Data: intro to exploratory analysis

Alla Tambovtseva

Missing values detection and patterns of NA's

Now we will work with the table containing data on famous Chilean plebiscite held in 1988 to decide whether to extend the rule of Augusto Pinochet or not. See details about variables here. We will load a table using a link:

```
df <- read.csv("http://math-info.hse.ru/f/2017-18/ps-ms/Chile.csv")
```

Let's look at the data frame using the code:

```
View(df) # V is capital
```

And explore its structure:

```
str(df)
```

```
## 'data.frame':    2700 obs. of  9 variables:
##  $ X          : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ region     : Factor w/ 5 levels "C","M","N","S",..: 3 3 3 3 3 3 3 3 3 3 ...
##  $ population: int  175000 175000 175000 175000 175000 175000 175000 175000 175000 175000 ...
##  $ sex        : Factor w/ 2 levels "F","M": 2 2 1 1 1 1 2 1 1 2 ...
##  $ age        : int  65 29 38 49 23 28 26 24 41 41 ...
##  $ education : Factor w/ 3 levels "P","PS","S": 1 2 1 1 3 1 2 3 1 1 ...
##  $ income     : int  35000 7500 15000 35000 35000 7500 35000 15000 15000 15000 ...
##  $ statusquo : num  1.01 -1.3 1.23 -1.03 -1.1 ...
##  $ vote       : Factor w/ 4 levels "A","N","U","Y": 4 2 4 2 2 2 2 2 3 2 ...
```

We can request the number of rows or columns separately:

```
nrow(df)
```

```
## [1] 2700
```

```
ncol(df)
```

```
## [1] 9
```

We can check whether rows contain missing values:

```
head(complete.cases(df)) # vector of TRUE/FALSE for all rows
```

```
## [1] TRUE TRUE TRUE TRUE TRUE TRUE
```

And count the number of complete cases, so rows with no 'NAs'.

```
sum(complete.cases(df)) # rows without NA 's
```

```
## [1] 2431
```

Although TRUE and FALSE values are logical, the sum() function works correctly since they are automatically converted into integers: 1 for TRUE and 0 for FALSE.

And how to count, vice versa, the number of rows with missing values? Simply using a negation sign before complete.cases(), which is often represented by ! in programming.

```
sum(!complete.cases(df)) # rows with NA 's
```

```
## [1] 269
```

If we want to look at the rows with NA's, we can filter them from the whole data set. Filtering is usually done by writing conditions in square brackets. At the first place we should indicate conditions for rows and at the second place - for columns. Now we need to choose rows that do not belong to complete cases, and all columns.

```
head(df[!complete.cases(df), ])
```

```
##       X region population sex age education income statusquo vote
## 13   13      N     175000   F  27        PS     NA   1.43448    Y
## 15   15      N     175000   M  36        PS  35000   1.49026 <NA>
## 28   28      N     175000   F  43         P     NA   0.15489    A
## 76   76      N     125000   F  32         S     NA  -0.85035    N
## 98   98      N     125000   F  34         P   2500   0.10807 <NA>
## 113 113      N     250000   F  46         S     NA   0.15489 <NA>
```

The second position we left blank since if we want to see all the columns, there is no need to specify them.

So as to take a close look at these rows, we can save them into a separate data frame and then view in a more convenient mode:

```
with_na <- df[!complete.cases(df), ]
```

```
View(with_na)
```

Sometimes it is important to know the patterns of missing values so as understand whether the lack of data is systematic or random. If it seems to be random, there should not be problems, but if missing values are systematic, i.e. occur in certain columns, it might be a sign of bias. For instance, if we see that there are NA's in answers to three questions simultaneously, in the same rows, it can serve as an evidence of poorly worded questions or questions on sensitive topics like money or politics.

To visualise the patterns we have to install two libraries: mice and VIM. All libraries (or packages) are installed via the function install.packages(). It is enough to install a library once. Then a library can be launched during any RStudio session.

```
# make sure your Internet connection is ok
install.packages("mice")
install.packages("VIM")
```

