R Code Master Thesis

Joffrey Mayer

12/30/2020

Construction of my dataset: Master-Thesis Joffrey Anthony Mayer-Boutter

Note before you start running this code:

Since the code is very long (over 8000 lines of code), I tried to explain every step I do by splitting the code into subsections and guiding the reader with many comments (denoted with hashtags "#"). Furthermore, by using more than 3 hashtags at the beginning and the end of comments // subsections, we can see a downward pointing arrow alongside the line of code in the R-script. By clicking on the downward pointing arrow, it will change its position to a horizontal pointing arrow and the whole code within a section will be masked. Here is an example (you should see a downward pointing arrow starting at line 18 of this R script. By klicking on it, it should then point horizontally and wrap up everything within the 'Test-Section'):

```
###### Test-Section:######
## THIS TEXT SHOULD VANISH // SHOULD BE WRAPPED UP WHEN KLICKING
## ON THE FIRST DOWNWARD-POINTING ARROW IN LINE 18!
######
```

BEFORE running the entire code, it may be useful to "close" every downward pointing arrow in each subsection and wrap the whole code up. This may not be a necessary step. However, since I only had a computer with limiting computational capabilities I need to wrap around all the code for each subsection everytime I open up the 'ddset-construction.R'-file, otherwise some computers may only run at a very, very slow pace.

Lastly, don't forget to change the working directory at line 39 of this R-script in order for the code to run.

My Dataset construction:

```
###############
## Step 1: Preliminary basic steps
###### 1) Load some packages, set working directory and clear the workspace. ######
rm(list = ls()) # clear workspace

setwd("~/Uni/Masterstudium/Masterarbeit/final-code") # set wd

# load some packages:
library(dummies)
library(foreign)
library(stargazer) # important to create tables
library(readstata13) # important to load the ddset
library(haven)
library(plyr)# to use functions like count()
# Note: install.packages("readstata13") first before you can load them!
```

Combine all separated waves into one "big dataset"

In the next step, I will load all the datasets / waves I downloaded from FORSBASE and construct a panel-dataset.

I already received a cleaned version of my dataset (but **without** the code to replicate it). In order to understand how it was constructed and to orient my later data-cleaning, I will look into it first:

```
###############
## Step 2: Combine all separated waves into one "big dataset"
###### 1) Load an already clean ddset:######

### This is an already cleaned ddset I got from my University: load it
data_matched <- read_dta("~/Uni/Masterstudium/Masterarbeit/Data/matched.dta")# with parent_ID & ID_chil

# While working & trying out different things with the above data, I
# realized a mistake in the coding of the 'ysport'-variable:
data_matched$sport_dummy_young <- ifelse(data_matched$ysport >= 1,1,0) #"child, young" // Period "t-1"
xtabs(~ age + sport_dummy_young, data=data_matched) # here you can see that the
# 'ysport'-variable was coded uncorrectly --> look at the children between
# 14 & 16, which make about 20% (see excel file 'summary-statistics')
# of my whole dataset: no individual in this age-range does sport,
# because - according to the individual-questionnaire, wave 15
# (p. 95 / 481) - the "sport in youth"-question is only asked to
# those individuals above the age of 16. The younger ones get all a "-3"
# (inapplicable), since they are not eligible for this question. The researcher
# probably re-coded all the negative values to "0" and got a dataset with
# a wrong 'ysport'-variable. Because I did not check every single variable
# and I was unsure if the 'sport'-variable was handled correctly (since
# I don't have access to their codes, I don't see how some variables were
# constructed and why there are so mnay NA's for the parents), I decided
# to create my own dataset, which turned out to be extremely tedious, but
# - nevertheless - was the best way to get to know the dataset in detail.
xtabs(~ age + ysport, data=data_matched) # we see it also here

### Let's construct a time-series dataset that contains 15 waves, e.g.
# which is 15 years long: from 2004 (= wave 6) to 2018 (= wave 20).
```

Construction of my own panel-dataset:

As we have seen, the variable 'ysport' - which indicates whether an individual did sport in his / her youth - was **coded incorrectly** in the already cleaned version I received. Since this variable will be part of my dependent variable in my discrete choice model, it will be very important to code it correctly myself, otherwise wrong conclusions will be drawn from it. In the next step, I will therefore be integrating each separate wave (= a wave is an equivalent to one particular year) into a standalone panel-dataset.

Let's start with **wave 15**, since it is the *main wave* (because the question whether an individual did sport in his youth is only asked in the year 2013) and represents the **year 2013**:

```
###### 2) Load the uncleaned main wave 15 from FORSBASE (= data-provider):######
data_full <- read_dta("~/Uni/Masterstudium/Masterarbeit/Data/dataset/shp13_p_user.dta")
```

```r
# wave 15 // year 2013 downloaded from FORSBASE (original). This is the
# individual-questionnaire.
hh15 <-  read_dta("~/Uni/Masterstudium/Masterarbeit/Data/dataset/shp13_h_user.dta")
# wave 15 // year 2013 downloaded from FORSBASE (original). This is the
# household-questionnaire.
```

Combining individual- and household datasets:

Concerning the generic structure of the questionnaire, FORSBASE - the data provider - has organized the survey in a way that - depending on a person's age - it lets the subjects choose between **3 types of questionnaires**:

- Fill out an *individual-questionnaire*, that contains all the personal information about a person mentioned in the first paragraph of this subsection. Ideally, this is the type of questionnaire which should be completed by an individual.
- Fill out a *proxy-questionnaire*, which mostly contains some information about work and health-status, but - relative to an individual questionnaire - offers only very little information on the individual-level.
- Fill out a *household-questionnaire*, which gives information on the composition of an individual's household, but does not include any personal-level data that can be used for the analysis.

