Code

Exploratory Data Analysis using the Crime Survey for England and Wales

Ana Morales-Gomez

2020-01-29

In this practical you will be analysing a sample of the teaching dataset of the Crime Survey for England and Wales available here (https://beta.ukdataservice.ac.uk/datacatalogue/studies/study?id=7911) from the UK Data Service (https://www.ukdataservice.ac.uk/). But you can access it from the folder **Crime_workshop** in the "C:" folder of the PC in the cluster.

1. Before we start

1.1. Define a working directory

You can use the directory on University-owned machines, as it provides a stable drive to work on ("C:"). If you're using your own machine, you can specify any folder you find convenient.

Also note that you need to take your data and results with you and delete what you store on the C-drive after this session (if you are not using your own laptop).

1.2. Open a new Script

First, open a new script in R studio and save it in your working directory, so you will be able to access this script at a later time if you want to revise or modify a code.

In R Studio:

Go to File… New File…. R script

1.3. Load the packages

Remember to load the packages. We will be using `tidyverse` and `haven`

```
library(haven)
library(tidyverse)
```

As a first step in a research project we need to start with a purpose:

2. Import the data into R

Import the dataset called crime_survey.dta into the console using your desired method (code or point and click). This dataset can only be imported if you have loaded the package `haven`.

```
# Do it yourself

csew<-
```

3. Exploratory Data Analysis

3.1 Dataset exploration

Task 1: Inspect the dataset, use the function View()

```
View(csew)
```

You can also use the function `head()` that shows you the first 6 rows of the data set, this function is useful to have a look at the data within the console, specially when you have large datasets.

There is also an antithesis function to `head()` called `tail()`. Can you guess what that function does?

(**Hint**: you can run the code in the console and see it by yourself, or a quick search in Google may also help.)

```
head(csew)
```

Data obtained from the UK Data Service always come with a wealth of resources and user guides that help us to understand the data, so I would always encourage you to check the documentation associated with the data used.This documentation also includes information about the sample, the questionnaire, the variables available and how to use them.

Sadly, this is not always the case with data from different sources, in those cases the initial exploration of the data is probably our main source of information about the data itself.

Now that we got the data in the console, let's start with our data exploration:

Task 1.1 How many observations does the data have?_____

Task 1.2 How many variables?_____

Another function used to get a "glimpse" at your data is using the function `glimpse()`.

```
glimpse(csew)
```

Note the data type of almost all variables is `<dbl+lbl>`. These are labelled variables in the haven package. It's the way to identify variables with labels from Stata.

3.2 Variable exploration

Let's use another function to have a closer look, this time at the variables. Let's use this with the variable 'sex'

```
attributes(csew$sex)
```

```
## $label
## [1] "Respondent's gender"
##
## $format.stata
## [1] "%8.0g"
##
## $class
## [1] "haven_labelled"
##
## $labels
##   Male Female
##      1      2
```

We have the `$class` attribute which says that `sex` is a haven_labelled type. We also have the `$labels` attribute which are the Value Labels of the variable `sex` : 1 Man, 2 Woman

We could have also used `class(hse$sex)` to check the type of the variable.

Task 2 Inspect other variables with attributes() and class()

4. Understanding our data

Now that we have had a look at what variables we have, we need to start looking for other aspects of the exploratory data analysis (EDA) process. These are some of the things to look out for:

- Checking for outliers (unusual values)
- Looking at the distribution of the variables
- Exploring some relationships and patterns
- Checking for missing cases

4.1. Univariate analysis

We can use a series of descriptive statistics and graphs to help us to understand and make sense of the data.

Univariate exploration is when we look at the characteristics of each variable of interest independently.

Univariate Descriptive statistic for categorical (factor) data

Let's see what happens when we ask for a frequency table

```
table(csew$sex)
```

```
##
##     1     2
## 16176 19195
```

Right now, the variable does not contain any visible labels that help us to identify each category of response, although we know the underlying values of each label thanks to the `attribute` function.

This is one of the inconveniences of working with data with labels in R, but it is good for you to know what to do in this case.

Although we can use the data as it is, I prefer to see the values of the variables, so we are going to change the class of some variables to make them easier to analyse.