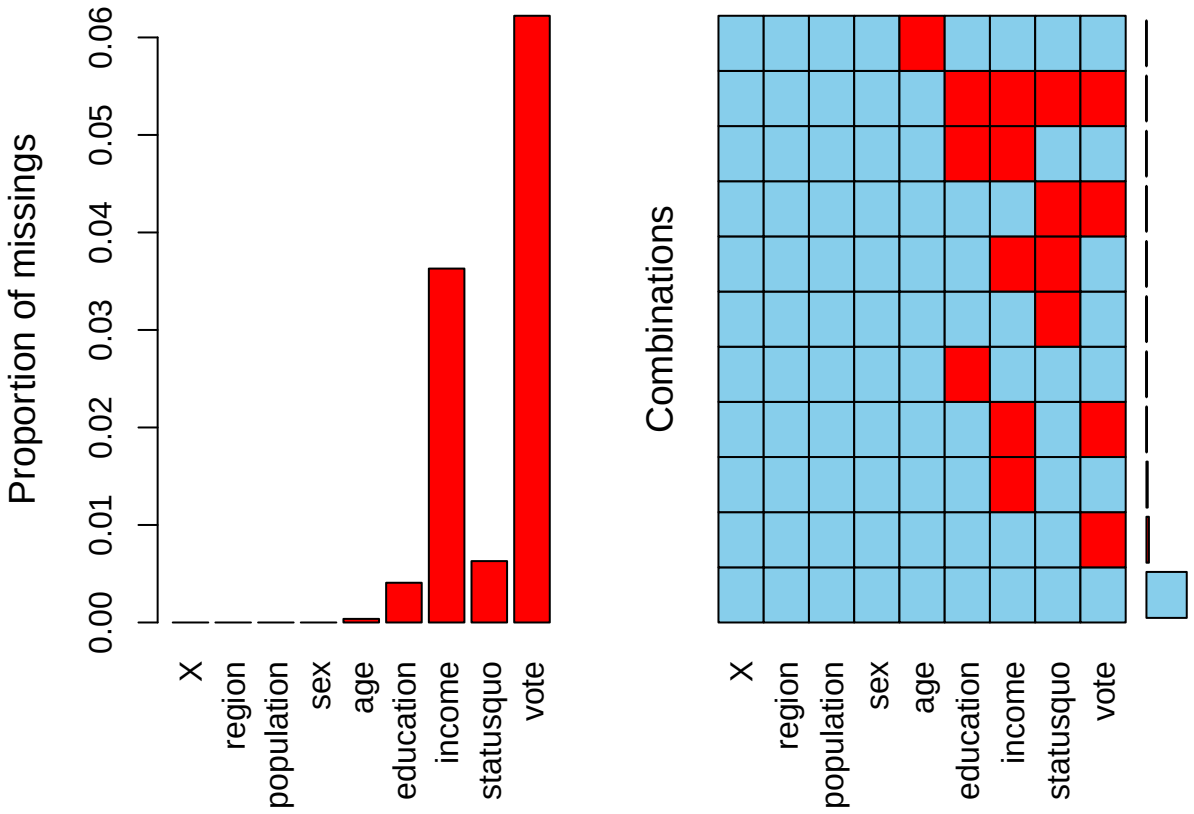
Once libraries are installed, we can use its functions. However, we usually have to launch a library first by referring to it via library(). This should be done in any new R session:

```
library(mice)
library(VIM)
```

Without these lines R will return an error Could not find function ... while typing commands from these libraries as it will not understand where to find the commands.

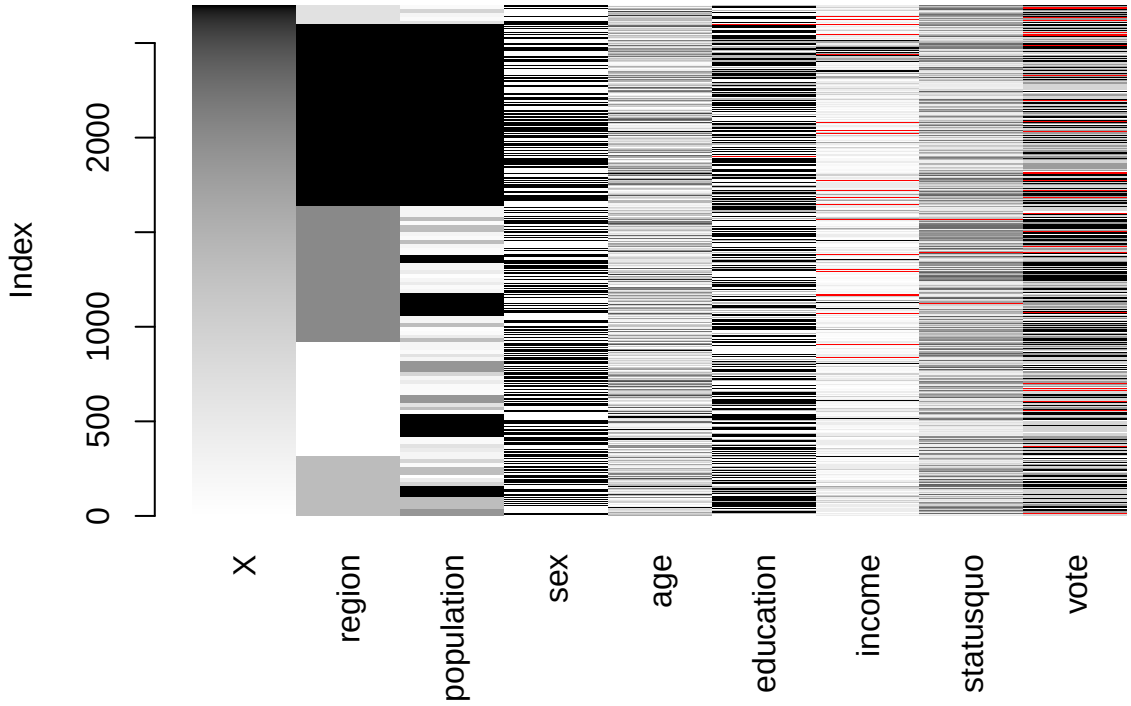
Now everything is ready to plot graphs for missing values.

```
aggr(df)
```



This graph is comprised of two parts. The left graph illustrates the frequencies of NA's in each column. Judging by this bar chart we can conclude that most missing values are concentrated in the columns vote and income. However, the percentage of NA's in vote is not very high (0.06 or 6%) if we consider the size of our data set (2700 rows). The right graph shows the patterns of missing values, the combinations of variables with NA's. From this graph we can see that people who did not indicate their income, said nothing about their intentions to vote. And they did not provide any information on their attitude towards statusquo as well. Thus, missing values in this data set are not distributed randomly, there is a certain pattern.

```
matrixplot(df)
```



Let's look at another graph. It also illustrates patterns of missing values. By y-axis go numbers of rows (indices of rows), by x-axis go variables. Cell colours range from white to black, where darker shades correspond to larger values. NA's are represented by red cells. So, if we see big red rectangles, it means that there are a lot of missing values. Besides, again we can see what columns contain NA's simultaneously.

