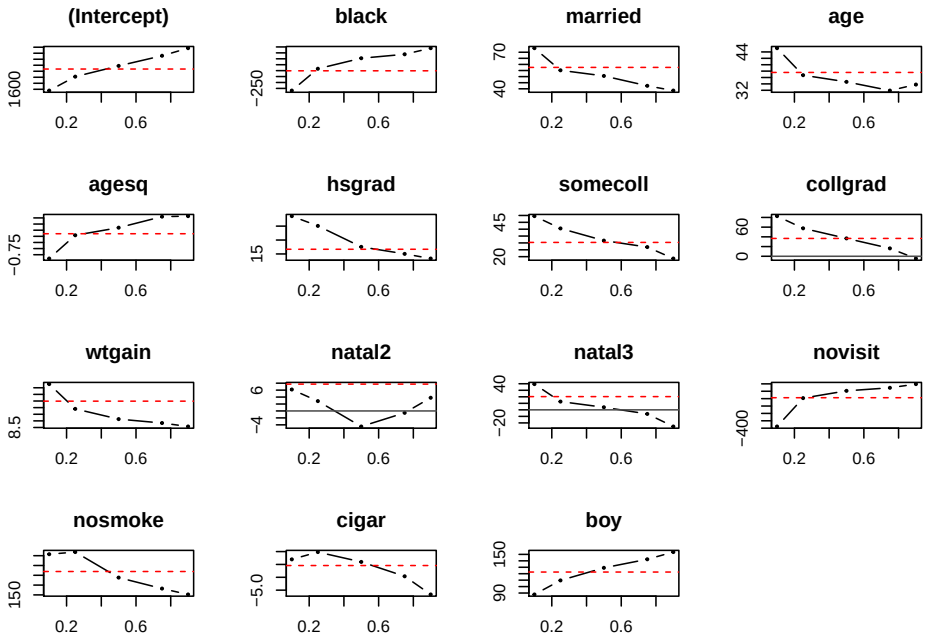
One of the most important ways to visualize and communicate the results from QR is plots. The 'quantreg' package has its own functionality. You will obtain

Table 2.2: Quantile Regression Results

	Dependent variable:					
	birwt					
			quantile regression			
	(1)	(2)	(3)	(4)	(5)	
black	−253.656***	−216.794***	−199.071***	−192.405***	−182.087***	−2
	(7.917)	(4.641)	(3.829)	(4.263)	(5.684)	
married	73.323***	55.181***	50.563***	42.456***	38.560***	5
	(6.554)	(4.235)	(3.578)	(4.081)	(5.428)	
boy	87.763***	109.568***	129.128***	142.402***	153.582***	12
	(4.503)	(2.934)	(2.553)	(2.862)	(3.838)	
nosmoke	170.750***	171.840***	158.821***	153.227***	150.219***	16
	(11.160)	(7.211)	(6.109)	(6.654)	(9.445)	
cigar	−3.803***	−3.519***	−3.902***	−4.461***	−5.168***	−
	(0.640)	(0.466)	(0.390)	(0.408)	(0.607)	
age	45.177***	36.722***	34.636***	31.909***	33.788***	3
	(4.273)	(2.757)	(2.291)	(2.500)	(3.451)	
agesq	−0.781***	−0.591***	−0.530***	−0.440***	−0.436***	−
	(0.076)	(0.049)	(0.041)	(0.044)	(0.062)	
hsgrad	28.580***	25.078***	17.543***	15.024***	13.355**	1
	(7.314)	(4.769)	(3.824)	(4.389)	(5.868)	
somecoll	49.494***	40.549***	31.698***	26.975***	18.640***	3
	(8.225)	(5.237)	(4.445)	(4.989)	(6.672)	
collgrad	82.645***	57.601***	36.839***	16.270***	−4.663	3
	(8.799)	(5.766)	(4.816)	(5.422)	(7.477)	
wtgain	11.776***	9.905***	9.134***	8.838***	8.576***	1
	(0.167)	(0.119)	(0.103)	(0.118)	(0.157)	
natal2	6.176	2.833	−4.477	−0.544	3.796	
	(6.555)	(4.171)	(3.673)	(4.075)	(5.633)	
natal3	39.434***	12.679	3.967	−6.239	−25.498**	2
	(15.115)	(9.600)	(7.234)	(8.870)	(10.738)	
novisit	−388.584***	−196.452***	−147.230***	−126.642***	−101.659***	−1
	(44.089)	(24.286)	(15.300)	(16.132)	(17.282)	(
Constant	1,556.905***	2,020.488***	2,377.883***	2,715.228***	2,967.683***	2,2
	(58.459)	(37.985)	(31.586)	(34.343)	(47.251)	(
Observations	199,181	199,181	199,181	199,181	199,181	
R^2						

graphs with or without confidence depending on the object you feed it: without CI if using the qr object before the summary, and with CI if it is fed the summary of a QR object. To plot we use the command `plot()`. In the first case we obtain all graphs for the previous estimation. A similar graph is presented in (Koenker and Hallock, 2001), in which the Authors the method and apply it to 1997 data (at this point you can reproduce the plots by yourself).

```
plot(quant_reg_res1992)
```



