Linguistic Data: Quantitative Analysis and Visualisation

Ilya Schurov, Olga Lyashevskaya, George Moroz, Alla Tambovtseva

19 January 2019

Working with data frames

Data loading

Now we will work with a csv-file. CSV stands for *comma separated values*, so it is a text file where columns are separated with a comma, like this:

```
a,b,c
1,5,2
0,4,3
```

At first, we will load a csv-file via a link. To do this we need the function `read.csv()`. And then we put the link into brackets (don't forget quotes):

```
dat <- read.csv("http://math-info.hse.ru/f/2018-19/ling-data/Chi.kuk.2007.csv")
```

Of course, in real life we do not have such links and load data from our laptops. So as not to spend more time on discussing working directories and writing paths to files, let's consider an interative function `file.choose()` that will ask to choose a file from a folder:

```
dat2 <- read.csv(file.choose())
```

So, it works like many other programs that suggest us to choose a file for working.

Look at our data in a convenient way:

```
View(dat)
```

This file contains data on the following research (description is taken from here).

The majority of examples in that presentation are based on Hau 2007. Experiment consisted of a perception and judgment test aimed at measuring the correlation between acoustic cues and perceived sexual orientation. Naïve Cantonese speakers were asked to listen to the Cantonese speech samples collected in Experiment and judge whether the speakers were gay or heterosexual. There are 14 speakers and following parameters:

- [s] duration (`s.duration.ms`)
- vowel duration (`vowel.duration.ms`)
- fundamental frequencies mean (F0) (`average.f0.Hz`)
- fundamental frequencies range (`f0.range.Hz`)
- percentage of homosexual impression (`perceived.as.homo`)
- percentage of heterosexal impression (`perceived.as.hetero`)
- speakers orientation (`orientation`)
- speakers age (`age`)

General information about data frames

When data are loaded, we can look at general info using `str()` function:

```
str(dat)
```

```
## 'data.frame':    14 obs. of  10 variables:
##  $ speaker                 : Factor w/ 14 levels "A","B","C","D",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ s.duration.ms           : num  61.4 63.9 55.1 78.1 64.7 ...
##  $ vowel.duration.ms       : num  112.6 126.5 126.8 119.2 93.7 ...
##  $ average.f0.Hz           : num  120 100 115 127 131 ...
##  $ f0.range.Hz             : num  52.5 114 103.2 58.8 37.4 ...
##  $ perceived.as.homo       : int  7 20 9 15 10 17 20 21 20 8 ...
##  $ perceived.as.hetero     : int  18 5 16 10 15 8 5 4 5 17 ...
##  $ perceived.as.homo.percent: num  0.28 0.8 0.36 0.6 0.4 0.68 0.8 0.84 0.8 0.32 ...
##  $ orientation             : Factor w/ 2 levels "hetero","homo": 1 1 2 2 2 2 1 1 2 2 ...
##  $ age                     : int  30 19 29 36 27 33 28 22 22 40 ...
```

This function is very helpful since it returns a lot of information at the same time: number of observations (rows), number of variables (columns), names of all columns, their types and first values in each column.

To get descriptive statisitics for all columns, we will need `summary()` function:

```
summary(dat)
```

```
##     speaker  s.duration.ms   vowel.duration.ms average.f0.Hz
##  A      :1   Min.   :45.13   Min.   : 93.68    Min.   :100.3
##  B      :1   1st Qu.:58.15   1st Qu.:118.31    1st Qu.:116.0
##  C      :1   Median :61.93   Median :123.75    Median :122.7
##  D      :1   Mean   :61.22   Mean   :124.06    Mean   :125.2
##  E      :1   3rd Qu.:64.51   3rd Qu.:132.27    3rd Qu.:130.3
##  F      :1   Max.   :78.11   Max.   :147.52    Max.   :155.3
##  (Other):8
##   f0.range.Hz     perceived.as.homo perceived.as.hetero
##  Min.   : 37.40   Min.   : 4.00     Min.   : 4.00
##  1st Qu.: 53.30   1st Qu.: 8.25     1st Qu.: 5.00
##  Median : 73.20   Median :12.50     Median :12.50
##  Mean   : 76.66   Mean   :13.50     Mean   :11.50
##  3rd Qu.:102.53   3rd Qu.:20.00     3rd Qu.:16.75
##  Max.   :118.20   Max.   :21.00     Max.   :21.00
##
##  perceived.as.homo.percent orientation      age
##  Min.   :0.16              hetero:7    Min.   :19.00
##  1st Qu.:0.33              homo  :7    1st Qu.:22.75
##  Median :0.50                          Median :28.50
##  Mean   :0.54                          Mean   :27.86
##  3rd Qu.:0.80                          3rd Qu.:30.00
##  Max.   :0.84                          Max.   :40.00
##
```

For numeric columns it returns descriptive statistics, for character or factor ones it returns absolute frequencies.