For the creation of future variables, I will need some variables from *all* of the 3 types of questionnaires. Therefore, I will first merge the seperate household- & indivividual-questionnaire together, since both contain useful variables for my analysis:

```r
###### 3) Combine the hh- & indiv.-questionnaire together for w15: ######

# Let's first concentrate on the main ddset I am going to
# analyze, which is wave 15 (= year 2013):
which(colnames(hh15)=="idhous13") #4
which(colnames(hh15)=="sthhre13") #6
which(colnames(hh15)=="canton13") #9
which(colnames(hh15)=="region13") #10
which(colnames(hh15)=="hhmove13") #11
which(colnames(hh15)=="nbpers13") #19
which(colnames(hh15)=="h13i12") #89
which(colnames(hh15)=="h13i54") #134
which(colnames(hh15)=="h13i57a1") #136
which(colnames(hh15)=="h13i57a2") #137
which(colnames(hh15)=="h13i57a3") #138
which(colnames(hh15)=="h13i57a4") #139
which(colnames(hh15)=="h13i57a5") #140
which(colnames(hh15)=="i13htyg") #147
which(colnames(hh15)=="i13htyn") #148
hh15 <- hh15[c(4,6,9,10:11,19,89,134,136:140,147:148)]
# the object hh15' should have 15 columns // variables.

## Let's merge the two objects 'hh15' & 'data_full' (= wave 15) via the
# ID-hh variable:
ih_datafull <- merge(data_full, hh15, by = "idhous13", all.x = TRUE) # merging
data_full <- ih_datafull
rm(ih_datafull)
# Merged Ddset should have: 513 + (15 - 1) = 527 columns
# --> "-1" because I merged over one variable.
```

Do the same for the other 14 waves:

I will now load all the remaining 14 waves from FORSBASE and do the exact same thing I did for my main wave above, e.g. select the the variables useful for my analysis and re-name them, since the names given by FORSBASE are not intuitive at all:

```
###### 4) Do the same for the other 14 waves (2004-2018): ######

# I will now load all the remaining 14 waves from FORSBASE; select the
# the variables useful for my analysis and re-name them, since the names
# given by FORSBASE are not intuitive at all.

## starting with wave 06 (= year 2004): load it
data_full06 <- read_dta("~/Uni/Masterstudium/Masterarbeit/Data/dataset/shp04_p_user.dta")# wave06 // ye
hh06 <-  read_dta("~/Uni/Masterstudium/Masterarbeit/Data/dataset/shp04_h_user.dta")# wave 06 downloaded

## merging hh- & indiv.-questionnaire together:
hh06 <- hh06[c(which(colnames(hh06)=="idhous04"),
               which(colnames(hh06)=="sthhre04"),
               which(colnames(hh06)=="canton04"),
               which(colnames(hh06)=="region04"),
               which(colnames(hh06)=="hhmove04"),
               which(colnames(hh06)=="nbpers04"),
               which(colnames(hh06)=="h04i12"),
               which(colnames(hh06)=="h04i54"),
               which(colnames(hh06)=="h04i57a1"),
               which(colnames(hh06)=="h04i57a2"),
               which(colnames(hh06)=="h04i57a3"),
               which(colnames(hh06)=="h04i57a4"),
               which(colnames(hh06)=="h04i57a5"),
               which(colnames(hh06)=="i04htyg"),
               which(colnames(hh06)=="i04htyn"))] # should have 15 columns

## merge via the ID-hh variable:
ih_datafull <- merge(data_full06, hh06, by = "idhous04", all.x = TRUE) # merging: 10'884 obs.?
data_full06 <- ih_datafull # before: 426 column // now: 440 columns
rm(ih_datafull)

## variable-selection:
testo06 <- data_full06[,c(which(colnames(data_full06)=="idpers"), #1
                          which(colnames(data_full06)=="status04"),
                          which(colnames(data_full06)=="sex04"),
                          which(colnames(data_full06)=="age04"),
                          which(colnames(data_full06)=="edyear04"),
                          which(colnames(data_full06)=="p04c01"),
                          which(colnames(data_full06)=="p04c44"),
                          which(colnames(data_full06)=="p04a04"),
                          which(colnames(data_full06)=="p04a01"),
                          which(colnames(data_full06)=="wstat04"),#10
                          which(colnames(data_full06)=="p04c02"),
                          which(colnames(data_full06)=="p04c12"),
                          which(colnames(data_full06)=="p04c15"),
                          which(colnames(data_full06)=="p04c45"),
                          which(colnames(data_full06)=="p04c46"),
                          which(colnames(data_full06)=="p04c19a"),
                          which(colnames(data_full06)=="p04c11"),
                          which(colnames(data_full06)=="p04c03"),
                          which(colnames(data_full06)=="civsta04"),
                          which(colnames(data_full06)=="ownkid04"),#20
                          which(colnames(data_full06)=="p04w46"),
                          which(colnames(data_full06)=="p04w77"),
                          which(colnames(data_full06)=="i04ptotg"),
                          which(colnames(data_full06)=="i04ptotn"),
                          which(colnames(data_full06)=="p04l26"),
                          which(colnames(data_full06)=="p04a05"),
                          which(colnames(data_full06)=="p04a06"),
                          which(colnames(data_full06)=="p04f01"),
                          which(colnames(data_full06)=="p04f02"),
                          which(colnames(data_full06)=="idhous04"),## 30 & start hh-var.
                          which(colnames(data_full06)=="sthhre04"),
                          which(colnames(data_full06)=="canton04"),
                          which(colnames(data_full06)=="region04"),
                          which(colnames(data_full06)=="hhmove04"),
                          which(colnames(data_full06)=="nbpers04"),
                          which(colnames(data_full06)=="h04i12"),
                          which(colnames(data_full06)=="h04i54"),
                          which(colnames(data_full06)=="h04i57a1"),
                          which(colnames(data_full06)=="h04i57a2"),
                          which(colnames(data_full06)=="h04i57a3"), # 40
                          which(colnames(data_full06)=="h04i57a4"),
                          which(colnames(data_full06)=="h04i57a5"),
                          which(colnames(data_full06)=="i04htyg"),
                          which(colnames(data_full06)=="i04htyn")
)] # 44 variables in total

## re-name columns:
names(testo06)[6] <- "sah04"
names(testo06)[7] <- "lsat04"
names(testo06)[8] <- "weekly_act04"
names(testo06)[9] <- "act04"
names(testo06)[10] <- "empstat04"
names(testo06)[11] <- "hsat04"
names(testo06)[12] <- "doctor04"
names(testo06)[13] <- "year_doct04"
names(testo06)[14] <- "height04"
names(testo06)[15] <- "weight04"
names(testo06)[16] <- "chronic_lthp04"
names(testo06)[17] <- "hprob_days04"
names(testo06)[18] <- "h_impr04"
names(testo06)[21] <- "CMJ_desire_hrs04"
names(testo06)[22] <- "CMJ_actual_hrsweek04"
names(testo06)[23] <- "ginc_year04"
names(testo06)[24] <- "ninc_year04"
names(testo06)[25] <- "prob_children04"
names(testo06)[26] <- "freetime_sat04"
names(testo06)[27] <- "leisure_sat04"
names(testo06)[28] <- "lalone_sat04"
```

```r
names(testo06)[29] <- "ltogeth_sat04"
names(testo06)[36] <- "car04"
names(testo06)[37] <- "min_inc04"
names(testo06)[38] <- "contrinc_one04"
names(testo06)[39] <- "contrinc_two04"
names(testo06)[40] <- "contrinc_three04"
names(testo06)[41] <- "contrinc_four04"
names(testo06)[42] <- "contrinc_five04"
names(testo06)[43] <- "hh_incg04"
names(testo06)[44] <- "hh_incn04"

### Continue with wave 07: load it
data_full07 <- read_dta("~/Uni/Masterstudium/Masterarbeit/Data/dataset/shp05_p_user.dta")# wave07 // ye
hh07 <-  read_dta("~/Uni/Masterstudium/Masterarbeit/Data/dataset/shp05_h_user.dta")# wave 07 downloaded

## merging hh- & indiv.-questionnaire together:
hh07 <- hh07[c(which(colnames(hh07)=="idhous05"),
               which(colnames(hh07)=="sthhre05"),
               which(colnames(hh07)=="canton05"),
               which(colnames(hh07)=="region05"),
               which(colnames(hh07)=="hhmove05"),
               which(colnames(hh07)=="nbpers05"),
               which(colnames(hh07)=="h05i12"),
               which(colnames(hh07)=="h05i54"),
               which(colnames(hh07)=="h05i57a1"),
               which(colnames(hh07)=="h05i57a2"),
               which(colnames(hh07)=="h05i57a3"),
               which(colnames(hh07)=="h05i57a4"),
               which(colnames(hh07)=="h05i57a5"),
               which(colnames(hh07)=="i05htyg"),
               which(colnames(hh07)=="i05htyn"))] # should have 15 columns
# Pay attention: hh07 has duplicates, which you need to eliminate before
# merging:
hh07 <- hh07[!duplicated(hh07),] # eliminates the duplicates

## merge via the ID-hh variable:
ih_datafull <- merge(data_full07, hh07, by = "idhous05", all.x = TRUE) # merging: 10'884 obs.?
data_full07 <- ih_datafull # before: 429 column // now: 443 columns
rm(ih_datafull)

## variable-selection:
testo07 <- data_full07[,c(which(colnames(data_full07)=="idpers"), #1
                          which(colnames(data_full07)=="status05"),
                          which(colnames(data_full07)=="sex05"),
                          which(colnames(data_full07)=="age05"),
                          which(colnames(data_full07)=="edyear05"),
                          which(colnames(data_full07)=="p05c01"),
                          which(colnames(data_full07)=="p05c44"),
                          which(colnames(data_full07)=="p05a04"),
                          which(colnames(data_full07)=="p05a01"),
                          which(colnames(data_full07)=="wstat05"),#10
                          which(colnames(data_full07)=="p05c02"),
                          which(colnames(data_full07)=="p05c12"),
                          which(colnames(data_full07)=="p05c15"),
                          which(colnames(data_full07)=="p05c45"),
                          which(colnames(data_full07)=="p05c46"),
                          which(colnames(data_full07)=="p05c19a"),
                          which(colnames(data_full07)=="p05c11"),
                          which(colnames(data_full07)=="p05c03"),
                          which(colnames(data_full07)=="civsta05"),
                          which(colnames(data_full07)=="ownkid05"),#20
                          which(colnames(data_full07)=="p05w46"),
                          which(colnames(data_full07)=="p05w77"),
                          which(colnames(data_full07)=="i05ptotg"),
                          which(colnames(data_full07)=="i05ptotn"),
                          which(colnames(data_full07)=="p05l26"),
                          which(colnames(data_full07)=="p05a05"),
                          which(colnames(data_full07)=="p05a06"),
                          which(colnames(data_full07)=="p05f01"),
                          which(colnames(data_full07)=="p05f02"),
                          which(colnames(data_full07)=="idhous05"),## 30 & start hh-var.
                          which(colnames(data_full07)=="sthhre05"),
                          which(colnames(data_full07)=="canton05"),
                          which(colnames(data_full07)=="region05"),
                          which(colnames(data_full07)=="hhmove05"),
                          which(colnames(data_full07)=="nbpers05"),
                          which(colnames(data_full07)=="h05i12"),
                          which(colnames(data_full07)=="h05i54"),
                          which(colnames(data_full07)=="h05i57a1"),
                          which(colnames(data_full07)=="h05i57a2"),
                          which(colnames(data_full07)=="h05i57a3"), # 40
                          which(colnames(data_full07)=="h05i57a4"),
                          which(colnames(data_full07)=="h05i57a5"),
                          which(colnames(data_full07)=="i05htyg"),
                          which(colnames(data_full07)=="i05htyn")
)] # 44 variables in total

# re-name columns:
names(testo07)[6] <- "sah05"
names(testo07)[7] <- "lsat05"
names(testo07)[8] <- "weekly_act05"
names(testo07)[9] <- "act05"
names(testo07)[10] <- "empstat05"
names(testo07)[11] <- "hsat05"
names(testo07)[12] <- "doctor05"
names(testo07)[13] <- "year_doct05"
names(testo07)[14] <- "height05"
names(testo07)[15] <- "weight05"
names(testo07)[16] <- "chronic_lthp05"
names(testo07)[17] <- "hprob_days05"
names(testo07)[18] <- "h_impr05"
names(testo07)[21] <- "CMJ_desire_hrs05"
names(testo07)[22] <- "CMJ_actual_hrsweek05"
names(testo07)[23] <- "ginc_year05"
names(testo07)[24] <- "ninc_year05"
names(testo07)[25] <- "prob_children05"
```

```r
names(testo07)[26] <- "freetime_sat05"
names(testo07)[27] <- "leisure_sat05"
names(testo07)[28] <- "lalone_sat05"
names(testo07)[29] <- "ltogeth_sat05"
names(testo07)[36] <- "car05"
names(testo07)[37] <- "min_inc05"
names(testo07)[38] <- "contrinc_one05"
names(testo07)[39] <- "contrinc_two05"
names(testo07)[40] <- "contrinc_three05"
names(testo07)[41] <- "contrinc_four05"
names(testo07)[42] <- "contrinc_five05"
names(testo07)[43] <- "hh_incg05"
names(testo07)[44] <- "hh_incn05"

### Continue with wave 08: load it

## NOTE: since the code has always the same structure for the remaining waves and in order
## to not make the PDF too long, I will not display the rest of the code here! etc...
```