If instead we want to inspect the effect we need more points, so more quantiles. We construct 19 observations and used the CI from the bootstrap option (as stated on the paper). In this case we only show the second independent variable (black dummy).

```
time_0 <- Sys.time()
reg_exp <- rq(formula_qr,tau = seq(0.05,0.95,by = 0.05), data = data, method = 'fn')
time_1 <- Sys.time()

paste0("Computing the quantiles took ",time_1-time_0)
```

```
## [1] "Computing the quantiles took 30.9130508899689"
```

```
sum_reg <- summary.rqs(reg_exp, method = 'boot')
```

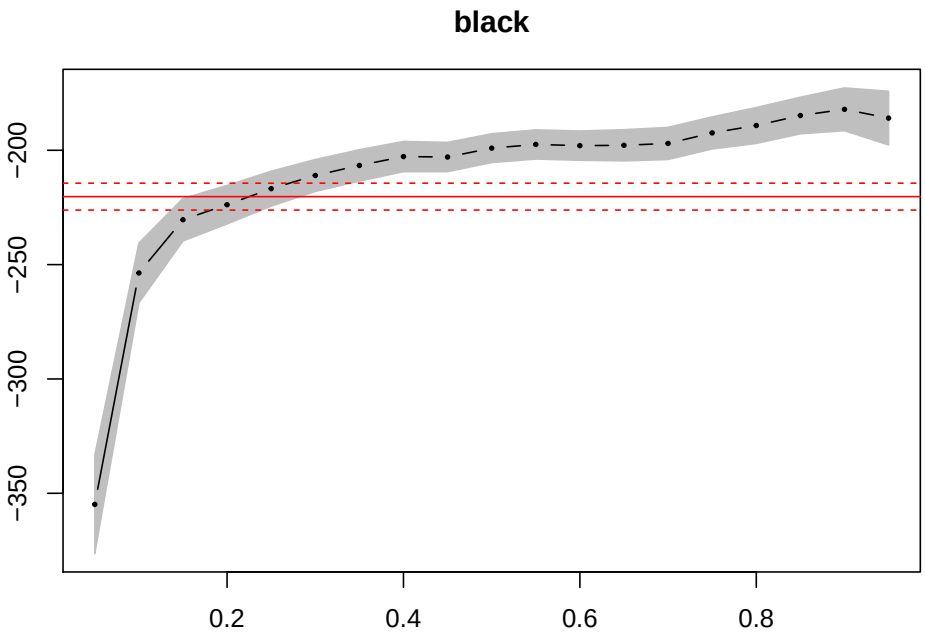
```
## Warning in summary.rq(xi, U = U, ...): 98 non-positive fis
```

```
time_2 <- Sys.time()

paste0("Computing the errors took ",time_2-time_1)
```

```
## [1] "Computing the errors took 1.12076198259989"
```

```
plot(sum_reg, parm = c(2))
```



The plot shows the estimates values, the confidence intervals and the OLS estimated value, so we can compare if the QR offers additional insights.

2.4 Exercises

2.4.1 Proposed exercise 1

- Compute the quantile regression for the year 1996 to complete the descriptive statistics of the paper's table. Are they similar to 1992 results? What is equal? What is different?
- Compute the quantile regression for the same set of variables but for a recent year (after year 2000). Does the outcome change? If yes, what changes and how would you interpret it?

2.4.2 Proposed exercise 2

- Compute the quantile regression for the year 1992 and 1996 including a variable of your interest. Does the result change significantly? Why do you consider it relevant for the analysis? How do you interpret the results obtained?

2.5 More on quantiles

In this section I will only list other implementations of quantile regression that might be usefull for your future research. The list is presentend without any particular order. If you have any suggestions, please let me know:

- Decomposition methods using quantiles (Machado and Mata, 2005)
- Un-conditional quantiles regression (Firpo et al., 2009)
- Decomposition methods using un-conditional quantile methods (Firpo et al., 2018)
- Quantile estimation with non linear effects
- Parallel quantile estimation
- Quantile regression for time series (CAViaR)
- Quantile regression for Spatial Data (package McSpatial)
- Quantile regression for panel data, which is under development since it does not exist (yet) a consistent estimator (package 'rqpd' and Ivan Canay's package)

2.6 References

This material was possible thanks to the slides of David MARGOLIS (in PSE resources), the slides of Arthur CHARPENTIER, and the slides of of the course of Xavier D'HAULTFOEUILLE (ENSAE).

Chapter 3

Session II - Maximum Likelihood Estimation (MLE)

3.1 Introduction

In this second session of the microeconometrics tutorial we are going to implement Maximum Likelihood Estimation in R. The essential steps are:

1. Understand the intuition behind Maximum likelihood estimation. We awill replicate a Poisson regression table using MLE. We will both write our own custom function and a built-in one.
2. Propose a model and derive its likelihood function. This part is not going to be very deep in the explanation of the model, derivation and assumptions. The aim of this sessions are on the estimation (computation) and not in the model per se. I will however refer the sources if you want to have a deeper look at the model. Given that the last session we did something on health economics, this time we change topic and will focus on labour economics. As anecdote, when I first saw this I was very impressed! I hope you are impressed too after today session!

3. Use the Current Population Survey (CPS) and understand how to handle and manage this particular dataset.

   ```
   + Import the data
   + Use the specified columns
   + Clean data
   ```

4. Estimate the structural parameters of the proposed model (both for the estimates and the standard errors, obtaind via the *delta method*).

3.2 Maximum likelihood estimator

For the theory please refer to the slides of the course (class 2). Here we just provide a brief definition and the intuition to the method application. What is Maximum likelihood estimation (MLE)? MLE is an estimation method in which we obtain the *parameters* of our model under an **assumed** *statistical model* and the available *data*, such that our sample is the most probable.