Now for simplicity we will delete rows with missing values so as to avoid problems with further calculations and graphs:

```
df <- na.omit(df)
```

We can check:

```
sum(!complete.cases(df)) # 0 NA 's
```

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

Now let's see the statistical summary, i.e. descriptive statistics, and proceed to visualisation of variables' distribution.

```
summary(df)
```

```
##        X          region      population      sex           age
##  Min.   :   1.0   C :548   Min.   : 3750   F:1250   Min.   :18.00
##  1st Qu.: 636.5   M : 75   1st Qu.: 25000   M:1181   1st Qu.:25.00
##  Median :1326.0   N :305   Median :175000            Median :36.00
##  Mean   :1326.9   S :655   Mean   :151605            Mean   :38.29
##  3rd Qu.:2006.5   SA:848   3rd Qu.:250000            3rd Qu.:49.00
##  Max.   :2700.0            Max.   :250000            Max.   :70.00
##  education      income          statusquo         vote
##  P :1002    Min.   :  2500   Min.   :-1.72594   A:177
##  PS: 419    1st Qu.:  7500   1st Qu.:-1.00974   N:867
##  S :1010    Median : 15000   Median :-0.08924   U:551
##             Mean   : 34020   Mean   :-0.01127   Y:836
##             3rd Qu.: 35000   3rd Qu.: 0.96969
```

```
##                Max.   :200000    Max.   : 1.71355
```

Nominal variables: visualising a distribution

First of all, for a nominal variable we can make a frequency table. For instance, we will take the variable vote and count the number of occurrences of each unique value of vote. To choose a variable from df, we need a dollar sign ($):

```
table(df$vote)
```

```
##
##   A   N   U   Y
## 177 867 551 836
```

By default R computes absolute frequencies, but be can get relative ones by hand:

```
table(df$vote)/sum(table(df$vote)) # shares, fractions
```

```
##
##          A          N          U          Y
## 0.07280954 0.35664336 0.22665570 0.34389140
```

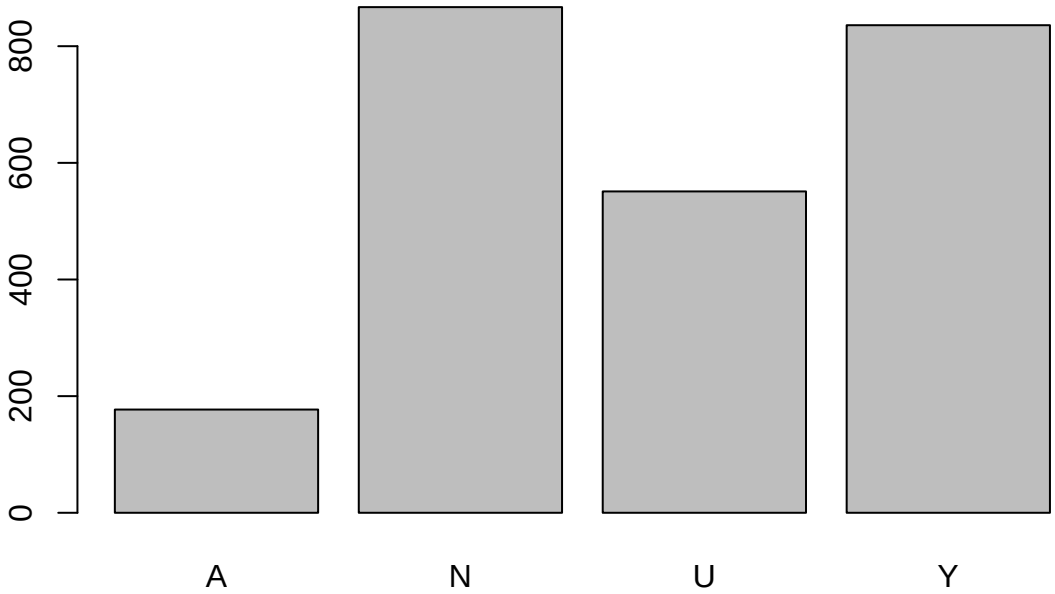
```
table(df$vote)/sum(table(df$vote)) * 100 # in percent (%)
```

```
##
##         A         N         U         Y
##  7.280954 35.664336 22.665570 34.389140
```

```
res <- table(df$vote)/sum(table(df$vote)) * 100
round_res <- round(res, 2) # round to the 2nd digit after a point
```