Merge all separated waves into one:

Next, let's combine all waves into one single, big dataset by using the **Reduce()-function** and do some cleaning on it:

```r
###### 5) Merge all separated waves into one dataset by using Reduce(): ######

# The Reduce()-function will merge all waves together and output a
# new dataset in "wide-format" (e.g. every variable is a column, for
# each year):
dd_ts <- Reduce(function(x,y) merge(x = x, y = y, by = "idpers", all.x= TRUE, all.y=TRUE),
                list(testo06, testo07, testo08, testo09, testo10, testo11,
                     testo12, testo13, testo14, testo,
                     testo16, testo17, testo18, testo19, testo20)) # be
# Careful: Reduce() is case sensitive (write it "Reduce()" and not
# "reduce()").
# Another detail: you need all.y = TRUE, otherwise the function will
# only pass the IDs of wave 04, but not individuals, which will join
# the dataset only later in the panel. For example individual with the
# ID 5102, which joins the wave 05, but I don't have this individual
# available, if I don't include the 'all.y = TRUE', this individual
# will NOT be available even though I start in wave 04, e.g. 1 wave
# earlier!

# not needed anymore:
# rm(list= c("testo", "testo06", "testo07", "testo08", "testo09", "testo10",
#            "testo11", "testo12", "testo13", "testo14", "testo16", "testo17",
#            "testo18", "testo19", "testo20"))

# re-ordering the dataset to have a better overview:
dd_ts <- dd_ts[, order(names(dd_ts))] # sort alphabetically
which(colnames(dd_ts)=="idpers") # idpers should be 1st column. At the moment: column 300
which(colnames(dd_ts)=="status04") # status04 is number 591: important
# to take 'status'-variables in front, since it gives "meta"-infos
```

```r
which(colnames(dd_ts)=="status18") # column 605
which(colnames(dd_ts)=="idhous04") # column 389
which(colnames(dd_ts)=="idhous18") # column 403
dd_ts <- dd_ts[c(404, 591:605, 389:403,
                 1:388, 405:590, 606:674)]
```