- Given a statistical model (ie, an economic model with suitable stochastic features), select the parameters that make the observed data most probable. In this way, we are doing inference in the population that generated our data and the DGP behind. We can formulate **any** model and we will obtain a result; the only restriction for the formulation is that it has probability 0.
- Even if it is intuitive, rely on the **assumptions** (model, statistical model, DGP), but not in the **validity**.
- Validity of the models?

  > "A model is a deliberate abstraction from reality. One model can be better than another, on one or several dimensions, but none are correct. They help us focus on the small set of phenomena in which we are interested, and/or have data regarding. When correctly developed and explained, it should be clear what set of phenomena are being excluded from consideration, and, at the end of the analysis, it is desirable to say how the omission of other relevant phenomena could have affected the results attained.
  >
  > An advantage of a structured approach to empirical analysis is that it should be immediately clear what factors have been considered exogenous and which endogenous, the functional form assumptions made, etc." (C. Flinn, lecture notes)

To understand how MLE works we will use two examples today: a Poisson regression and a structural estimation.

3.2.1 Poisson regression

In this section we are going to replicate a paper by Daniel Treisman published in AER 2016. It applies Poisson regression to the number of millioners in a set of countries, conditional on some country characteristics (Treisman, 2016).

- **What does the paper says?**

The main idea of the paper is *provide* a robust *model* to predict the number of rich in a given country, given an economic environment specific to said country.

In comparing data and predictions the Author finds that, regardless of the model specification, Russia reports the highest number of anomalies (underpredictions).

- **Why a Poisson regression?**

In Treisman's paper the dependent variable — the number of billionaires y_i in country i — is modelled as a function of GDP per capita, population size, and years membership in GATT and WTO. He also presents 4 alternative specifications.

> "... since the dependent variable is a count, Poisson rather than OLS regression is appropriate."

3.2.1.1 Estimation

You can acces the data and the paper in the provided links. Download and unzip the information. The file also contains a companion STATA code to reproduce the tables in the paper. Unzip the information and load the dataset in R using the `haven` library (Wickham and Miller, 2018):

```
data_mil <- read_dta("YOUR PATH GOES HERE")
```

To reproduce table 1 from the paper we need to filter the information for 2008, as it is the year considered in the main analisys.

```
data_mil  <- data_mil[data_mil$year == 2008,]
```

Our **goal** is to **estimate a Poisson regression model** and there are built-in functions to do these kind of estimations using a one-line command like `glm(..., family = "poisson")`. Our **goal** instead is to **use Maximum Likelihood estimation to reproduce such parameters** and understand how this works. In order to have a benchmark for comparison let's see how the output of the first proposed model looks like:

```r
summary(glm(formula = numbil0 ~ lngdppc + lnpop + gattwto08, family = poisson(link='log
```

```
##
## Call:
## glm(formula = numbil0 ~ lngdppc + lnpop + gattwto08, family = poisson(link = "log")
##     data = data_mil)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -8.7615  -0.7585  -0.3775  -0.1010   9.0587
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -29.049536   0.638193 -45.518  < 2e-16 ***
## lngdppc       1.083856   0.035064  30.911  < 2e-16 ***
## lnpop         1.171362   0.024156  48.491  < 2e-16 ***
## gattwto08     0.005968   0.001908   3.127  0.00176 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 5942.23  on 196  degrees of freedom
## Residual deviance:  669.95  on 193  degrees of freedom
##   (354 observations deleted due to missingness)
## AIC: 885.08
##
## Number of Fisher Scoring iterations: 5
```

Let's start the construction of the maximum likelihood introducing the Poisson regression model. The starting assumptions are: Poisson regression is a generalized linear model form of regression analysis used to model count data (1), in which the dependent variable has a Poisson distribution (2), and the logarithm of the expected value can be modeled as a linear combination (3).

1. Count data: notice that all the numbers from a Poisson distribution are integers.
2. Poisson pdf is defined as:

$$f(k, \lambda) = Pr(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

   Keep in mind that the **mean** and variance of the Posson distribution are equal to the constant λ.

3. The **mean** can be expressed as a linear combination of the parameters.

$$\mu = E[y] = E[e^y] = E(y|\boldsymbol{X}) = e^{\theta' \boldsymbol{X}} = e^{\theta_0 + \theta_1 x_{i1} + \ldots + \theta_k x_{ik}}$$

Combining condition (2) and (3), we obtain the probability density function:

$$f(y_i, \lambda) = f(y_i, \mu) = \frac{\mu_i^y e^{-\mu}}{y_i!}$$

The joint probability density function is hence equal to:

$$f(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\theta}) = f(y_1, \mu) f(y_2, \mu) \ldots f(y_n, \mu) = \prod_{i=1}^{n} f(y_i, \mu|\boldsymbol{X}, \boldsymbol{\theta})$$

Our likelihood function instead takes as given the data (vector $\boldsymbol{Y}$ and matrix $\boldsymbol{X}$) and calculates the likelihood given a parameter θ.

$$\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}, \boldsymbol{X}) = \prod_{i=1}^{n} f(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{Y}) = \prod_{i=1}^{n} \frac{\mu_i^{y_i} e^{-\mu_i}}{y_i!}$$

Finding the $\hat{\boldsymbol{\theta}}$ that maximizes the likelihood function therefore boils down to estimate the model parameters. Before this step we monotically transform the obective function to feed the algorithm the log-likelihood instead of the function itself. Why? Think about derivatives of sums vs. derivatives of products.

