Real world data with R

Practical

Alexandru Cernat

Practical 1 - Preparing the European Social Survey

The European Social Survey is one of the biggest cross-national surveys in Europe. It is carried out every 2 years and it includes around 20 countries. It can be used for multilevel analysis both at the country level (although note the small number of countries) and at the regional level (NUTS). It is renowned for its quality and the wide ranging topics covered from the social sciences. In this exercise we are going to clean the data the tidyverse way.

First make sure you have the data. Open Rstudio. Open a new script (*File -> New -> Rscript*). You should write everything in the Rscript and then run it (using *ctrl/cmnd + Enter*).

We will first do some admin. We will install some packages. One is tidyverse and one is for importing "foreign" data (`haven`) *(if you already installed the packages you can skip this step)*. We will also set the working directory. Make sure to have all the data in the respective folder to facilitate data handling *(note that in `R` paths are written with `/` or `\\` as opposed to the default `/` in Windows)*. Here is an example of the syntax:

```
install.packages("tidyverse") # install package for data managemant
install.packages("haven") # install package for importing data

setwd("P:/Data/") # set working directory (change to fit your computer)
```

If you have difficulties setting up the working directory you can use the menu: Session -> Set Working Directory -> Choose directory (or cntrl/cmnd + shift + H)

Also, if you want to learn more about certain commands you can use `?commandname` in `R` or search using your favorite search engine. Note `#` is used to comment in `R`.

We will next import the full ESS data. First we need to load the two packages we have installed using the `library()` command. In `R` installing and loading packages are two different processes.

We will import the data (which is in Stata format) using the `read_dta()` command from the `haven` package. `R` is an object oriented program. As such, we need to put the data in an appropriate object. Let's call it "ess_full". Finally, let's describe the data using the `head()`, `dim()` and `glimpse()` commands. Here is an example of this syntax:

```
# load packages
library(tidyverse)
library(haven)

# read ESS data and put in object
ess_full <- read_dta("ESS9e01_1.dta")

# read country level data
ess_cntry_full <- read_dta("ESSMD-2018-cntry_F1.dta")

# describe
```

```
head(ess_full)
dim(ess_full)
glimpse(ess_full)
```

Based on this how many variables and cases do we have? Use the same commands to explore the country level data.

The data is on the larger side. Let's try to make it smaller so it's easier to handle. The first thing we might want to do is to select only the variables we are interested in. We do this using the `select()` command. The simplest version of this takes as options the data name and the variables we want to keep. Try to use the command bellow. You will note that we are saving this smaller data in a new object called "ess". We then check if everything worked (you should always do these kind of checks!).

You might wonder how I know what variables to choose. I looked at the questionnaire and found the variables that would be interesting for our work. Normally you need to figure this out on your own.

```
# select variables
ess <- select(ess_full, idno, cntry, imbgeco, imueclt, imwbcnt,
              vote, uempla, agea, eduyrs, gndr, region)

# descriptive of the data
head(ess)
glimpse(ess)
```

The `glimpse()` command lists all the variables and some information about them. One important thing in `R` is the type of objects you have. This will influence what you can do with the data. We see that some variables that we expected to be numeric were imported as `fctr` (i.e., factor). We see that most variables are *dbl+lbl*. This means that the variables are numerical but have labels. We will need to transform some of these to prepare for graphs and analysis.

Let's have a look at the gender variable ("gndr") using the `count()` and `attributes()` commands:

```
# table of variable
count(ess, gndr)

# attributes of variable
attributes(ess$gndr)
```

Let's look at the education variable as well. Here we ask `R` to print 50 rows as by default it only prints 10 for the count command.

```
# descriptive table for years of education
count(ess, eduyrs) %>% print(n = 50)
```

Are there missing cases? How many of those are present?