Splitting the dataset

At this point, it is to note that, while most of the questions stay the same during each wave of the questionnaire, there are some **variables that were only asked within one particular wave**, or - sometimes - in very irregular survey-intervals. The sport-variables are an example of such non-recurring questions, since they appear only in wave 15 and 17. That is why I split the dataset into 2 parts, since it will allow me to handle the irregular sport-variables differently from the variables, which appear yearly in the survey.

First Split with Non-Sport Variables:

Here, let's split the dataset by eliminating the sport-variables that only appear irregularly:

```r
###### 6) Split the new ddset into 2 parts --> "first split" (only non sport-variables) ######

### Next, I split the new "big ddset" into 2: one part contains only
### the sport-variables, the other part the rest. This step
### will be the "first split" (only non sport-variables).

# Sport-variables are available only for 2 waves (wave 15 & wave 17),
# while the other variables are asked every year in the questionnaire.
# That's why I decided to take the sport-variables in a separate dataset.
# Let's select all sport-variables and put them all on the very front
# of the ddset (in order for the split afterwards):
which(colnames(dd_ts)=="type15")
which(colnames(dd_ts)=="frequency15")
which(colnames(dd_ts)=="duration15")
which(colnames(dd_ts)=="compet15")
which(colnames(dd_ts)=="waythere15")
which(colnames(dd_ts)=="agesport15")
which(colnames(dd_ts)=="ref_per15")
which(colnames(dd_ts)=="type13")
which(colnames(dd_ts)=="frequency13")
which(colnames(dd_ts)=="duration13")
which(colnames(dd_ts)=="compet13")
which(colnames(dd_ts)=="waythere13")
which(colnames(dd_ts)=="agesport13")
which(colnames(dd_ts)=="ref_per13")
which(colnames(dd_ts)=="yref_per13")
which(colnames(dd_ts)=="ytype13")
which(colnames(dd_ts)=="yfrequency13")
which(colnames(dd_ts)=="yduration13")
which(colnames(dd_ts)=="ycompet13")

dd_ts_15y <- dd_ts[c(62:63,155:156,247:248,294:295,558:559,621:624,655:656,672:674,
                     1:61,64:154,157:246,249:293,296:557,560:620,625:654,657:671)]
save(dd_ts_15y, file="dd_ts_15y.Rda")## save dataset: dd_ts == dataset with
# sport & no sport variables in wide-format.
```

```r
load("dd_ts_15y.Rda")
dd_ts <- dd_ts_15y
rm(dd_ts_15y)

# now, all the sport-variables should be the first 19 columns:
names(dd_ts)
dd_ts_nosp <- dd_ts[c(20:674)] # filter out all "non sport"-variables.
# --> this is the "first split".
```

Make a Panel:

