Data analysis with R

Lecture 3

Jouni Junnila

Example of sapply

- With *sapply* we can calculate eg. mean for different elements of a list with single statement.

```
l <- list(a = 1:10, b = 11:20)
sapply(l, mean)
a    b
5.5 15.5
```

Example of tapply

- The *tapply* function is simple to use. First, we will generate some data.
- `clinical <- data.frame(patient = 1:100, age = rnorm(100, mean = 60, sd = 12), treatment = gl(2, 50, labels = c("Treatment", “Placebo")))`
- `tapply(clinical$age, clinical$treatment, mean)`

```
Treatment Control
56.53683  59.77712
```

by()

- For calculation in subgroups, we can also use *by()* – function. This well end up in the same result as *tapply*.

```
by(iris[, 1:4], Species, mean)
Species: setosa
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.006        3.428        1.462        0.246
------------------------------------------------------------
Species: versicolor
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       5.936        2.770        4.260        1.326
------------------------------------------------------------
  Species: virginica
Sepal.Length  Sepal.Width Petal.Length  Petal.Width
       6.588        2.974        5.552        2.026
```

R-dates

- Calendar dates can be represented in R using objects of class 'Date'.
- You can convert character representation of dates to objects of class 'Date' by using the function *as.Date()*.
- Default format for the character representation is "2010-11-08“.
- With *date()* we can get the current date and time.

R-dates (2)

- Dates can be inserted also in different formats than ‘2010-11-07’.
  - We need to use format-option in the as.Date-function for this to happen.
    - `as.Date(“07/11/2010”,format="%d/%m/%Y“)`
- Dates are used in data analysis quite often, eg. in
  - Time to event calculations (eg. survival analysis)
  - Age, duration of exposure, time on study, etc...
  - Time series

Sampling

- With R we can do simple random sampling easily.
- Lets consider a deck of cards. We can select 5 of them randomly by typing:

```
>sample(52,5,replace=F)
[1] 32 34 31 14 29
```

- With the replace-option we can choose, do we return the card to the deck or not.

cbind & rbind

- *cbind()* takes a sequence of vector, matrix or data frames arguments and combines them by columns.
  - *cbind* is useful eg. when we have our data stored in vectors and we want to construct a data-frame of them.
- *rbind()* takes a sequence of vector, matrix or data frames arguments and combines them by rows.
  - An example of the use *rbind():* we have several different data entries from eg. different people and we want to combine all of them in a single data-frame

cbind & rbind (2)

- With *cbind()* R requires an even amount of observations (rows) in the vectors we are combining
- Likewise with *rbind()* we need to have an even amount or columns, for *rbind()* to work.
    - This can be problematic in some cases as we might have a subset of data with less variables than the other.
        - By creating eg. dummy (empty) variables we can surpass this problem.
- These functions are simple and effective to use and thus they are heavily used along R-programmers.

Merging datasets

- With previous function *cbind()* we cannot have an id-variable when we combine the vectors.
  - Usually though, when we are merging datasets, we have to consider an ID-variable (eg. patient number), so that we can be sure that the data goes to the right invidual.
- In cases like this, we can use a function called *merge()*.

Merging datasets (2)

- With *merge*-function we have to first naturally tell R, what are the datasets we want to merge and then the Id-variable, which does not have to be called the same in the two datasets.
  - Let's consider two datasets a and b. In dataset a we have an ID called PATID and in b an ID called PATNO

```
merge(a,b, by.x="PATID", by.y="PATNO")
```

  - The code above will do the merging for us. If the ID-variables are called the same we can use just *by=id-variable*

Merging datasets (3)

- With *all.x* and *all.y* options in the *merge*-function we can determine do we want append observations from the datasets that doesn't have a pair in the other datasets.
  - NAs will be printed into the columns of the dataset, where a certain ID is missing.
- Data is practically always stored in more than one place, so data analyst has to do quite a lot of merging.

Vocabulary for variables

- During the course we use a set of adjectives to determine the attributes of a single variable.
  - Integer variables.
    - These variables are numbers, where there is now decimals. Eg. Number of children
  - Numeric variables.
    - These variables can get whatever number value
  - Categorical variables
    - These variables have some pre-defined classes and there cannot be any other values.

Vocabulary for variables (2)

- For example gender, age-category, etc.
  - These variables are stored in R with numbers 1,2,3... And are then given level-names. R calls these variables factors.

– Character variables

- These variables are stored in as text. No calculations can be done with character variables.