We can use the function `as_factor()` from the package **haven** to convert the variable into a factor variable (equivalent to categorical variable). Copy the following code in your console.

```
csew$sexf<-as_factor(csew$sex)
table(csew$sexf)
```

```
##
##   Male Female
##  16176  19195
```

Here we created a new variable with the suffix "f" (for factor).

Task 3: Convert the following variables into factor using the example code before.

- nation: adult respondent nationality
- educat3: respondent education (5 categories)
- ethgrp5a: ethnic group
- bcsvictim: Experience of any crime in the previous 12 months

Missing values

In R missing values are identified as NA.

Now inspect whether there are missing values in our variables

```
#is.na(csew$sexf)

sum(is.na(csew$sexf))
```

```
## [1] 0
```

```
sum(is.na(csew$nationf))
```

```
## [1] 0
```

The function `is.na` is a logical function which looks at each observation and evaluates whether it is a valid case or a missing case. It will show a series of values `TRUE` for those observations that are missing and `FALSE` for valid cases (is.na = FALSE means that the cases are valid).

It is a good function to explore missingness, but it is not very practical on its own (as you could see in the example) that is why it is often used along other statistics.

I used the statistic `sum` which will add all the `TRUE` values of `is.na` so the result will be the number of NA (missing cases) for a particular variable.

The data shows that there are no missing values, but we can infer that the values 8 and 9 can be classified as missing cases, so we recode them as missing

```
csew$nationf<- recode(csew$nationf, "Refused" = "NA", "Don't know" = "NA")

table(csew$nationf)
```

```
##
##      UK, British          English         Scottish            Welsh
##            21106             9780              306             1519
##         Northern Irish (Republic)            Other               NA
##               73              269             2298               20
```

We have successfully recoded those values as missing, now we need to tell R Studio to treat those values as missing too

```
csew$nationf<- na_if(csew$nationf, "NA")

csew$nationf<- droplevels(csew$nationf) #with this code we got rid of the cat
egory for NA
table(csew$nationf)
```

```
##
##      UK, British          English         Scottish            Welsh
##            21106             9780              306             1519
##         Northern Irish (Republic)            Other
##               73              269             2298
```

Task 4: Check for missing values in the following variables and deal with them accordingly:

- educat3: respondent education (5 categories)
- ethgrp5a: ethnic group
- bcsvictim: Experience of any crime in the previous 12 months

Univariate Descriptive statistic for continuous (numeric) data

We can get summaries of the main statistics using the base function `summary()`. This function gives us the minimum and maximum values, the mean, median and the quartiles.

It is really useful to check for extreme values. Let's have a go

```
#This uses basic functions from R
summary(csew$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   16.00   36.00   51.00   54.35   66.00  999.00
```

`summary()` does not give any measure of the spread of the data, we can use other functions available in base R for that `sd()` for standard deviation.

```
# Standard deviation
sd(csew$age)
```

```
## [1] 57.68425
```

so far we have used two different functions of base R for exploratory data analysis. We can obtain the same statistics and more using a combination of different function from the package `tidyverse` taking advantage of the `%>%` (pipe) operator from the package **magrittr** which is fully implemented in tidyverse, so there is no need to install anything else.

I'm a firmer believer (although I may be wrong) that the best way to learn is putting into practice, so this is a perfect opportunity to put `%>%` into practice. (You can check your notes of tutorial 1 for more details on how to use them).

Let's start with the `summarise` function from the package `dplyr` (also part of tidyverse) This function allows us to create summary statistic of our variables in one single code

```
#using function from the package dplyr (tidyverse)
csew %>%
  summarise(mean_age= mean(age)) # we create the variable mean_age
```

```
## # A tibble: 1 x 1
##   mean_age
##      <dbl>
## 1     54.4
```

That was good but not very impressive, was it? We can add more functions to that code and ask for more:

```
csew %>%
  summarise(mean_age= mean(age),
            sd_age= sd(age),
            median_age = median(age),
            min_age = min(age),
            max_age = max(age))
```

```
## # A tibble: 1 x 5
##   mean_age sd_age median_age min_age max_age
##      <dbl>  <dbl>  <dbl+lbl>   <dbl>   <dbl>
## 1     54.4   57.7         51      16     999
```

One of the advantages of using `%>%` is that we can add more functions to our code as a set of instructions. In the example below, I added the function `filter()` of the package `dplyr` to exclude all missing cases from the analysis.