Next, let's transform the "first split" (= the part of the "big dataset" with non-sport variables) using use the reshape()-function:

```r
###### 7) Transform "first split from wide- into long-format using reshape() #####

### Next, I will transform the "first split" (= the part of the "big
### dataset" with non-sport variables) from wide-format into
### long-format, e.g. build a panel-dataset with reshape()-function:

# now take all variables in front that DON'T vary over time:
which(colnames(dd_ts_nosp)=="hprob_when")
which(colnames(dd_ts_nosp)=="hprob_phys")
which(colnames(dd_ts_nosp)=="hprob_cause")
which(colnames(dd_ts_nosp)=="nat_1")
which(colnames(dd_ts_nosp)=="nat_2")
which(colnames(dd_ts_nosp)=="nat_3")
which(colnames(dd_ts_nosp)=="hab_ch")
which(colnames(dd_ts_nosp)=="ch_sinbirth")
which(colnames(dd_ts_nosp)=="sib_outhh")
dd_ts_nosp <- dd_ts_nosp[c(1,92,318,379,395:396,487:489,595,
                          2:91,93:317,319:378,380:394,397:486,490:594,596:655)]

## Make my dataset from wide-format into "long-format" (panel-data format)
which(colnames(dd_ts_nosp)=="status04") #depends on which the starting wave is --> status08 if you star
which(colnames(dd_ts_nosp)=="status18")
# now, let's transform the ddset from wide- into "long-format"
# (e.g. every variable is now a time-series):
data.long <- reshape(dd_ts_nosp, direction = "long",
                     varying = list(names(dd_ts_nosp)[11:25], #varying status
                                    names(dd_ts_nosp)[26:40], #variying activity
                                    names(dd_ts_nosp)[41:55], #varying age
                                    names(dd_ts_nosp)[56:70],
                                    names(dd_ts_nosp)[71:85],
                                    names(dd_ts_nosp)[86:100],
                                    names(dd_ts_nosp)[101:115],
                                    names(dd_ts_nosp)[116:130],
                                    names(dd_ts_nosp)[131:145],
                                    names(dd_ts_nosp)[146:160],
                                    names(dd_ts_nosp)[161:175],
                                    names(dd_ts_nosp)[176:190],
                                    names(dd_ts_nosp)[191:205],
                                    names(dd_ts_nosp)[206:220],
                                    names(dd_ts_nosp)[221:235],
                                    names(dd_ts_nosp)[236:250],
                                    names(dd_ts_nosp)[251:265],
                                    names(dd_ts_nosp)[266:280],
                                    names(dd_ts_nosp)[281:295],
                                    names(dd_ts_nosp)[296:310],
                                    names(dd_ts_nosp)[311:325],
                                    names(dd_ts_nosp)[326:340],
                                    names(dd_ts_nosp)[341:355],
                                    names(dd_ts_nosp)[356:370],
                                    names(dd_ts_nosp)[371:385],
                                    names(dd_ts_nosp)[386:400],
                                    names(dd_ts_nosp)[401:415],
                                    names(dd_ts_nosp)[416:430],
                                    names(dd_ts_nosp)[431:445],
                                    names(dd_ts_nosp)[446:460],
                                    names(dd_ts_nosp)[461:475],
                                    names(dd_ts_nosp)[476:490],
                                    names(dd_ts_nosp)[491:505],
                                    names(dd_ts_nosp)[506:520],
                                    names(dd_ts_nosp)[521:535],
                                    names(dd_ts_nosp)[536:550],
                                    names(dd_ts_nosp)[551:565],
                                    names(dd_ts_nosp)[566:580],
                                    names(dd_ts_nosp)[581:595],
                                    names(dd_ts_nosp)[596:610],
                                    names(dd_ts_nosp)[611:625],
                                    names(dd_ts_nosp)[626:640],
                                    names(dd_ts_nosp)[641:655]),
                     idvar = "idpers", # DON'T use c(1) because you lose the original ID of the person
                     timevar = "Year", # column with name "year" for time-stamp
                     times = 2004:2018) #time-stamp for varying variables. Pay attention: starting from

# first re-ordering & re-name the (time-series) dataset:
which(colnames(data.long)=="Year")
dd_ts_nosp <- data.long[c(1,11,2:10,12:54)]

names(dd_ts_nosp) # we see that the variables kept the names of the
# starting-point wave, which is confusing, that's why I change their
# names in the next step. --> re-name some columns:
names(dd_ts_nosp)[12] <- "status"
names(dd_ts_nosp)[13] <- "idhous"
names(dd_ts_nosp)[14] <- "act"
names(dd_ts_nosp)[15] <- "age"
names(dd_ts_nosp)[16] <- "canton"
names(dd_ts_nosp)[17] <- "car"
names(dd_ts_nosp)[18] <- "chronic_lthp"
names(dd_ts_nosp)[19] <- "civsta"
names(dd_ts_nosp)[20] <- "CMJ_actual_hrsweek"
names(dd_ts_nosp)[21] <- "CMJ_desire_hrs"
names(dd_ts_nosp)[22] <- "contrinc_five"
names(dd_ts_nosp)[23] <- "contrinc_four"
names(dd_ts_nosp)[24] <- "contrinc_one"
names(dd_ts_nosp)[25] <- "contrinc_three"
names(dd_ts_nosp)[26] <- "contrinc_two"
names(dd_ts_nosp)[27] <- "doctor"
names(dd_ts_nosp)[28] <- "edyear"
names(dd_ts_nosp)[29] <- "empstat"
names(dd_ts_nosp)[30] <- "freetime_sat"
names(dd_ts_nosp)[31] <- "ginc_year"
names(dd_ts_nosp)[32] <- "h_impr"
names(dd_ts_nosp)[33] <- "height"
names(dd_ts_nosp)[34] <- "hh_incg"
names(dd_ts_nosp)[35] <- "hh_incn"
names(dd_ts_nosp)[36] <- "hhmove"
names(dd_ts_nosp)[37] <- "hprob_days"
names(dd_ts_nosp)[38] <- "hsat"
names(dd_ts_nosp)[39] <- "lalone_sat"
names(dd_ts_nosp)[40] <- "leisure_sat"
names(dd_ts_nosp)[41] <- "lsat"
names(dd_ts_nosp)[42] <- "ltogeth_sat"
names(dd_ts_nosp)[43] <- "min_inc"
names(dd_ts_nosp)[44] <- "nbpers"
names(dd_ts_nosp)[45] <- "ninc_year"
names(dd_ts_nosp)[46] <- "ownkid"
names(dd_ts_nosp)[47] <- "prob_children"
names(dd_ts_nosp)[48] <- "region"
names(dd_ts_nosp)[49] <- "sah"
names(dd_ts_nosp)[50] <- "sex"
names(dd_ts_nosp)[51] <- "sthhre"
names(dd_ts_nosp)[52] <- "weekly_act"
names(dd_ts_nosp)[53] <- "weight"
names(dd_ts_nosp)[54] <- "year_doct"

#2nd re-ordering the dataset:
names(dd_ts_nosp) # check if re-naming worked? --> yes!
dd_ts_nosp <- dd_ts_nosp[c(which(colnames(dd_ts_nosp)=="idpers"),
                           which(colnames(dd_ts_nosp)=="idhous"),
                           which(colnames(dd_ts_nosp)=="Year"),
                           which(colnames(dd_ts_nosp)=="status"),
                           which(colnames(dd_ts_nosp)=="age"), # 5
                           which(colnames(dd_ts_nosp)=="sex"),
                           which(colnames(dd_ts_nosp)=="civsta"),
                           which(colnames(dd_ts_nosp)=="edyear"),
                           which(colnames(dd_ts_nosp)=="empstat"),
                           which(colnames(dd_ts_nosp)=="CMJ_actual_hrsweek"), # 15
                           which(colnames(dd_ts_nosp)=="CMJ_desire_hrs"),
                           which(colnames(dd_ts_nosp)=="ownkid"),
                           which(colnames(dd_ts_nosp)=="prob_children"),
                           which(colnames(dd_ts_nosp)=="ltogeth_sat"),
                           which(colnames(dd_ts_nosp)=="lalone_sat"), # 20
                           which(colnames(dd_ts_nosp)=="weight"),
                           which(colnames(dd_ts_nosp)=="height"),
                           which(colnames(dd_ts_nosp)=="hprob_when"),
                           which(colnames(dd_ts_nosp)=="hprob_phys"),
                           which(colnames(dd_ts_nosp)=="hprob_cause"), # 25
                           which(colnames(dd_ts_nosp)=="chronic_lthp"),
                           which(colnames(dd_ts_nosp)=="doctor"),
                           which(colnames(dd_ts_nosp)=="year_doct"),
                           which(colnames(dd_ts_nosp)=="hprob_days"),
                           which(colnames(dd_ts_nosp)=="h_impr"), # 30
                           which(colnames(dd_ts_nosp)=="hsat"),
                           which(colnames(dd_ts_nosp)=="sah"),
                           which(colnames(dd_ts_nosp)=="lsat"),
                           which(colnames(dd_ts_nosp)=="freetime_sat"),
                           which(colnames(dd_ts_nosp)=="leisure_sat"), # 35
                           which(colnames(dd_ts_nosp)=="act"),
                           which(colnames(dd_ts_nosp)=="weekly_act"),
                           which(colnames(dd_ts_nosp)=="ginc_year"),
                           which(colnames(dd_ts_nosp)=="ninc_year"),
                           which(colnames(dd_ts_nosp)=="min_inc"), # 40
                           which(colnames(dd_ts_nosp)=="nbpers"),
                           which(colnames(dd_ts_nosp)=="hh_incg"),
                           which(colnames(dd_ts_nosp)=="hh_incn"),
                           which(colnames(dd_ts_nosp)=="contrinc_one"),
                           which(colnames(dd_ts_nosp)=="contrinc_two"),
                           which(colnames(dd_ts_nosp)=="contrinc_three"), # 45
                           which(colnames(dd_ts_nosp)=="contrinc_four"),
                           which(colnames(dd_ts_nosp)=="contrinc_five"),
                           which(colnames(dd_ts_nosp)=="nat_1"),
                           which(colnames(dd_ts_nosp)=="nat_2"), # 50
                           which(colnames(dd_ts_nosp)=="nat_3"),
                           which(colnames(dd_ts_nosp)=="ch_sinbirth"),
                           which(colnames(dd_ts_nosp)=="hab_ch"),
                           which(colnames(dd_ts_nosp)=="sib_outhh"),
                           which(colnames(dd_ts_nosp)=="region"),
                           which(colnames(dd_ts_nosp)=="canton"),
                           which(colnames(dd_ts_nosp)=="hhmove"),
                           which(colnames(dd_ts_nosp)=="car"),
                           which(colnames(dd_ts_nosp)=="sthhre")
)]
save(dd_ts_nosp, file="dd_ts_nosp_15y.Rda")## save dataset
# --> dd_ts_nosp == long-format // time-series dataset that does not
# contain any sport-variables, but only relevant socio-economic status
# information.
```