Sometimes we might want to select only certain cases. This can be for different reasons. We might be interested in subgroups, we have missing data or some people are out of scope (ineligible). We can easily do that using the `filter()` command. Let's imagine that for our research the education variable is essential. As such, we are going to use only those respondents that answered to this question. The `filter()` command tells `R` which are the observations that it **should keep**. The `NA` has a special treatment in `R`. As such, we need to use the `is.na()` command to identify the cases that have missing on this variable. This will result in a vector that gets `TRUE` if a case has `NA` or `FALSE` otherwise. For the `filter()` command we want the reverse. As such, we have to use the negation command (using `!`). Here is an example of this. You will see that this also overrides the original data, as such those people will not be available anymore (one alternative would be to create a new dataset with the subset of cases).

```r
 # filter data if they have missing on education
ess <- filter(ess, !is.na(eduyrs))

# descriptive table for years of education
count(ess, eduyrs) %>% print(n = 50)
```

Joining data

We have imported data both at the individual and country level. Now that we have the cases we want at the individual level let's bring in the country data. First, let's look at the country data:

```r
dim(ess_cntry_full)
head(ess_cntry_full)
glimpse(ess_cntry_full)
```

There are only a few cases but lots of variables. Let's select only the variables from 2018 plus the "cntry" variable which is the key in this dataset.

```r
# select only variables that have 2018 at the end of the name
ess_cntry <- select(ess_cntry_full, cntry, ends_with("2018"))

# check new data
glimpse(ess_cntry)
```

Now we have the data we want both at the individual and country level. Let's see how we can do the merging. Our safest bet is the "cntry" variable that is present in both datasets. Let's check if this is unique in country level data:

```r
ess_cntry %>% # select the data we want
  group_by(cntry) %>% # group data by country
  count() %>% # count how many times each group appears
  filter(n > 1) # keep only the countries that appear more than once
```

What is the conclusion? Is it unique?

Check if "cntry" is a unique id in the individual data. What is the conclusion?

Before doing the linkage we want to make sure we have a unique id at the individual level. **Check if "idno" uniquely identifies the cases in the "ess" data.**

Let's make a unique id before we do the merging. We will first arrange the data and then use the row number to create a unique id.

```r
# make unique id
ess <- ess %>%
  arrange(cntry, idno) %>% # arrange rows by country and id
  mutate(id = row_number()) # make a new variable called id = row number
```

Check if the new id is unique.

We are now ready for the merging. The main data we are interested in is at the individual level. As such, we will use that as the reference and only get country data for those that are present there. Because of that we can use `left_join()` and put the individual data as the first input. The key of the merger will be "cntry". We saw that this is the key in the country data and is a foreign key for the individual data. Let's check if we have info on all the countries.

We will use a combination of the `unique()` function that finds unique values and `%in%` which does a logical check if the values on the left side are present in the right side.

```r
# check if the ids look the same in both data
count(ess, cntry)
count(ess_cntry, cntry)

# check that all the countries in the individual data
# are in the country data
unique(ess$cntry)
unique(ess_cntry$cntry)

cntry_present <- unique(ess$cntry) %in% unique(ess_cntry$cntry)

# some countries have FALSE ->
# are in individual data but not country data
cntry_present

# which countries are not present in the country data?
unique(ess$cntry)[!cntry_present]
```

Which countries don't have country level info?

We will do the merger but it's good to know that at least three countries will not have this information. So when using the info we will trade-off the extra information for losing some cases.

```r
# merging the individual data and country data by the cntry var
ess2 <- left_join(ess, ess_cntry, by = "cntry")

#check results
dim(ess2)
glimpse(ess2)
```

Cleaning data

Now that we are happy with the variables and cases we have let's clean the data.

Let's look at the "imbgeco" variable ("Immigration bad or good for country's economy"). We can use the `count()` command to see the frequency. To make it easier to read you can also calculate a new variable that gives the proportion.

```r
# frequency with proportions for support for immigration
count(ess2, imbgeco) %>%
  mutate(prop = n/sum(n))

# see the labales
attributes(ess2$imbgeco)
```

Repeat this process and check "imueclt" and "imwbcnt".

Do the variables look OK? Are the missing cases coded as NA?

Let's have a look at the age variable:

```r
count(ess2, agea) %>% print(n = 150)

# quick plot (more on plots in part 2)
qplot(ess2$agea)
```

Does it look OK to you? Things to look at: the range, the distribution, missing cases, etc.