The function `is.na()` (is missing?) will select all missing cases, but when used along the `!` it means totally the opposite, so in this code this stand for **filter all the cases of variable age that *are not* missing**

```
csew %>%
  filter(!is.na(age)) %>%
  summarise(mean_age= mean(age),
            sd_age= sd(age),
            median_age = median(age),
            min_age = min(age),
            max_age = max(age))
```

```
## # A tibble: 1 x 5
##   mean_age sd_age median_age min_age max_age
##      <dbl>  <dbl>  <dbl+lbl>   <dbl>   <dbl>
## 1     54.4   57.7         51      16     999
```

We can even add more variables! but for that we might need another function and the chance to use more pipes %>%.

- select (https://dplyr.tidyverse.org/reference/select.html): This function allows to select specific variable from the datasets.
- summarise_all (https://dplyr.tidyverse.org/reference/summarise_all.html): allows to apply the same transformation to multiple variables
- list (https://www.rdocumentation.org/packages/base/versions/3.6.2/topics/list): used to make a list of functions to apply, in this example, mean and sd

```
csew %>%
  select(age, antisocx) %>%
  summarise_all(list(mean= mean, sd= sd, median = median), na.rm = TRUE)
```

```
## # A tibble: 1 x 6
##   age_mean antisocx_mean age_sd antisocx_sd age_median antisocx_median
##      <dbl>         <dbl>  <dbl>       <dbl>  <dbl+lbl>           <dbl>
## 1     54.4 0.00000000547   57.7       1.000         51          -0.177
```

What can we say about age?

The distribution of the variable age seems reasonable, the mean and the median are very similar.

The standard deviation shows how spread the data are. A standard deviation of 57.68 is quite high, which might indicate a lot of variation in the distribution of age. This is clearly hinting that there is something odd about the variable.

We also saw that the maximum value for age is **999**, which is clearly a number out of range; Nobody can live for 999 years! This is probably the reason why the standard deviation is so high, **we clearly need further exploration of this unusual value**

In certain cases the maximum values can be less obvious, so it would be difficult to determine whether there are outliers or data errors, that is why we need to complement these initial checks with visual exploration.

Diagnostic, critical points regarding age

- high standard deviation
- Implausible values (error?, missing values?, outliers?)

Actions

- both problems might be related
- check 999 values and deal with them accordingly
- check if SD changes after 999 values are taken into account

```
class(csew$age)
```

```
## [1] "haven_labelled"
```

```
attributes(csew$age)
```

```
## $label
## [1] "Respondent's age"
##
## $format.stata
## [1] "%8.0g"
##
## $class
## [1] "haven_labelled"
##
## $labels
##    Refused Don't know
##        998        999
```

using the function `attributes()` we now know

- age is formatted as "%8.0g" which is a way of storing numeric data in Stata.
- age is a "haven_labelled" variables, which means there are certain values with labels
- The values and labels 998 and 999 are for "refused" and "don't know" respectively.

998 and 999 can be classified as missing cases (unless you are interested in knowing the patterns of non-response). So we can easily solve that problem.

```
csew$age[csew$age==998 | csew$age==999 ]<-NA
```

Task 5: Get summary statistics for the variables **confx**, **fair** and **effectx**. Make note of any patterns or unusual values. A description of the variables can be found in this user guide (http://doc.ukdataservice.ac.uk/doc/7911/mrdoc/pdf/7911_csew_2013-14_teaching_dataset_user_guide.pdf)