To get the number of observations in a data frame, we can use `nrow()` function:

```
nrow(dat)
```

```
## [1] 14
```

Selection of columns and rows

Let's choose a column by its name. Take `age`, for example:

```
# $ and then the name of a column
dat$age
```

```
##  [1] 30 19 29 36 27 33 28 22 22 40 30 25 20 29
```

Then we can work with a separate column and R will treat it as a vector:

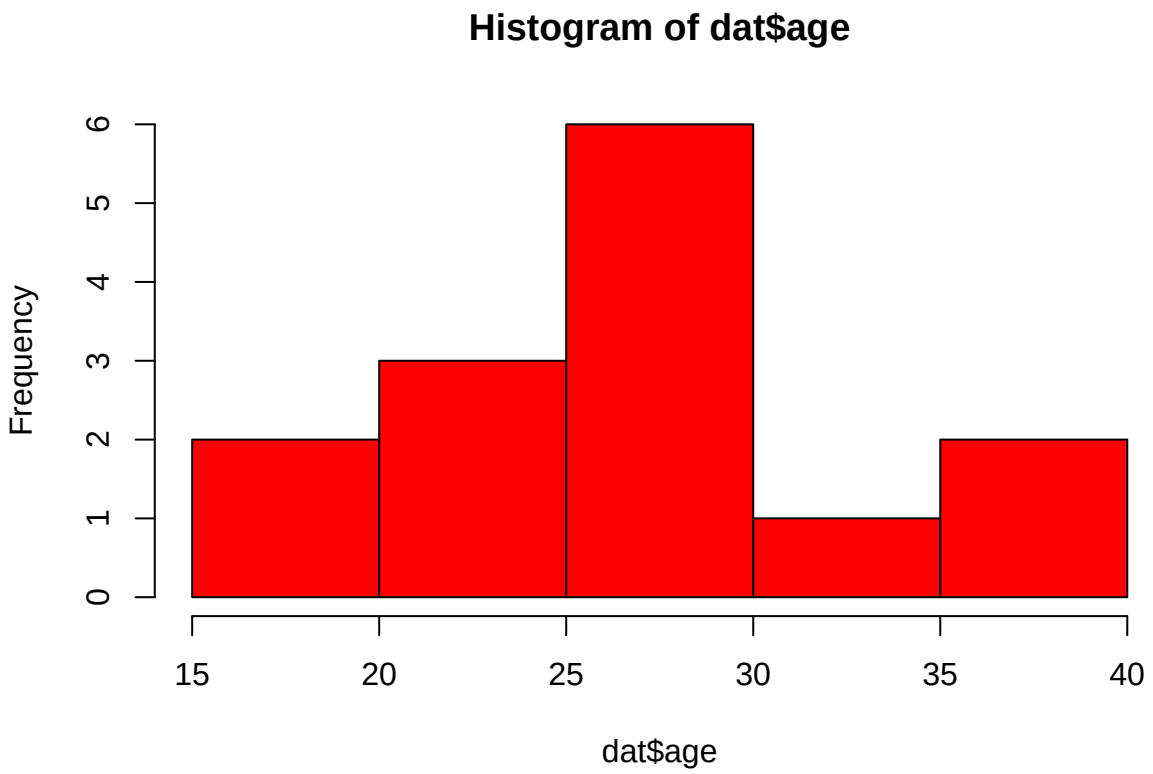
```
# summary of a column
summary(dat$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   19.00   22.75   28.50   27.86   30.00   40.00
```

```
# average age
mean(dat$age)
```

```
## [1] 27.85714
```

```
# histogram for a column
hist(dat$age, col="red")
```



Now let's see how to choose rows and columns by their index (position in a data frame). As a data frame is a table with rows and columns, to choose a certain cell from this table we have to specify the row number and the column number. In R row numbers usually go first:

```
# 1st row, 2nd column
dat[1, 2]  # 1st speaker, 2nd column - s.duration.ms
```

```
## [1] 61.4
```

So as to select the 1st row and all the columns, so, all the characteristics of the 1st speaker, we should leave the second position blank:

```
dat[1,]  # type nothing after ,
```

```
##   speaker s.duration.ms vowel.duration.ms average.f0.Hz f0.range.Hz
## 1       A          61.4             112.6        119.51        52.5
##   perceived.as.homo perceived.as.hetero perceived.as.homo.percent
## 1                 7                  18                      0.28
##   orientation age
## 1      hetero  30
```