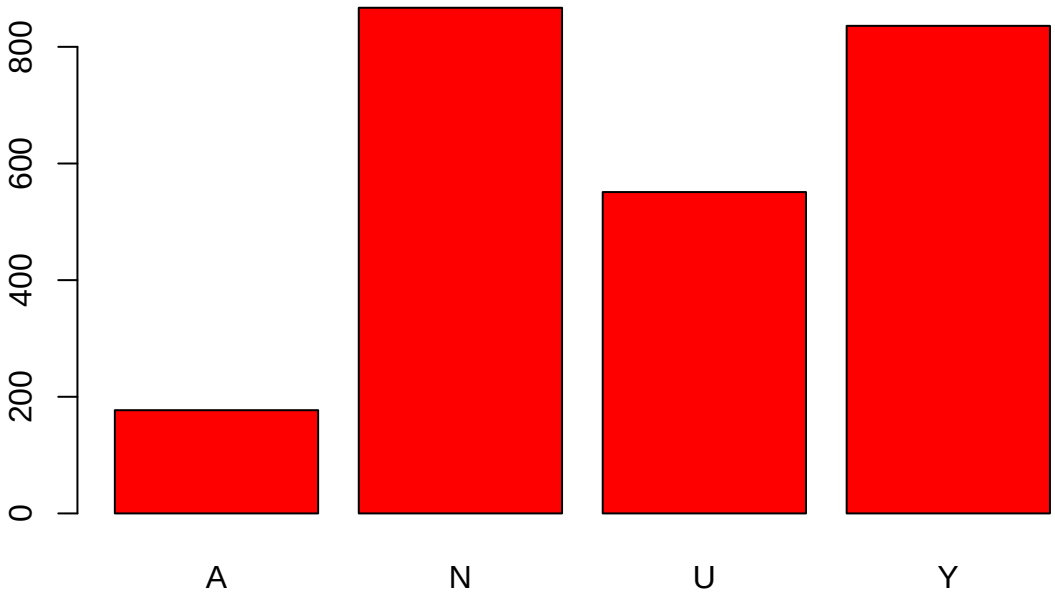
Secondly, now we can plot a basic R bar chart for this variable:

```
barplot(table(df$vote))
```



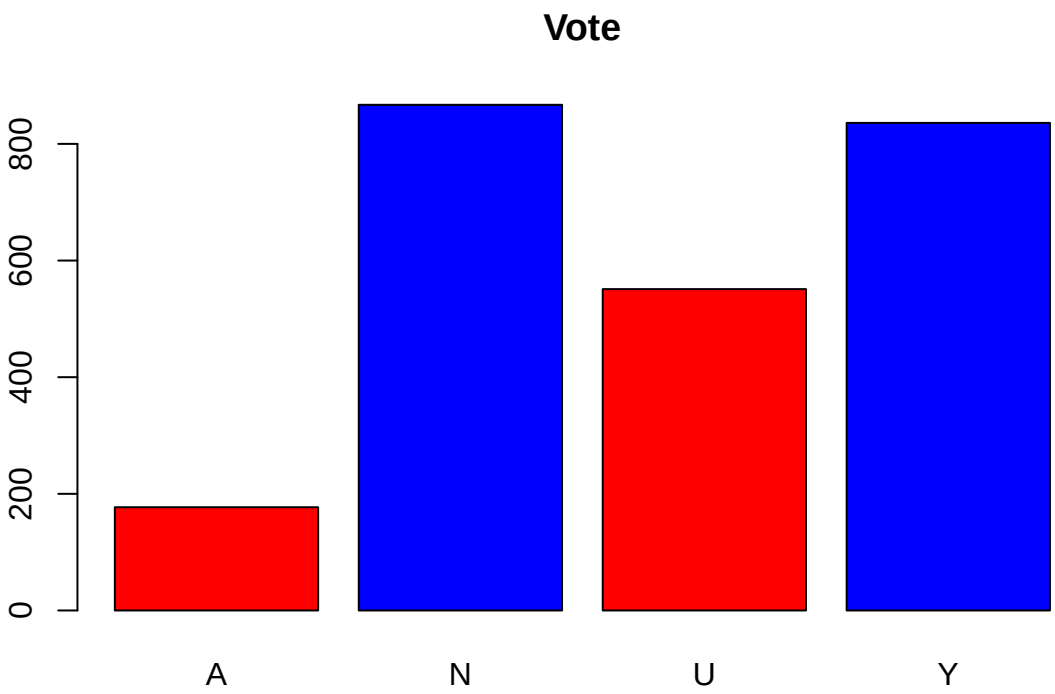
A basic plot is grey and dull. Let's change its colour:

```
barplot(table(df$vote), col = "red")
```



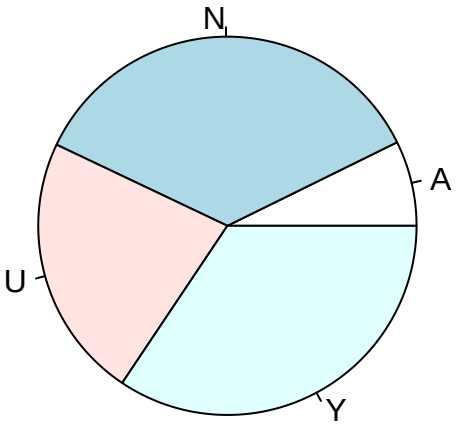
R has a very rich pallete of colours, see the full catalogue here. Most colours have specific non-standard names, but by default R recognizes them all. Besides, if we want to get columns of different colours, we can pass a vector of their names as an argument col:

```
barplot(table(df$vote),
        col = c("red", "blue", "red", "blue"),
        main = "Vote") # add a title as well
```



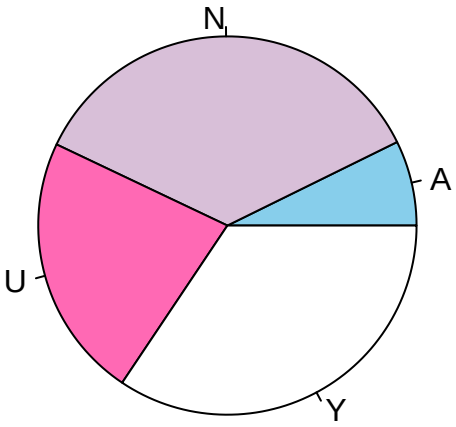
Thirdly, we can create a pie chart with frequencies in %:

```
pie(table(df$vote))
```