Task 5.1 Write down a short description of the main statistics you've obtained

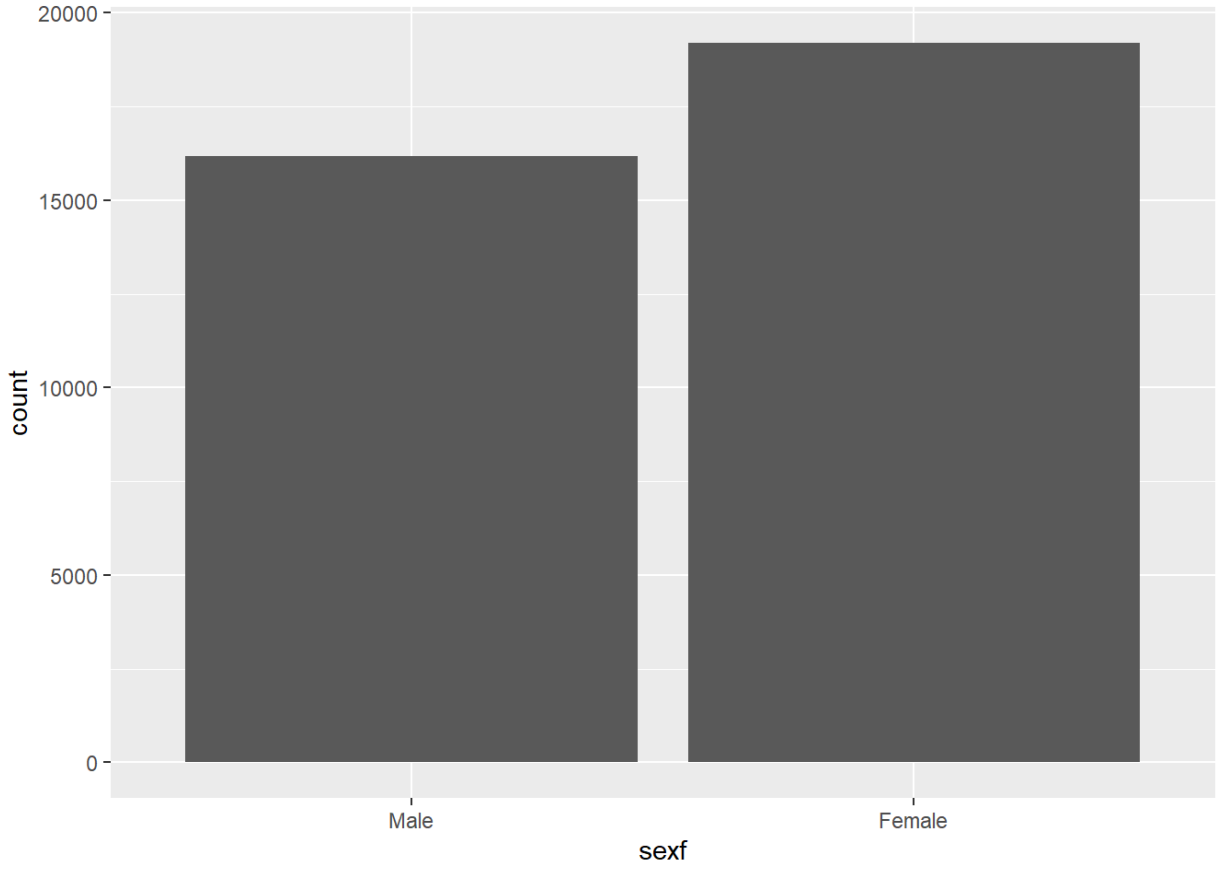
Task 5.2 Is there any unusual value in the variables you have analysed?

4.2. Visual exploration

We can use a combination of bar plots, histograms, boxplots and scatter plots as the basis for our data exploration