Second Split, with Sport-Variables:

Now comes the “second split” of the new “big dataset”. This time, I will take the part which contains only the sport-variablest. This step will be the “second split” (only sport-variables):

```
###### 8) "Second split" of the "big ddset" which contains sport-variables: ######

# First, let's re-load the "big dataset" which contains all the waves:
load("dd_ts_15y.Rda")
dd_ts <- dd_ts_15y
rm(dd_ts_15y)

# Some non(!!) sport-variables will be useful for the cleaning of the
# sport-variables later, that's why I will include them too in this
# "pure" sport-variables ddset:
```

```r
which(colnames(dd_ts)=="status13")
which(colnames(dd_ts)=="status15")
which(colnames(dd_ts)=="act13")
which(colnames(dd_ts)=="act15")
which(colnames(dd_ts)=="age13")
which(colnames(dd_ts)=="age15")
which(colnames(dd_ts)=="doctor13")
which(colnames(dd_ts)=="doctor15")
which(colnames(dd_ts)=="weekly_act13")
which(colnames(dd_ts)=="weekly_act15")
which(colnames(dd_ts)=="year_doct13")
which(colnames(dd_ts)=="year_doct15")

## Now construct sport-dataset: select sport-variables & some important
## "other variables" (see above) for this dd-set:
dd_ts_sp_wide <- dd_ts[c(1:20,30,32,60,62,75,77,256,258,639,641,669,671)]
save(dd_ts_sp_wide, file="dd_ts_sp_wide_15y.Rda")# save
load("dd_ts_sp_wide_15y.Rda")
```