Not an aesthetic thing, though. But we might change default colours:

```
cols2 <- c("skyblue", "thistle", "hotpink", "grey100")
pie(table(df$vote), col = cols2)
```

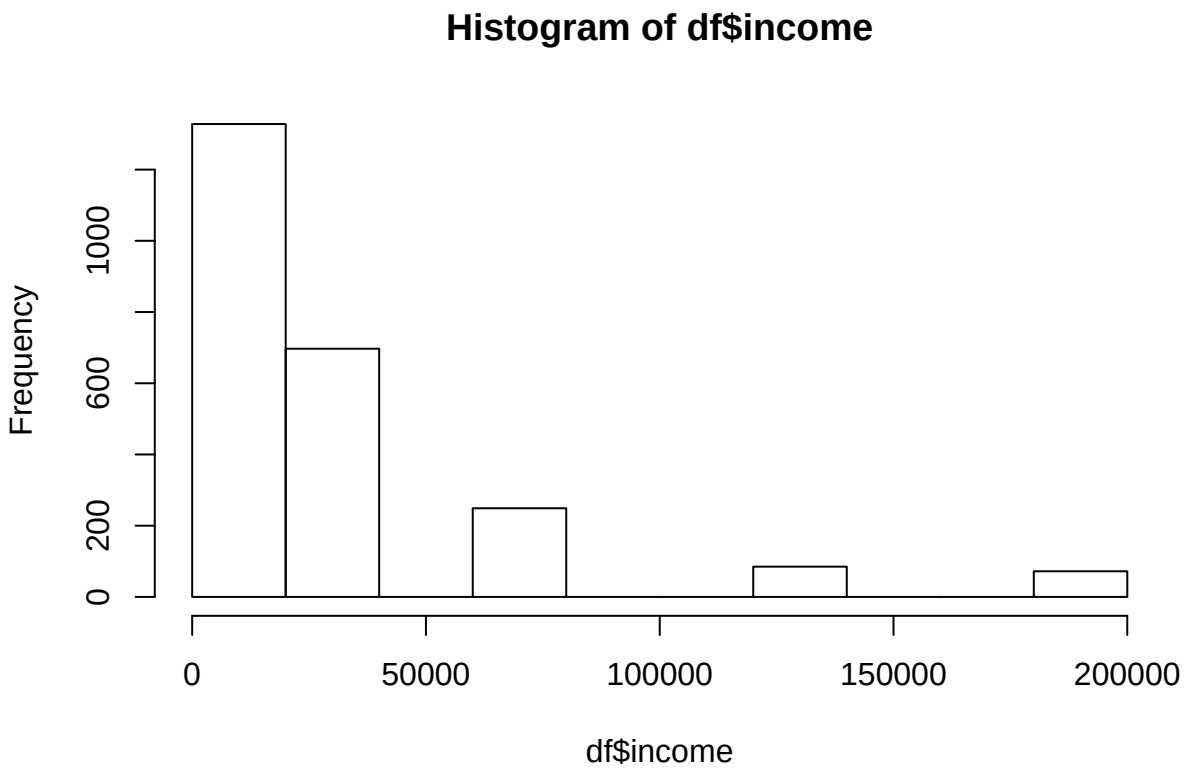


Quantitative variables: visualising a distribution

For a quantitative variable we can plot a histogram. Although a histogram sometimes resembles a bar chart, there is a principal difference between them: a histogram cannot be created for nominal data since it is impossible to arrange nominal values in an ascending order.

Let's plot a histogram for respondents' income:

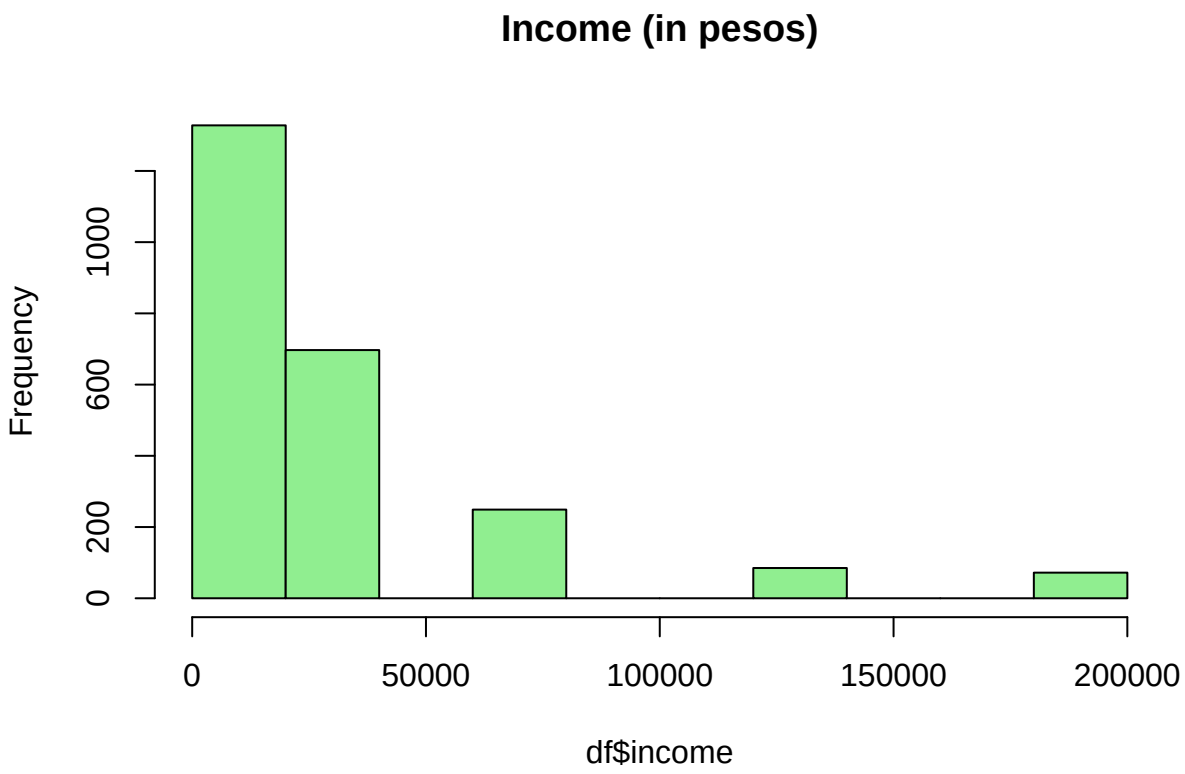
```
hist(df$income)
```



As we can see, 1) most people in our table earn less than 50000 pesos; 2) there is a small group of respondents that are much richer than others and earn more than 180000 pesos. The distribution of income is right-skewed: there is a long right tail.

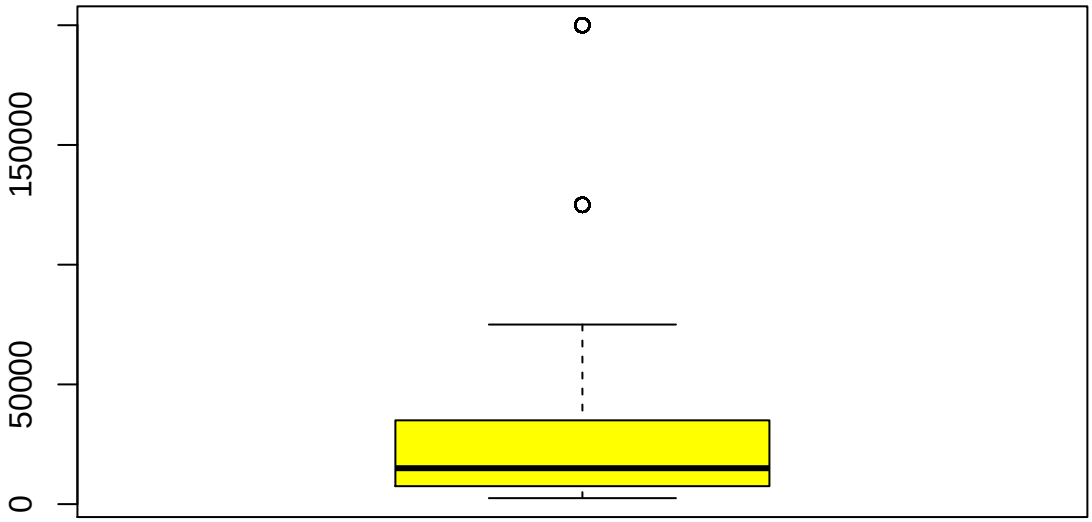
We can make our graph more beautiful:

```
hist(df$income, col = "lightgreen",
     main = "Income (in pesos)")
```



Another way to summarise descriptive statistics of a variable and visualise them is to create a box plot:

```
boxplot(df$income, col = "yellow")
```



From this picture it is clear why this type of graph is called a box plot (box-and-whiskers plot). There is a "box" and there are two "whiskers", lower and upper one.

How is this graph plotted? Horizontal lines that contours a "box" are lower and upper quartiles, and a thick horizontal line inside a box is a median. A median should not be exactly in the middle of a box, it depends on data, so it can be closer either to a lower quartile or to an upper quartile.

So as to understand what whiskers stand for, let us describe how to detect non-typical values of a variable. Typical values of a variable belong to the following interval:

$$[Q_1 - 1.5 \times IQR;\ Q_3 + 1.5 \times IQR]$$

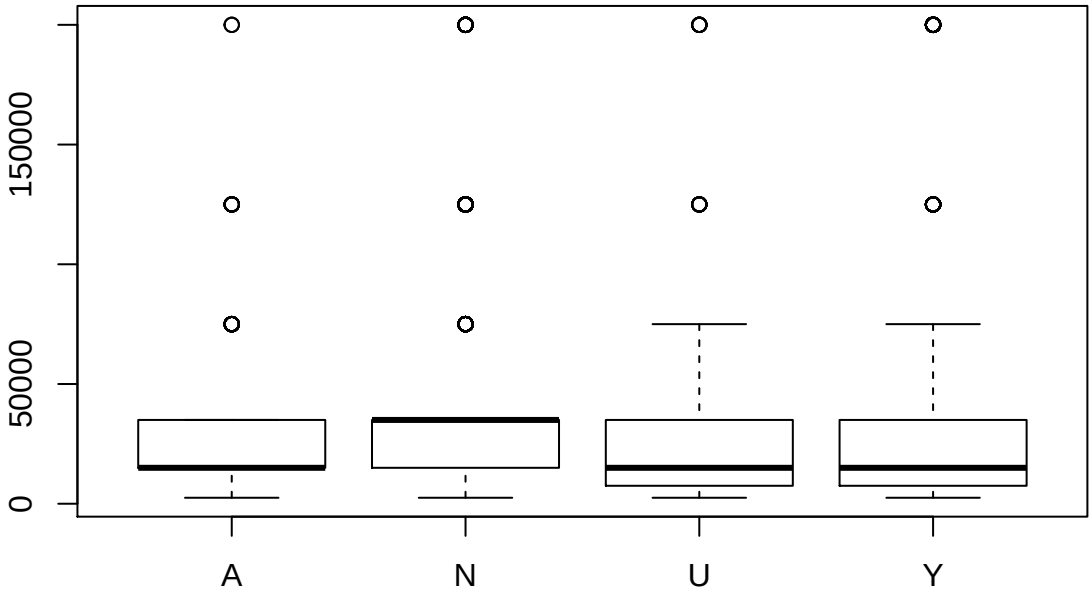
Values that fall behind this interval are non-typical and called outliers. In a box plot outliers are marked as dots that lie outside whiskers. Endpoints of whiskers, however, are defined in different ways:

- if there are no outliers on the lower side, the endpoint of a lower whisker corresponds to the minimum value;
- if there are no outliers on the upper side, the endpoint of an upper whisker corresponds to the maximum value;
- if there are outliers on either side, the endpoints of whiskers correspond to $Q_1 - 1.5 \times IQR$ or $Q_3 + 1.5 \times IQR$.

Thus, in our case we see that there two outliers that refer to people with very high income. Two outliers usually mean two objects, e.g. two people, but sometimes one outlier might correspond to several objects (values coincide and point overlap). Further we will learn how to calculate a number of outliers and how to know which rows include non-typical values.

Now let's look how we can use box plots to compare distributions. We will plot graphs by groups based on values of vote. In other words, we will compare the income distribution of people who voted for or against Pinochet, intended to abstain or were undecided. When we want to plot graphs by groups, we need to add a tile ~ and indicate the grouping variable.

```
boxplot(df$income ~ df$vote)
```



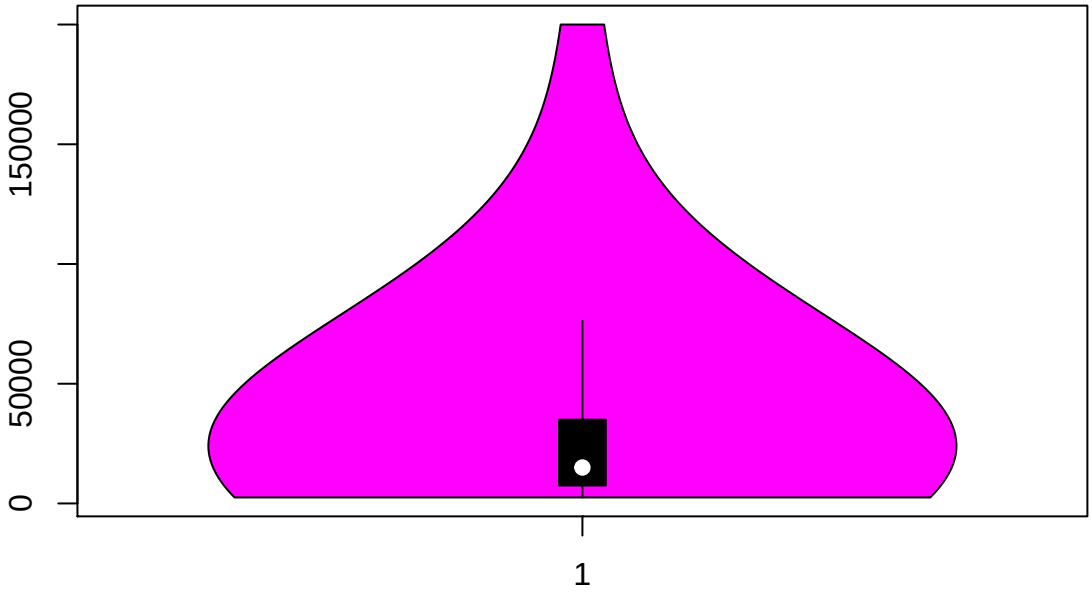
Interestingly, but the shape of income distribution of undecided respondents and respondents for Pinochet is the same! People who intended to vote against, in general, are less rich than those who wanted to vote for. Respondents who were undecided or for Pinochet are more diverse in their income rather than ones prone to abstain or vote against (box plots for the latter are more narrow, with no upper whiskers at all). One more remarkable thing: the presence of very rich people is not a distinctive feature of a particular group. In all groups there are chileans with high income.

At the end of this lecture we have some time to learn one more type of graphs that are called violin plots or bean plots (some people see violins, other - beans, nothing special). To create a basic violin plot, we have to install the package vioplot.

```
install.packages("vioplot")
```