$$\hat{\boldsymbol{\theta}} = \max_{\boldsymbol{\theta}} log(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}, \boldsymbol{X})) = \min_{\boldsymbol{\theta}} -log(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}, \boldsymbol{X}))$$

Taking the logarithm then we have:

$$\begin{aligned} \log(\mathcal{L}(\boldsymbol{\theta}|\boldsymbol{Y}, \boldsymbol{X})) &= \log\left(\prod_{i=1}^{N} \frac{\mu_i^y e^{-\mu}}{y_i!}\right) \\ &= \sum_{i=1}^{N} \log\left(\frac{\mu_i^{y_i} e^{-\mu_i}}{y_i!}\right) \\ &= \sum_{i=1}^{N} y_i \log(\mu_i) - \sum_{i=1}^{N} \log(\mu_i) - \sum_{i=1}^{N} \log(y_i!) \end{aligned} \tag{3.1}$$

Where

$$\mu = e^{\theta' \boldsymbol{X}} = e^{\theta_0 + \theta_1 x_{i1} + \ldots + \theta_k x_{ik}}$$

.

Absent a closed-form solution, we are going to use the R optimizer to maximize the log-likelihood function above. The *optim* function will serve this purpose. According to the help vignette `?optim`, we need:

- A vector of initial values to start the search from.
- A well specified **function** to **minimize**. It must take as *input* **the vector of parameters** and *should* **return a scalar**.
- *Method* of optimization (?).
- Optional: a hessian (boolean). We will use this to parse out the standard errors around the estimated parameters, so it will be useful later on.

In the last installment we introduced the basics of custom functions in R. In this tutorial we just recall as good practice that we are going to differenciate the inputs of the function: the parameters are the inputs that are going to change in the optimization process, while the data for example will be a **static** input. Another static input is the *formula*, a type of R object that will allow to extract the relevant information such as variables name, data columns, and instructs the algorithm on the relationship between variables (dependent or independent, interactions, dependance, etc).

We start by writing the function. We call the function *'LLcalc'*, and define the inputs in parentheses. One useful function that we use in the first line is `model.matrix()`. This function takes a formula and extract from the whole dataset the related matrix of observations including the vector of ones of the intercept, dummies, and interaction terms.

We are going to compute the value of μ and then the value of the individual contribution to the log-likelihood. Then, we just sum and flip sign, as the optimizer minimizes by default. There is one technical detail that we are addressing using the package `Rmpfr`, which allow to store big numbers in 128 bits (as the factorail of 400) (Maechler, 2019).

```
LLcalc <- function(theta, formula, data){
        ### Calculate the log-likelihood of a Poisson regression,
        ### given a vector of parameters (1),
        ### a relationship ()formula (2)
        ### and a dataset (type: dataframe)

        rhs <- model.matrix(formula, data = data)
        colnames(rhs)[1] <- "Intercept"
        Y <- as.matrix(data[rownames(rhs),toString(formula[[2]])])
        # Expected values \mu
        mu <- exp(rhs %*% theta)
```

```
        # Value of the log likelihood
        LL <- Y * log(mu)-mu-as.numeric(log(gamma(as(Y+1,"mpfr"))))

        #print(cbind(colnames(rhs),round(theta,3)))
        return(-sum(LL, na.rm = T))

}
```

To check whether the function works properly we feed it the data, a random θ, and the formula of the first column of table I in the paper.

```
theta_test <- rnorm(4)
formula_1 <- as.formula(numbil0 ~ lngdppc + lnpop + gattwto08)
LLcalc(theta = theta_test, formula = formula_1, data = data_mil)
```

```
## [1] 26030.13
```

Now that we know is working, it is time to set up the optimizer. As discussed before we need the function, some starting parameters, the data and the formula, the static inputs of our function.

```
theta_0 <- c(-30,1,1,0)
dTbp_beta <- optim(par = theta_0, fn = LLcalc, data = data_mil, formula = formula_1, method = "Ne
round(dTbp_beta$par,3)
```

```
## [1] -29.052   1.084   1.171   0.006
```

To carry out the estimation we need to compute the standard errors. As a by-product, we also have the Hessian, this is useful in couple with the theory from the main class:

- The negative of the hessian (provided as a linearization around the optimum of the LL maximization) is equal to the Fisher information matrix.

$$[\mathcal{I}(\theta)]_{i,j} = \mathbf{E}\left[\left(\frac{\partial}{\partial\theta_i}\log f(X|\theta)\right)\left(\frac{\partial}{\partial\theta_j}\log f(X|\theta)\right)|\theta\right] \approx \left[\frac{\partial^2}{\partial\theta_i\partial\theta_j}\log f(X|\theta)|\theta\right]$$

- The Fisher information matrix, when inverted, is equal to the variance covariance matrix. (Formally, Cramer-Rao state that the inverse is the lower bound of the variance if the estimator is unbiased.)

- The variance covariance matrix has in the diagonal the variane for each parameter. Taking square root of it gives the standard errors.
- since we minimize we do not have to flip sign, as the values of the LL are calculated over the negative likelihood function.