bar plots of categorical data

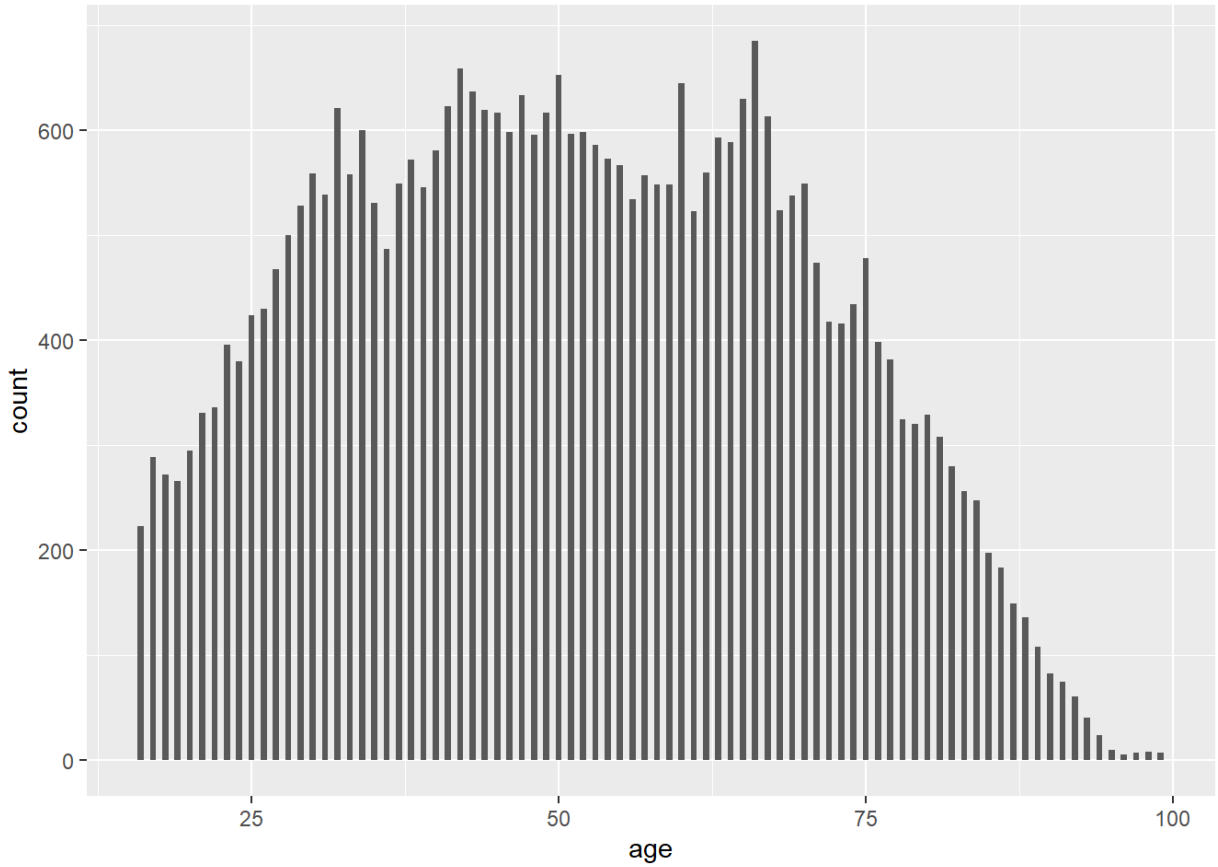
```
ggplot(data = csew) +
  geom_bar(mapping = aes(x = sexf))
```



histogram for continuous data

```
ggplot(data = csew) +
  geom_histogram(mapping = aes(x=age), binwidth = 0.5, na.rm = TRUE)
```

```
## Don't know how to automatically pick scale for object of type haven_labell
ed. Defaulting to continuous.
```



Both bar charts and histograms can be used to detect outliers and unusual patterns of the data.

Task 6: Visually explore the variables that we have used in the previous task. Make a note of any unusual patterns that you can find.

4.3. Association between variables

More insight can be gained when looking at relationships between variables. This can be done in two main ways: using numerical representation of this association and visual representations.

Associations using “crosstabs” or “contingency” tables

For categorical data we can use contingency tables (also called crosstabs)

Here we will be using a new package called `janitor` .

```
#install.packages("janitor")
library(janitor)
```

```
## Warning: package 'janitor' was built under R version 3.6.2
```

```
csew %>%
  tabyl(sexf, nationf, show_na = FALSE) %>% #excludes NA values from the tabl
e
  adorn_percentages("row") %>% # takes row percentage
  adorn_pct_formatting(digits = 2) %>% # limit % to 2 digits
  adorn_ns() #add the cell count
```

```
##    sexf    UK, British       English    Scottish       Welsh   Northern
##    Male 57.10%  (9232) 30.23% (4887) 0.91% (147) 4.26% (689) 0.18% (29)
##  Female 61.90% (11874) 25.51% (4893) 0.83% (159) 4.33% (830) 0.23% (44)
##  Irish (Republic)        Other
##       0.79% (127) 6.53% (1056)
##       0.74% (142) 6.47% (1242)
```

Task 7: Explore the relationship between victim of crime and sex, ethnic group, nationality and educational level. Make notes of your findings

Looking at association through visualisations

Now we need to focus on our variable of interest (outcome usually represented as the 'y' variable) and its association with other explanatory variables (x variables).