The same can be done for columns:

```
# all rows, only the 3rd column
dat[,3]
```

```
##  [1] 112.60 126.49 126.81 119.17  93.68 127.87 147.52 120.13 140.44 121.01
## [11] 137.37 112.05 133.74 118.02
```

Filtering

Of course, in practice we often have to filter our data based on some conditions rather than choose specific columns or rows by their indices. The logics of filtering rows of a data frame is the same as in vectors: we have to write the condition in square brackets.

```
# take all speakers older than 32 year old
dat[dat$age > 32,]
```

```
##    speaker s.duration.ms vowel.duration.ms average.f0.Hz f0.range.Hz
## 4        D         78.11             119.17        126.61        58.8
## 6        F         67.00             127.87        150.79        42.0
## 10       J         59.59             121.01        123.90       111.7
##    perceived.as.homo perceived.as.hetero perceived.as.homo.percent
## 4                 15                  10                      0.60
## 6                 17                   8                      0.68
## 10                 8                  17                      0.32
##    orientation age
## 4         homo  36
## 6         homo  33
## 10        homo  40
```

However, now we should not forget that our condition is tested for rows, so it should be placed before a comma. If we miss this comma, we will get incorrect results:

```
# all the speakers, of any age!
dat[dat$age > 32]
```

```
##    average.f0.Hz perceived.as.homo age
## 1         119.51                 7  30
## 2         100.29                20  19
## 3         114.90                 9  29
## 4         126.61                15  36
## 5         130.76                10  27
## 6         150.79                17  33
## 7         128.96                20  28
## 8         105.26                21  22
## 9         109.86                20  22
## 10        123.90                 8  40
```

```
## 11        119.48                21  30
## 12        146.20                 8  25
## 13        155.34                 9  20
## 14        121.48                 4  29
```

We can test multiple conditions at the same time. For example, let's choose speakers older than 20 year old that are homosexual:

```
# & - at the same time
dat[dat$age > 20 & dat$orientation == "homo",]
```

```
##    speaker s.duration.ms vowel.duration.ms average.f0.Hz f0.range.Hz
## 3        C         55.08            126.81        114.90       103.2
## 4        D         78.11            119.17        126.61        58.8
## 5        E         64.71             93.68        130.76        37.4
## 6        F         67.00            127.87        150.79        42.0
## 9        I         60.45            140.44        109.86        96.4
## 10       J         59.59            121.01        123.90       111.7
## 11       K         62.94            137.37        119.48        87.6
##    perceived.as.homo perceived.as.hetero perceived.as.homo.percent
## 3                  9                  16                      0.36
## 4                 15                  10                      0.60
## 5                 10                  15                      0.40
## 6                 17                   8                      0.68
## 9                 20                   5                      0.80
## 10                 8                  17                      0.32
## 11                21                   4                      0.84
##    orientation age
## 3         homo  29
## 4         homo  36
## 5         homo  27
## 6         homo  33
## 9         homo  22
## 10        homo  40
## 11        homo  30
```

Again to join these conditions we use `&` that stands for *simultaneously true* (recall working with vectors). Now let's calculate how many homosexuals older than 20 are in our data frame:

```
nrow(dat[dat$age > 20 & dat$orientation == "homo",])
```

```
## [1] 7
```

To do this, we use `nrow()` function. Why we do not need `length()`?

```
length(dat[dat$age > 20 & dat$orientation == "homo",])
```

```
## [1] 10
```

As R treats any data frame as a vector of vectors, `length()` function returns number of elements, so number of vectors in the data frame. Any column is a vector, so we obtain 10 here that is the number of columns.

As we have seen, looking at data that appear in the console is not convenient. So, we can save a subset of a data frame in a variable and then use `View()` to see it in a separate tab:

```
dat_small <- dat[dat$age > 20 & dat$orientation == "homo",]
View(dat_small)
```

Now let's suggest criteria you are interested in and we will choose speakers that satisfy these criteria.

Suggestion 1

Choose homosexuals that are mostly perceived as homosexuals.

Note: we decided that *mostly perceived as homosexulas* are speakers with the proportion of listeners assigned them to homosexuals is more than 0.5.

```
homo <- dat[dat$perceived.as.homo.percent > 0.5 &
      dat$orientation == "homo",]
View(homo)
```

Now let's calculate the percentage of homosexuals perceived as homosexuals.

```
nrow(homo)/nrow(dat) * 100
```

```
## [1] 28.57143
```

Suggestion 2

Choose speakers who are mostly perceived as homosexuals with either the intonation greater than the average or with vowel duration greater than the average.