Considering these notions we obtain the following standard errors that are equal to the previous regression.

```
fisher_info <- dTbp_beta$hessian
vcov <- solve(fisher_info)
se <- sqrt(diag(vcov))
se
```

```
## [1] 0.638233253 0.035068731 0.024155062 0.001907374
```

To complete the exercise we are going to reproduce the whole table 1 from the paper. We need to set up all of the formulas (models) estimated. The paper provides the robust errors, so we calculate them and format with stargazer (Hlavac, 2018) output.

```
formula_2 <- as.formula(numbil0 ~  lngdppc + lnpop +  gattwto08 +  lnmcap08 +  rintr +
formula_3 <- as.formula(numbil0 ~  lngdppc + lnpop +  gattwto08 +  lnmcap08 +  rintr +
formula_4 <- as.formula(numbil0 ~  lngdppc + lnpop +  gattwto08 +  lnmcap08 +  rintr +

r1 <- glm(formula = formula_1, family = poisson(link='log'), data = data_mil)
r2 <- glm(formula = formula_2, family = poisson(link='log'), data = data_mil)
r3 <- glm(formula = formula_3, family = poisson(link='log'), data = data_mil)
r4 <- glm(formula = formula_4, family = poisson(link='log'), data = data_mil)

se_rob <- list(sqrt(diag(sandwich::vcovHC.default(r1,type = "HC0"))),
               sqrt(diag(sandwich::vcovHC.default(r2,type = "HC0"))),
               sqrt(diag(sandwich::vcovHC.default(r3,type = "HC0"))),
               sqrt(diag(sandwich::vcovHC.default(r4,type = "HC0"))))
```

```
stargazer::stargazer(r1,r2,r3,r4, title = "Table 1 - Poisson regression",
          type=ifelse(knitr::is_latex_output(),"latex","html"), se = se_rob, out = "./
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu % Date and time: Mon, Oct 07, 2019 - 18:13:54

3.3 Example II - Structural estimation

In this section the aim is to estimate the parameters from the likelihood function of a given model and be able to calculate it in the statistical software (in this

Table 3.1: Table 1 - Poisson regression

	Dependent variable:			
	numbil0			
	(1)	(2)	(3)	(4)
lngdppc	1.084***	0.717***	0.737***	0.963***
	(0.138)	(0.244)	(0.233)	(0.243)
lnpop	1.171***	0.806***	0.929***	1.153***
	(0.097)	(0.213)	(0.195)	(0.293)
gattwto08	0.006	0.007	0.004	0.0003
	(0.007)	(0.006)	(0.006)	(0.004)
lnmcap08		0.399**	0.286*	0.114
		(0.172)	(0.167)	(0.237)
rintr		−0.010	−0.009	−0.007
		(0.010)	(0.010)	(0.009)
topint08		−0.051***	−0.058***	−0.060***
		(0.011)	(0.012)	(0.015)
nrrents			−0.005	0.013
			(0.010)	(0.013)
roflaw			0.203	0.342
			(0.372)	(0.283)
fullprivproc				−0.002*
				(0.001)
Constant	−29.050***	−19.444***	−20.858***	−25.951***
	(2.578)	(4.820)	(4.255)	(6.240)
Observations	197	131	131	113
Log Likelihood	−438.540	−259.731	−256.024	−179.661
Akaike Inf. Crit.	885.079	533.461	530.049	379.322
Note:				*p<0.1; **p<0.05; ***p<0.01

case, R). We are going to estimate the structural parameters of a very simple search model (Flinn and Heckman, 1982), following Flinn and Heckman 1982. This tutorial is not devoted to understanding such labor model, so we are going to describe the model and its assumptions; from that point we are going to derive the ML function. Then we are going to feed that function to the computer as in the previous case and maximize it to find the parameters of the model.

3.3.0.1 The model:

Agents are well behaved and maximize the income they receive. When they are unemployed, they face a searching cost c. Upon paying such cost, offers from a **Poisson process** will arrive. The arrival offer rate is denoted by λ, and its probability by unit of time is also λ. When they meet a **wage** is proposed from a **log-normal distribution** and the individuals can refuse to form the match or 'seal the deal'. We also assume that the *length of the employment spells* follow a **exponential distribution** and that there is a **constant risk of loosing the job** with period probability η. Time is discounted at a rate ρ.

Such model is characterized by two Bellman equations. The first equation is the **value of being employed** [eq. (3.2) of the paper]:

$$V_e(w) = \frac{1}{\rho}[w + (1-\eta)V_e + \eta V_u]$$

Reorganizing terms boils down to:

$$V_e(w) = \frac{1}{\rho+\eta}[w + \eta V_u]$$

The second equation is the **value of being unemployed**:

$$V_u = \frac{1}{\rho}[c + (1-\lambda)V_u + \lambda + \mathbf{E}\max\{V_e, V_u\}]$$

Which, after rearranging, is equal to:

$$\rho V_u = -c + \frac{\lambda}{\rho+\eta}\int_{\rho V_u}(w - \rho V_u)f(w)$$

These two equations describe the whole behaviour of the economy under the assumptions of the model. Now let's list all the assumptions that we have until now, since they are going to be usefull for the construction of the likelihood function:

- The minimum accepted wage is equal to $w^* = \rho V_u$. In the paper the minimum accepted wage is not estimated, but instead is taken from the data.

- Arrival rate of offers: λ.

- Termination rate: η

- Distribution of employment duration: $f[t_e] = \eta e^{(-\eta t_e)}$; the average duration is then $\mathbf{E}[t_e] = \eta^{-1}$.

- Rate of leaving the unemployed is equal to: $\lambda(1 - F(w^*))$. We are going to call this h_u

- Distribution of unemployment duration is exponential $f[t_u] = h_u e^{-h_u t_u)}$. The expected value in unemployment then is then equal to $(\lambda(1 - F(w^*)))^{-1} = h_u^{-1}$

- The distribution of accepted wages is equal to: $\frac{f(w)}{1-F(w^*)}$. The term below is to adjust it to a distribution since a proper distribution must integrate to 1 and our is left censored.

- The distribution of employments spells is right-censored. Given that it is negative exponential it coincides with the population.

Considering all this information, we can proceed to calculate the probability to sample an employed individual out of the population as weel as the probability to sample an unemployed individual. Let's start with the latter:

$$P(U) = \frac{\mathbf{E}[t_u]}{\mathbf{E}[t_u] + \mathbf{E}[t_e]} = \frac{h_u^{-1}}{h_u^{-1} + \eta^{-1}} = \frac{\eta}{h_u + \eta}$$

Now let's derive the probability of sampling an employed individual:

$$P(E) = \frac{\mathbf{E}[t_e]}{\mathbf{E}[t_u] + \mathbf{E}[t_e]} = \frac{\eta^{-1}}{h_u^{-1} + \eta^{-1}} = \frac{h_u}{h_u + \eta}$$

Usually, data on the duration of unemployment t_u^o and the wages w^o are observed from census or surveys, hence we can calculate such parameters because now we can define the likelihood:

$$
\begin{aligned}
\mathcal{L} &= \prod_U [P(U) \times f(t_u^o)] \times \prod_E [P(E) \times f(w^o)] = \\
&= \prod_U \left[P(U) \times \lambda(1 - F(w^*)) e^{-\lambda(1-F(w^*))t_u} \right] \times \prod_E \left[P(E) \times \frac{f(w|\mu,\sigma)}{1 - F(w^*)} \right] = \\
&= \prod_U \left[\frac{\eta}{h_u + \eta} \times \lambda(1 - F(w^*)) e^{-\lambda(1-F(w^*))t_u} \right] \times \prod_E \left[\frac{h_u}{h_u + \eta} \times \frac{f(w|\mu,\sigma)}{1 - F(w^*)} \right] = \\
&= \prod_U \left[\frac{\eta}{\lambda(1 - F(w^*)) + \eta} \times \lambda(1 - F(w^*)) e^{-\lambda(1-F(w^*))t_u} \right] \times \\
&\prod_E \left[\frac{\lambda(1 - F(w^*))}{\lambda(1 - F(w^*)) + \eta} \times \frac{f(w|\mu,\sigma)}{1 - F(w^*)} \right]
\end{aligned}
\tag{3.2}
$$

After taking logs and rearranging we obtain the following **log-likelihood**:

$$\log \mathcal{L} = N \log(h_u) - h_u \sum t_u + N_u \log(\eta) + \sum f(w|\mu,\sigma) - N_e \log(1 - F(w^*)) - N_e \log(h_u + \eta)$$

We just need to feed this function to the optimizer with some data and we can obtain the parameters of the model. Before carrying out the liklihood estimation, we have a look at the data and try to parse out some useful extra information.

3.3.0.2 The Data

To estimate the model we just need two vectors of data: duration of unemployment and hourly wages. To make a real example we are going to use the Current population survey (CSP), but most of the household surveys contain these kind of data (eg, colombian GEIH).

First we go to the page of the CPS, and learn about the nature of the data. We can also download the historical monthly data from the NBER webpage. For this exercise we are going to use the January 2019 data, which can be obtained following this link. Download and unzip the dataset in your working directory. Take also a moment to check the required variables, looking the documentation for the data. The open the dataset using the functionality of the `reader` package (Wickham et al., 2017).