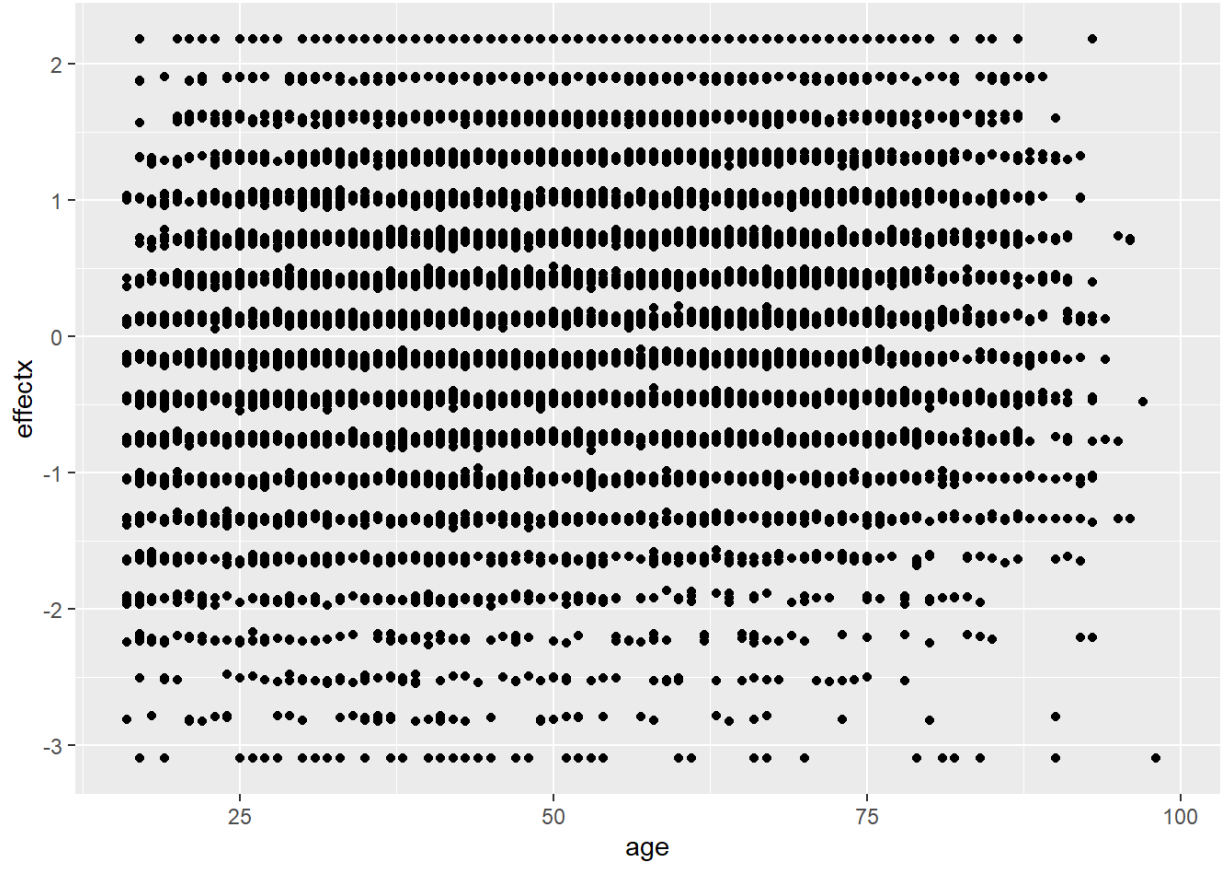
Here we will be looking at the associations between confidence in the effectiveness of Criminal Justice System (effectx) and other individual characteristics of the respondents.

Two continuous variables

-Scatter plots: For looking at the relationship between two continuous variables. This is useful to assess co-variation.

```
ggplot(data = csew, mapping = aes(x = age, y = effectx)) +
  geom_point(na.rm = TRUE)
```

```
## Don't know how to automatically pick scale for object of type haven_labell
ed. Defaulting to continuous.
```



We received a warning regarding a labelled variable, further exploration reveals that the variable age is numeric but contains two labels ("haven_labelled") for "refused" and "don't know".