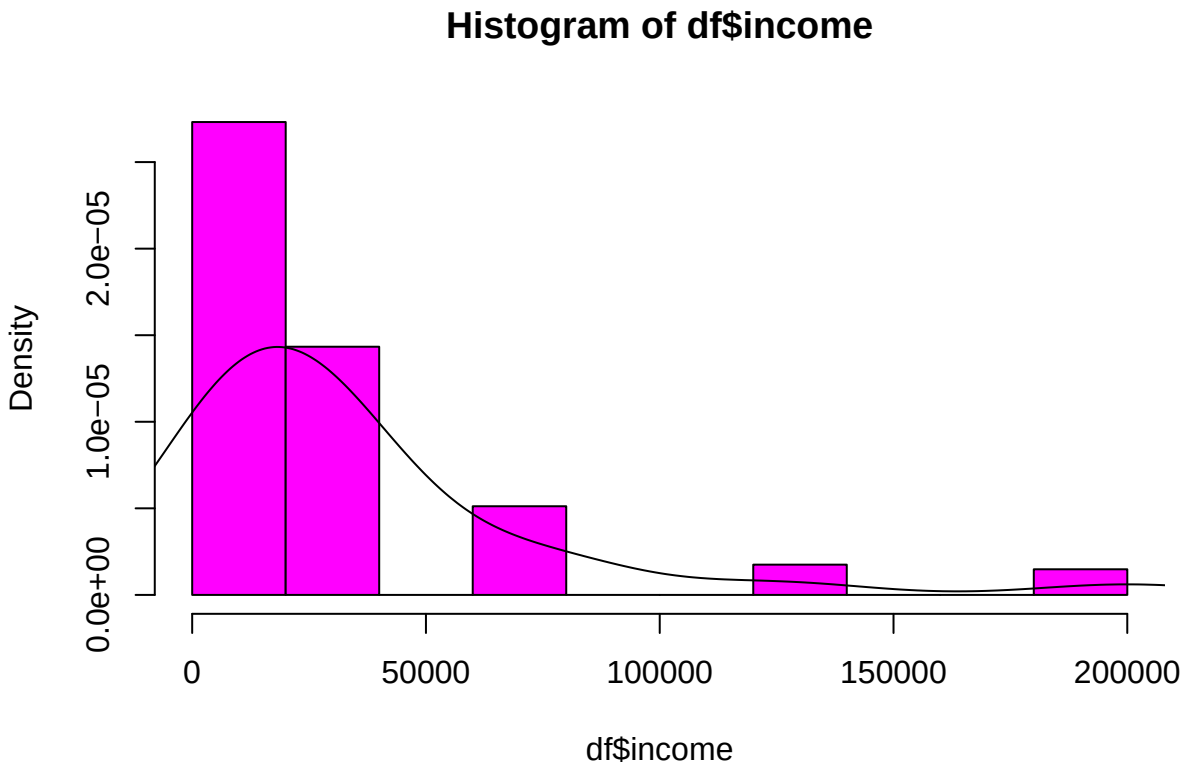
Now we can plot a violin plot for the variable income:

```
library(vioplot)
vioplot(df$income)
```

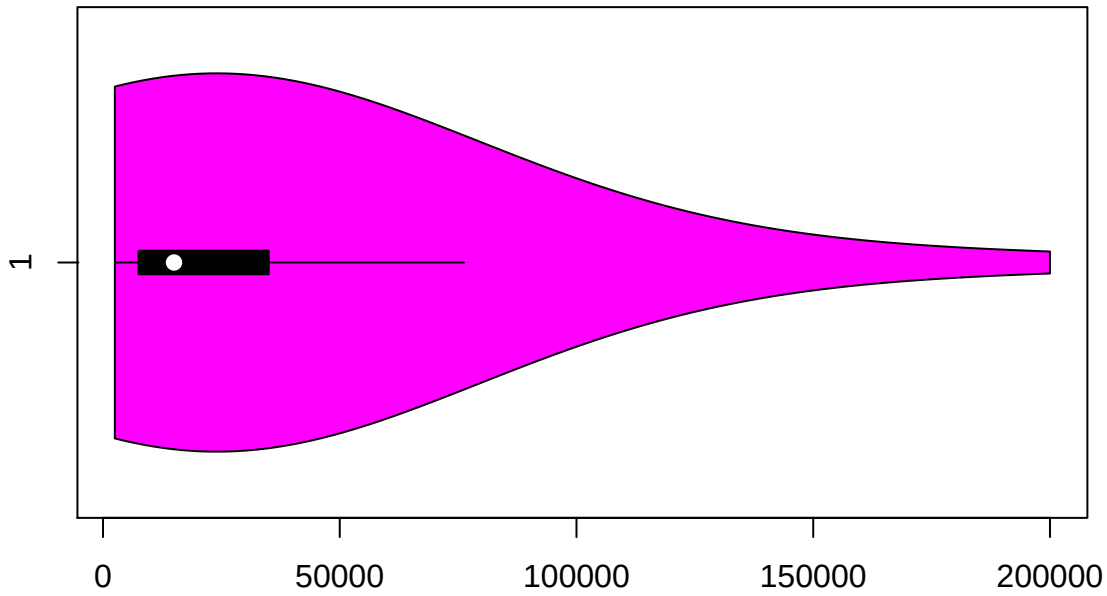


This plot might seem to be strange if you have not seen it before, but it is still useful. It shows the shape of a variable distribution in a more proper way than a box plot, and it can be regarded as a combination of a histogram and a box plot. If we divide it into two symmetric halves, we will see that each half is just a smoothed histogram turned by 90 degrees (from the horizontal mode to a vertical one). If it is not clear, you can compare:

```
hist(df$income, col="magenta", prob = TRUE) # probabilities instead of frequencies
lines(density(df$income, adjust=5)) # adjust - smoothing parameter
```




```
vioplot(df$income, horizontal = TRUE) # horizontal
```



In the center of a "violin" there is a small box plot with a white dot corresponding to the mean value. Judging by this graph we can conclude that the distribution of income is right-skewed (long tail on the right), and the lower values prevail. Next time we will discuss when it is more helpful to use a violin plot rather than a box plot and what are the assumptions inherited in each type of graph.