```
data <- read_csv("jan19pub.dat", col_names = FALSE, trim_ws = FALSE)
```

As you see, the data only contain 1 vector of text – in jargon, observations are in long format. Each row is a long string of text. If you took the time to

read the documentation you will see that you can find next to each variable the description and the location of the specific information in such long string. To extract information about unemployment duration and employment wages we need the following parts:

GROUP OF INTEREST	VARIABLE	LENGTH	DESCRIPTION	LO
EMPLOYED	HRMIS	2	MONTH-IN-SAMPLE	63
EMPLOYED	PEERNPER	2	PERIODICITY	50
EMPLOYED	PEERNHRO	2	USUAL HOURS	52
EMPLOYED	PRERNWA	8	WEEKLY EARNINGS RECODE	52
EMPLOYED	PRERNHLY	4	RECODE FOR HOURLY RATE	52
EMPLOYED	PEHRUSL1	2	HOW MANY HOURS PER WEEK DO YOU WORK	21
EMPLOYED	PTWK	1	WEEKLY EARNINGS - TOP CODE	53
UNEMPLOYED	PEMLR	2	MONTHLY LABOR FORCE RECODE	18
UNEMPLOYED	RUNEDUR	3	DURATION OF UNEMPLOYMENT FOR	40
ALL	GESTFIPS	2	FEDERAL INFORMATION STATE	93

After identifying this information we can filter the data. To do so we create a database for the employed and a database for the unemployed, since the identification requires different information and variables. Let's begin with the employed: we create a *'data.frame'* containing the relevant colums and we extract the information by the position in the string.