So far we looked at continuous variables. Let's have a look at a few categorical variables as well.

Let's check first gender:

```
attributes(ess2$gndr)
count(ess, gndr)
```

We see that we have codes of 1 and 2 which makes them hard to read. Let's make it a factor. This will insure that graphs and statistical analysis show the correct labels and makes it easier to use:

```
# make new factor variable
ess2 <- ess2 %>%
  mutate(sex = factor(gndr, labels = c("Male", "Female")))

# check if new varaible is correct
count(ess2, gndr, sex)
```

Let's next check the vote variable:

```
attributes(ess2$vote)
count(ess2, vote)
```

We have three categories now. We want to code the ineligibles as missing and then convert it to a factor. A very useful function to know is `case_when()`. This can get as an input multiple conditions that are used to create a new variable. By default, if there are cases that are not mentioned in the conditions the function will code them as missing. So, we can use that, code 1 as "Yes", 2 as "No" and not mention cases with 3. This will create a string or character variable. We will transform it to a factor using the `as.factor()` command. Let's have a look at how to implement it.

```
# recode vote and make a factor
ess2 <- ess2 %>%
  mutate(voted = case_when(vote == 1 ~ "Yes",
                           vote == 2 ~ "No"),
         voted = as.factor(voted))

# check if it is correct
ess2 %>%
  count(vote, voted)
```

Now you do it. Make a new variable: "unemployed" based on "uempla". The new variable should be a factor and take the value "Yes" if someone is looking for a job and "No" otherwise.

We are getting to the end of the coding. When we start doing graphs and models we often want to be flexible in how we use our variables. A typical example is modeling non-linear effects. There are two main ways to do this. One, you can divide a continuous variable into discrete categories and model separately each category. Second, we can include non-linear effects, such as square or cube effects. Let's prepare this for age:

```
ess2 <- ess2 %>%
  mutate(agea2 = agea ^ 2, # age squared
         agea_cat = cut(agea, c(14, 25, 45, 65, 100))) # age groups

# check
qplot(ess2$agea, ess2$agea2)
qplot(ess2$agea, ess2$agea_cat)
```

Do the same for education (choose appropriate cut-off points).

Another typical thing we do when preparing variables for analysis is to center them. This involves moving the mean of the variable to 0 (the rest of the distribution stays the same). **Why do you think we do this?**

This is easy to do. We just want to subtract the mean from each of the values of the variable. We can use the `mean()` command. Let's do this for years of education. Save the result in a new variable called "edu_center".

```
# center multiple variables
ess2 <- ess2 %>%
  mutate(eduyrs_center =  eduyrs - mean(eduyrs, na.rm = T),
         agea_center =  agea - mean(agea, na.rm = T))

# check results
qplot(ess2$agea, ess2$agea_center)
qplot(ess2$eduyrs, ess2$eduyrs_center)
```

Additionally, we can center by group. This can be very useful when doing statistical models such as multilevel models.

```
# center variable by country
ess2 <- ess2 %>%
  group_by(cntry) %>%
  mutate(eduyrs_cntry_center =  eduyrs - mean(eduyrs, na.rm = T),
         agea_cntry_center =  agea - mean(agea, na.rm = T)) %>%
  ungroup()

# check
qplot(data = ess2, agea, agea_cntry_center, facets = ~cntry)
qplot(data = ess2, eduyrs, eduyrs_cntry_center, facets = ~cntry)
```

Use qplot and tables to check the country level variables. Are there variables that have many missing cases?

Finally, let us rename the country level variables to make them easier to read. Let us delete the "_2018" prefix from all of them using `rename_all()`. This applies a function to all the variables in the dataset. In this case it uses `str_remove()` on all the variables (represented by `.`).

```
ess2 <- ess2 %>%
  rename_all(~str_remove(., "_2018"))
```

Before saving the data we can also eliminate some variables we won't use, either because they are re-coded or because they have too many missings. We also want to reorder the variables to be easier to find.

```
ess3 <- select(ess2, id, cntry, everything(),
               -vote, -uempla, -gndr, -c_gini)
```

Let's do a quick summary of all the data:

```
summary(ess3)
```

Finally, let's save the dataset for future use:

```
write_rds(ess3, "ess_prep.rds")
```

Practical 2 - Visualizing ESS data

Next we will use `ggplot` to explore the variables we have cleaned in the ESS data. Use the slides to create some of the graphs mentioned bellow.