Make a second Panel:

Again, let's transform the "second split" (= the part of the "big dataset" with sport variables) into a panel-dataset with the reshape()-function:

```r
###### 9) Transform second split from wide- into long-format with reshape() ######
# At the moment, the dataset is in wide-format with only sport variables
# and SOME socio-econ infos, so let's build a "short panel", since the
# sport-variables are only asked in waves 15 and 17.

# make the dataset in long-format:
names(dd_ts_sp_wide)

# re-order a bit first:
dd_ts_sp_wide <- dd_ts_sp_wide[c(20,1:19,21:32)]
dd_ts_sp_wide <- dd_ts_sp_wide[c(1:14,21:32,15:20)]
dd_ts_sp_wide <- dd_ts_sp_wide[c(1:13,15:26,14,27:32)]

# use reshape()-function for the transformation from wide- into
# long-format:
data.long <- reshape(dd_ts_sp_wide, direction = "long",
                     varying = list(names(dd_ts_sp_wide)[2:3],
                                    names(dd_ts_sp_wide)[4:5],
                                    names(dd_ts_sp_wide)[6:7],
                                    names(dd_ts_sp_wide)[8:9],
                                    names(dd_ts_sp_wide)[10:11],
                                    names(dd_ts_sp_wide)[12:13],
                                    names(dd_ts_sp_wide)[14:15],
                                    names(dd_ts_sp_wide)[16:17],
                                    names(dd_ts_sp_wide)[18:19],
                                    names(dd_ts_sp_wide)[20:21],
                                    names(dd_ts_sp_wide)[22:23],
                                    names(dd_ts_sp_wide)[24:25],
                                    names(dd_ts_sp_wide)[26:27]),
                     idvar = "idpers", # variables that stay fixed and are not transformed. DON'T use c
```

```r
                      timevar = "Year", # column with name "year" for time-stamp
                      times = c(2013, 2015)) #time-stamp for varying variables

# re-ordering the dataset again:
which(colnames(data.long)=="idpers")
which(colnames(data.long)=="Year")
dd_ts_sp <- data.long[c(1,7,2:6,8:20)]
names(dd_ts_sp)

# re-name the columns:
names(dd_ts_sp)[3] <- "ycompet"
names(dd_ts_sp)[4] <- "yduration"
names(dd_ts_sp)[5] <- "yfrequency"
names(dd_ts_sp)[6] <- "yref_per"
names(dd_ts_sp)[7] <- "ytype"
names(dd_ts_sp)[8] <- "agesport"
names(dd_ts_sp)[9] <- "compet"
names(dd_ts_sp)[10] <- "duration"
names(dd_ts_sp)[11] <- "frequency"
names(dd_ts_sp)[12] <- "ref_per"
names(dd_ts_sp)[13] <- "type"
names(dd_ts_sp)[14] <- "status"
names(dd_ts_sp)[15] <- "act"
names(dd_ts_sp)[16] <- "age"
names(dd_ts_sp)[17] <- "doctor"
names(dd_ts_sp)[18] <- "weekly_act"
names(dd_ts_sp)[19] <- "year_doct"
names(dd_ts_sp)[20] <- "waythere"

# some further re-ordering:
dd_ts_sp <- dd_ts_sp[c(which(colnames(dd_ts_sp)=="idpers"),
                       which(colnames(dd_ts_sp)=="Year"),
                       which(colnames(dd_ts_sp)=="status"),
                       which(colnames(dd_ts_sp)=="age"),
                       which(colnames(dd_ts_sp)=="act"), # 5
                       which(colnames(dd_ts_sp)=="weekly_act"),
                       which(colnames(dd_ts_sp)=="ytype"),
                       which(colnames(dd_ts_sp)=="type"),
                       which(colnames(dd_ts_sp)=="agesport"),
                       which(colnames(dd_ts_sp)=="yref_per"), # 15
                       which(colnames(dd_ts_sp)=="ref_per"),
                       which(colnames(dd_ts_sp)=="compet"),
                       which(colnames(dd_ts_sp)=="duration"),
                       which(colnames(dd_ts_sp)=="frequency"),
                       which(colnames(dd_ts_sp)=="waythere"),
                       which(colnames(dd_ts_sp)=="ycompet"),
                       which(colnames(dd_ts_sp)=="yduration"),
                       which(colnames(dd_ts_sp)=="yfrequency"),
                       which(colnames(dd_ts_sp)=="year_doct"),
                       which(colnames(dd_ts_sp)=="doctor")
)]
dd_ts_sp <- dd_ts_sp[c(1:9,20,14,10,12:13,15:17,19,18,11)]
names(dd_ts_sp) # check if ordering ok? --> yes!
```

```
save(dd_ts_sp, file="dd_ts_sp.Rda")# save dataset dd_ts_sp == long-format
# dataset with only sport information of year 2013 & 2015.

# Remove some datasets (I saved them all):
# rm(list = c("data.long", "dd_ts", "dd_ts_nosp", "dd_ts_sp", "dd_ts_sp_wide"))
```

And So On…

This was only about **10 percent** of the whole data cleaning process. It only served as an illustration of how I work with data and illustrate a bit what I am capable of. I hope you liked it =)

If you wish to gain access to the whole code, I will be pleased to send it to you via E-Mail!