```
employed <- data.frame(matrix(NA, nrow = nrow(data), ncol = 8))

colnames(employed) <- c(  "HRMIS",
                          "PEERNPER",
                          "PEERNHRO",
                          "PRERNWA",
                          "PRERNHLY",
                          "PTWK",
                          "PEHRUSL1",
                          "GESTFIPS")

employed$HRMIS          <- apply(data,1, function(x) as.numeric(substr(toString(x),63,64)))
employed$PEERNPER       <- apply(data,1, function(x) as.numeric(substr(toString(x),502,503)))
employed$PEERNHRO       <- apply(data,1, function(x) as.numeric(substr(toString(x),525,526)))
employed$PRERNWA        <- apply(data,1, function(x) as.numeric(substr(toString(x),527,534)))
employed$PRERNHLY       <- apply(data,1, function(x) as.numeric(substr(toString(x),520,523)))
employed$PTWK           <- apply(data,1, function(x) as.numeric(substr(toString(x),535,535)))
employed$PEHRUSL1       <- apply(data,1, function(x) as.numeric(substr(toString(x),218,219)))
employed$GESTFIPS       <- apply(data,1, function(x) substr(toString(x),93,94))
```

Reading the documentation we identify that the invalid cases are coded 0 or

negative values $-1, -2, \ldots$. We reclassify this coded infromation as `NA`, a *not available* or *missing value*. We are going to keep the observations that are from the outgoing rotations (have earnings information). After that we are going to filter the valid hourly wages and convert the weekkly wages to hours using the number of hours. We are going to keep only the people that have this information. We are also going to take the right number of decimal for the variables that specify it. We are going to keep only the observations that are above the legal federal minimum wage of 7.5 USD, and we are going to trim the data at the percentile 99.9%.

```
employed[employed<=0] <- NA

employed <- employed[which(employed$HRMIS %in% c(4,8)),]
employed <- employed[which(employed$PEERNPER > 0),]
employed <- employed[-which(employed$PRERNHLY == 9999),]
employed$PRERNHLY <- employed$PRERNHLY/100
employed <- employed[-which(employed$PTWK == 1),]
employed$PRERNWA <- employed$PRERNWA/100

employed$wages <- ifelse(employed$PRERNHLY > 0 & !is.na(employed$PRERNHLY),
                         employed$PRERNHLY,
                         ifelse(!is.na(employed$PRERNWA) & !is.na(employed$PEHRUSL1),
                                employed$PRERNWA/employed$PEHRUSL1,
                                NA)
                         )

employed <- employed[which(!is.na(employed$wages)),]
employed <- employed[which(employed$wages >= 7.25),]
employed <- employed[which(employed$wages <= quantile(employed$wages, 0.999)),]

tokeep_e <- c("GESTFIPS", "wages")
employed <- employed[,tokeep_e]
employed$duration_U <- 0
```

Now we apply the same selection to the unemployed: We collect the infromation in each variable following the position. We take the people for which the labor employment status is unemployment and that have information on the duration. Then we convert the information to monthly data.

```
unemployed <- data.frame(matrix(NA, nrow = nrow(data), ncol = 3))
colnames(unemployed) <- c("PEMLR",
                          "RUNEDUR",
                          "GESTFIPS")

unemployed$PEMLR          <- apply(data,1, function(x) as.numeric(substr(toString(x),18
```

```
unemployed$RUNEDUR        <- apply(data,1, function(x) as.numeric(substr(toString(x),407,409)))
unemployed$GESTFIPS       <- apply(data,1, function(x) substr(toString(x),93,94))

unemployed <- unemployed[which(unemployed$PEMLR %in% c(3,4)),]
unemployed <- unemployed[which(unemployed$RUNEDUR > 0),]
unemployed$duration_U <- unemployed$RUNEDUR/4.333
unemployed <- unemployed[,c("GESTFIPS","duration_U")]
unemployed$wages <- 0
unemployed <- unemployed[,c("GESTFIPS","wages", "duration_U")]
```

As a final step, we merge the two dataset:

```
data <- rbind(employed,unemployed)
```

We are going to select and calculate the MLE using only one state, as minimum wages laws vary locally. For this exercise we are going to use the information on Nevada ($GESTFIPS = 32$)

```
data_sub <- data[which(data$GESTFIPS=="32"),]
```

3.3.0.3 Estimation

In order to estimate the log likelihood we are going to code two auxiliary funcions that appear all the time in the procedure. The first one is the log-normal density function:

$$LN\left(x;\mu,\sigma\right)=\phi\left(\frac{\log(x)-\mu}{\sigma}\right)\left(\frac{1}{x\sigma}\right)$$

Where ϕ is the probability density function of the $N(0,1)$ distribution.

```
lognorm <- function(x,mu,sigma){
        res <- dnorm((log(x)-mu)/(sigma))*(1/(sigma*x))
        return(res)
}
```

The second function is the *survival function*, which is the probability to be over the minimum accepted wage for a given distribution. We define the survival as $(1-F(w^{*}))$. The cumulative distribution function of the log-normal is equal to:

$$\Phi\left(\frac{\log(x)-\mu}{\sigma}\right)$$

Where Φ is the cumulative distribution function of the standard normal distribution.

```
surv <- function(val,mu,sigma){
        res <- 1 - pnorm((log(val)-mu)/(sigma))
        return(res)
}
```

There are other built-in functions already developed that are suitable for this kind of problems. One xample is the `mle2` function from the *'bbmle'* package (Bolker and Team, 2017). This is useful since the function deals with standard errors and provides other information that might be useful. There are some difference between the `optim()` method already covered and the `mle2()` function. This latter function requires:

- Function to calculate negative log-likelihood
- Starting values for the optimizer
- The optimizer used
- The data

Now we are going to code the log-likelihood function we recovered using the same procedure as in the previous example:

First we are going to code each of the parameters and the data as inputs in the function. Then we are going to code the likelihood using the definition derived earlier.