Let's start by importing the data:

```
ess3 <- read_rds("ess_prep.rds")
```

Uni-variate descriptive

We will start by doing some uni-variate graphs.

Use `geom_histogram()` and create a graph for each: "imbgeco", "imueclt", "imwbcnt".

Do the same for "eduyrs" and "agea" but play around with the bin-width to see how it influences the presentation. Also try using `geom_density()`.

Let's look at some categorical variables. Use `geom_bar()` to explore: "sex", "voted", "unemployed", "eduyrs_cat", "agea_cat". Try to compute the proportion instead of the counts for some of them.

Bi-variate relationships

We can make the graphs more interesting by looking at multiple variables. Let's start with some scatter plots. These are very useful for looking at the relationship of continuous variables.

Let's start by trying to understand the relationship between education ("eduyrs") and "imbgeco" (support for immigration):

```
ggplot(ess3, aes(eduyrs, imbgeco)) +
  geom_point()
```

Do you think there is a relationship?

To help make the graph easier to read let's change it a little. Add `geom_smooth()` and use the `alpha` to increase the transparency of the points.

We have a few extreme cases. Let's see if the relationship changes if we just look at cases that have under 30 years of education.

```
ess3 %>%
  filter(eduyrs < 30) %>%
  ggplot(aes(eduyrs, imbgeco)) +
  geom_point(alpha = 0.05) + geom_smooth()
```

Do a similar graph for "imueclt" and "imwbcnt". If you don't remember what they mean check the attributes.

Let's look at the relationship between categorical and continuous variables. **Investigate the relationship between "imbgeco" and "sex" and "voted" using `geom_boxplot()` and `geom_violin()`.**

Sometimes it's easier to see relationships when we have categorical data. **Look at the relationship between "imbgeco" with "agea" and "agea_cat".** You can use scatter-plot for the first one and box-plots for the second one. **What is your conclusion about the relationship between the two variables?**

Finally, let's look at the relationship between two categorical variables. Let's see if there is a relationship between voting and the categories of age we made. First let's do a bar-plot:

```
ess3 %>%
  ggplot(aes(voted, fill = agea_cat)) +
  geom_bar()
```

This is a little hard to read. **Use first the `position = "dodge"` option and then the `position = "fill"` to see if it helps with reading the plot.**

Look at the example in the slides. Also try to look at the relationship using `geom_tile()`.

Multivariate relationships

As we saw in the slides `ggplot` is extremely flexible so we can easily look at multiple variables at the same time. This can help us see important patterns and inform possible non-linear or interaction effects.

Starting from the plot bellow expand it to: change color by sex and do the graph separately for each country.

```
ess3 %>%
  filter(eduyrs < 30) %>%
  ggplot(aes(eduyrs, imbgeco)) +
  geom_point(alpha = 0.05) + geom_smooth()
```

Are there differences between men and women in the effect of education of immigration support? Does this vary by country?

Look at the relationship between voting and categorical education by country. Do you notice any patterns?

Changing the look

Let's make our graphs nicer. **Take the graph bellow and try out different themes.** (see list of themes in the slides).

```
ess3 %>%
  filter(eduyrs < 30) %>%
  ggplot(aes(eduyrs, imbgeco, color = sex)) +
  geom_point(alpha = 0.05) + geom_smooth(method = lm) +
  facet_wrap(~cntry)
```

If you feel experimental you can install the `ggthemes` package and try the themes they have there.

Once you decide on a theme add labels to the graph. Add the following labels: title, subtitle, x, y, caption, color.

Finally, let's save the graph for later use. **Save the previous graph as an object called "grph2" and save it:**

```
ggsave("nice_graph.png", grph2, dpi = 300)
```