The `as.numeric()` function will create a copy of the variable age of class "numeric". The function will get rid of the labels.

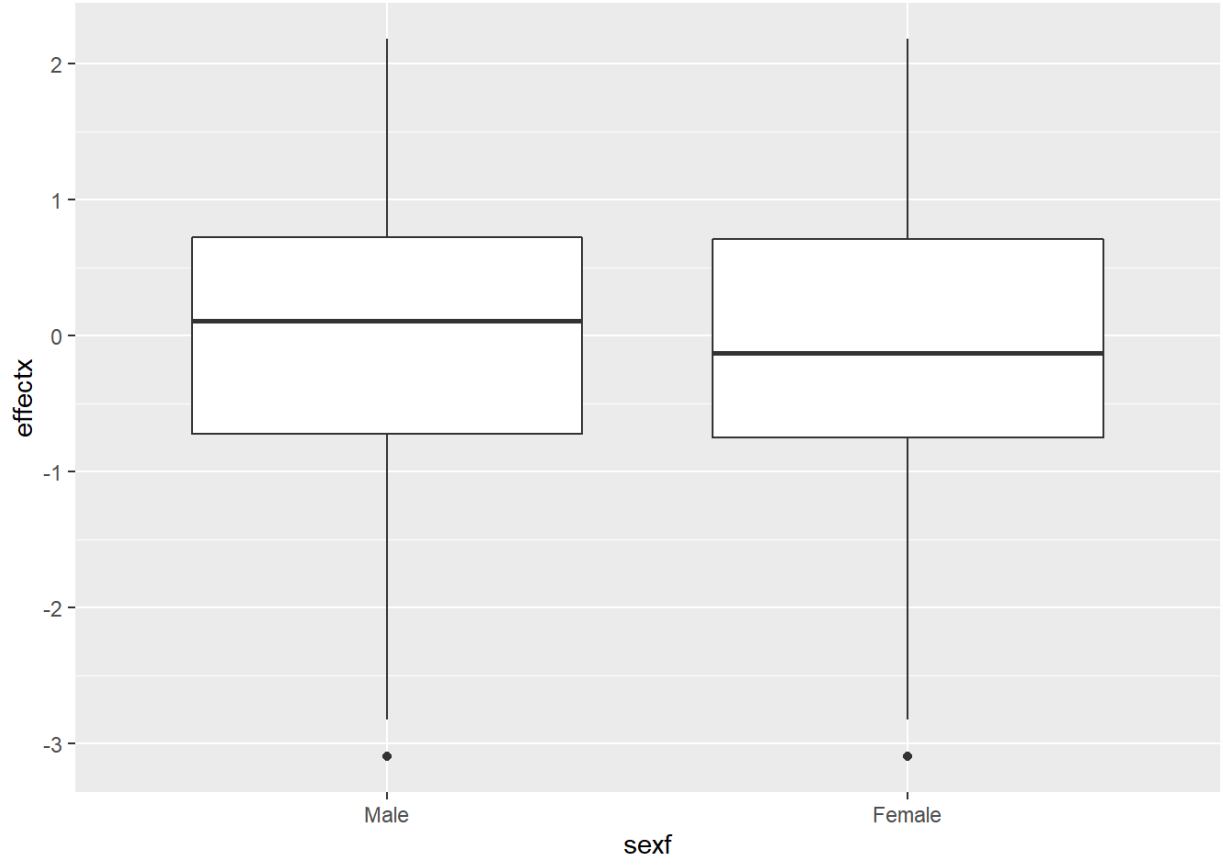
```
csew$age2<-as.numeric(csew$age)
```

You can check that we now got rid of the warning.