First, to avoid overloading, we can calculate the average of vowel duration and the average of intonation (`f0`):

```
# save
mean_duration <- mean(dat$vowel.duration.ms)
mean_intonation <- mean(dat$average.f0.Hz)
```

Then we can choose rows needed:

```
homo2 <- dat[(dat$vowel.duration.ms > mean_duration |
      dat$average.f0.Hz > mean_intonation) &
      dat$perceived.as.homo.percent > 0.5, ]
nrow(homo2)
```

```
## [1] 6
```

Note: the structure of our conditions is the following: `(cond1 | cond2) & cond3`. So, `cond3` should be true all the time (choose only homosexuals) while between `cond1` and `cond2` at least one should be true. In other words, our filter captures the following situations:

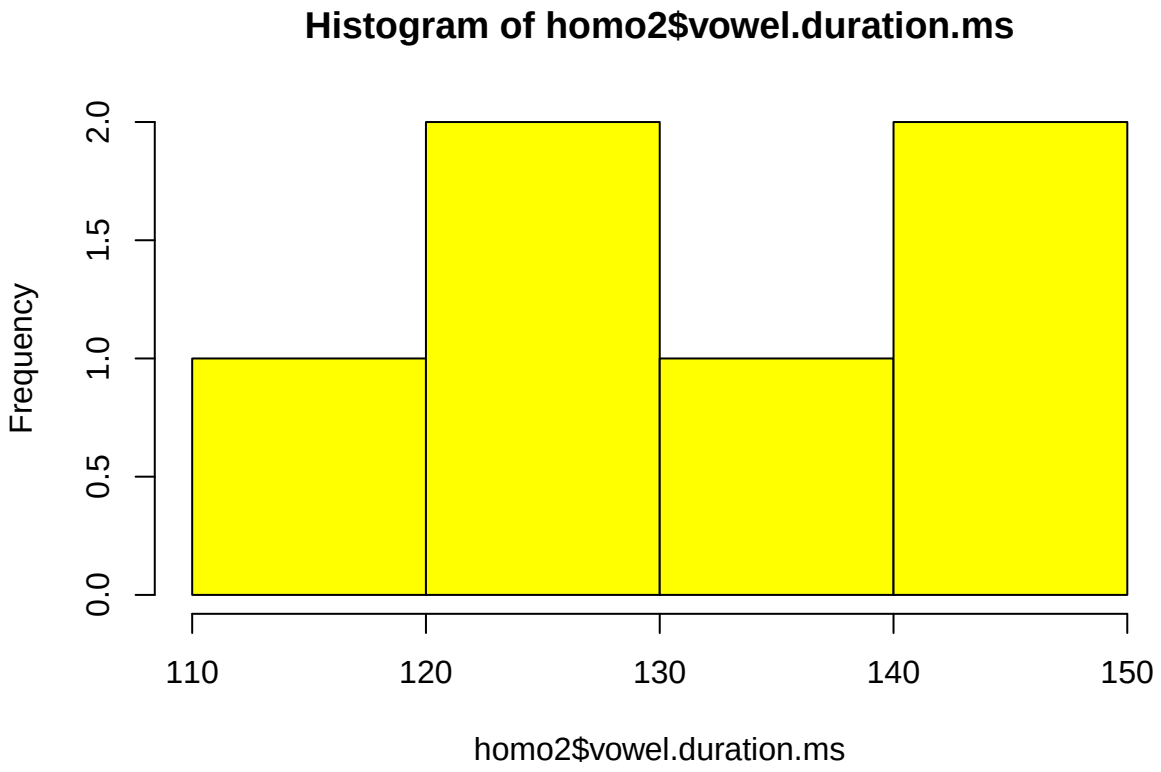
- homosexuals that have vowel duration greater than the average duration
- homosexuals that have intonation greater than the average intonation
- homosexuals that have both vowel duration and intonation greater than the average.

One more crucial point: brackets `()` here are compulsory; without them we will get a different condition, like this: `cond1 | (cond2 & cond3)`. This is because operator `&` is stronger.

Suggestion 3

Plot a histogram of vowel duration for speakers chosen at the previous step:

```
hist(homo2$vowel.duration.ms, col="yellow")
```


Histogram of homo2$vowel.duration.ms
Frequency
0.0
0.5
1.0
1.5
2.0
110
120
130
140
150
homo2$vowel.duration.ms