```
LLfnct_mle <- function(alambda,aeta,amu,asigma, data){


        lambda = exp(alambda)
        eta    = exp(aeta)
        mu     = amu
        sigma  = exp(asigma)

        w_star <- min(data[which(data$wages > 0),]$wages)


        h_u <- lambda * surv(w_star, mu, sigma)
        n <- nrow(data)
        n_u <-  nrow(data[which(data$duration_U > 0),])
        n_e <- (n-n_u)
```

```
        LL <-   n * log(h_u) -
                n_e * log(surv(w_star, mu, sigma)) -
                h_u * sum(data$duration_U) +
                n_u * log(eta) +
                sum(log(lognorm(data[which(data$wages>0),]$wages,mu,sigma))) -
                n_e * log(eta + h_u)

        #print(cbind(c("lambda = ","eta = ","mu = ","sigma = "),c(lambda,eta,mu,sigma)))
        return(-LL)

}
```

After that we just have to set up the maximizer and feed it the function:

```
m0 <- mle2(LLfnct_mle,
           start = list(alambda =-1,
                        aeta = -1,
                        amu = 3,
                        asigma = -1),
           data = list(data = data_sub),
           optimizer = "nlminb")

m0@coef
```

```
##    alambda       aeta        amu     asigma
## -0.4541099 -1.8175680  2.7146674 -0.4361533
```

All the coeficients need to be transformed to recover and present the results. Before that we are going to calculate the standard errors using the delta method and the information from the hessian. At the end what we are doing is just rescaling the errors.

```
lambda <- exp(m0@coef["alambda"])
eta <- exp(m0@coef["aeta"])
mu <- m0@coef["amu"]
sigma <- exp(m0@coef["asigma"])


fisher_info <- m0@details$hessian
vcov_mle <- solve(fisher_info)
prop_sigma<-sqrt(c(lambda^2,eta^2,mu^2,sigma^2) * diag(vcov_mle))
names(prop_sigma) <- c("lambda","eta","mu","sigma")
```

```
stargazer(cbind("parameter" = names(prop_sigma), "theta" =c(lambda,eta,mu,sigma), prop_
          title = "Model parameters MLE - Nevada (Jan 2019)",
          type=ifelse(knitr::is_latex_output(),"latex","html"), out = "./images/SE_Neva
```

% Table created by stargazer v.5.2.2 by Marek Hlavac, Harvard University. E-mail: hlavac at fas.harvard.edu % Date and time: Mon, Oct 07, 2019 - 18:14:29

Table 3.3: Model parameters MLE - Nevada (Jan 2019)

	parameter	theta	prop_sigma
alambda	lambda	0.635012931441372	0.0900528154150777
aeta	eta	0.16242027479781	0.0379917018868457
amu	mu	2.71466743713899	0.270516421768113
asigma	sigma	0.646518617780906	0.0678137201168286

3.4 References

This material was possible thanks to the slides of David MARGOLIS (in PSE resources), the course of C. FLinn at CCA 2017, and QuantEcon MLE.

Bibliography

Abrevaya, J. (2002). The effects of demographics and maternal behavior on the distribution of birth outcomes. In *Economic applications of quantile regression*, pages 247–257. Springer.

Bolker, B. and Team, R. D. C. (2017). *bbmle: Tools for General Maximum Likelihood Estimation*. R package version 1.0.20.

Dowle, M. and Srinivasan, A. (2019). *data.table: Extension of 'data.frame'*. R package version 1.12.2.

Firpo, S., Fortin, N., and Lemieux, T. (2018). Decomposing wage distributions using recentered influence function regressions. *Econometrics*, 6(2):28.

Firpo, S., Fortin, N. M., and Lemieux, T. (2009). Unconditional quantile regressions. *Econometrica*, 77(3):953–973.

Flinn, C. and Heckman, J. (1982). New methods for analyzing structural models of labor force dynamics. *Journal of Econometrics*, 18(1):115–168.

Hlavac, M. (2018). *stargazer: Well-Formatted Regression and Summary Statistics Tables*. R package version 5.2.2.

Koenker, R. and Hallock, K. F. (2001). Quantile regression. *Journal of economic perspectives*, 15(4):143–156.

Machado, J. A. and Mata, J. (2005). Counterfactual decomposition of changes in wage distributions using quantile regression. *Journal of applied Econometrics*, 20(4):445–465.

Maechler, M. (2019). *Rmpfr: R MPFR - Multiple Precision Floating-Point Reliable*. R package version 0.7-2.

Treisman, D. (2016). Russia's billionaires. *American Economic Review*, 106(5):236–41.

Wickham, H., Chang, W., Henry, L., Pedersen, T. L., Takahashi, K., Wilke, C., Woo, K., and Yutani, H. (2019). *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.2.1.

Wickham, H., François, R., Henry, L., and Müller, K. (2018). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.6.

Wickham, H., Hester, J., and Francois, R. (2017). *readr: Read Rectangular Text Data*. R package version 1.1.1.

Wickham, H. and Miller, E. (2018). *haven: Import and Export 'SPSS', 'Stata' and 'SAS' Files*. R package version 1.1.2.