Categorical and continuous variables

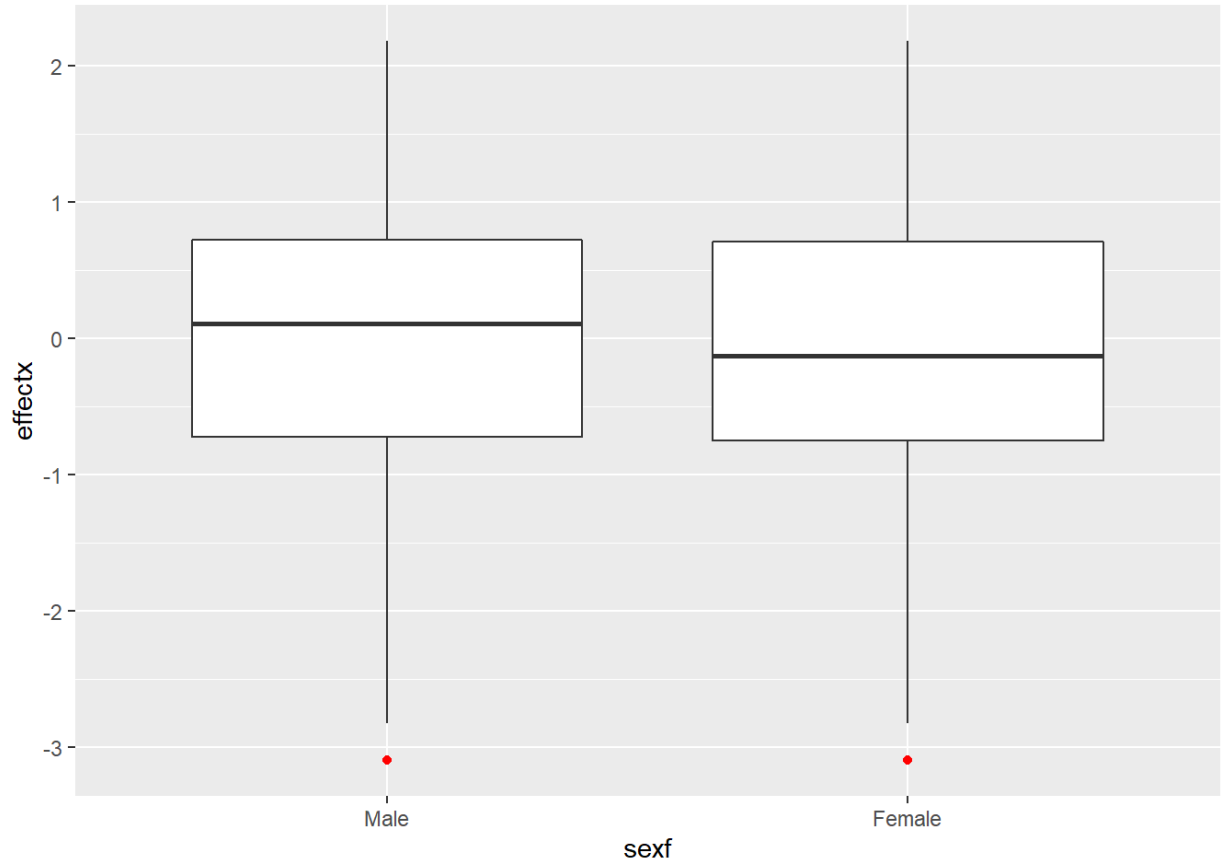
- boxplot: ideal for detecting outliers. Here we obtain boxplots by groups

```
ggplot(data = csew, mapping = aes(x = sexf, y = effectx)) +
  geom_boxplot(na.rm = TRUE)
```



We can add colours to the ouliers

```
ggplot(data = csew, mapping = aes(x = sexf, y = effectx)) +
  geom_boxplot(outlier.colour = "red", na.rm = TRUE)
```



Task 8: Explore the associations between effectiveness in the Criminal Justice System and nationality, ethnic group, victim of crime

Task 9: After all the bivariate exploration that you have done. Have you found any difference in the perception of the effectiveness in the Criminal Justice System according to:

- sex_____
  _____
- ethnic group: _____
  _____
- Victims: _____
  _____

Task 10: Can you think in any hypothesis that can help us to understand the differences in the perception of how effective the Criminal Justice System is?

5. Saving data

We can save the data we have been using, we can select different export methods, depending on the format in which we want our data to be stored. Here we will save the changes that we have made to the "csew" dataset as R data. This means that we don't need to import it again (we still need to load it) to use in another R session.

```
save(csew, file= "csew.RData")
```