ITA-2017 R-Tutorial

Hands-On Exercise 1 - Vectors

Task: In this first excercise, we well work with vectors.

Execute the following three statememnts, which define three vectors of different types. Each of the three vectors has the same number of elements and alltogether represent a bunch of people, with their age and sex.

```
names<-c("Steffen", "Sabine", "Andreas", "Thomas", "Rick",
         "Inge", "Marie")
ages<-c(45, 49, 51, 33, 42, 39, 28)
sex<-c('m', 'f', 'm', 'm', 'm', 'f', 'f')
```

First, we look how many elements in the vector `names` reside:

```
length(names)
```

Next, we return the names in alphabetical order

```
sort(names)
```

Then, we want to know, how many persons are male and how many are female

```
table(sex)
```

If we want it graphically, we can type:

```
barplot(table(sex))
```

To print the informations about Sabine, we execute the following code (Sabine is the 2. dataset):

```
i=2
print(paste(names[i], ages[i], sex[i], sep=','))
```

Next, we want to know the names of all female persons. To do so, we first look for the value `'f'` in the vector `sex`:

```
 sex =='f'
```

The result is a vector of boolean values, which can then be used as a filter for the `names` vector:

```
names[sex=='f']
```

Nice, isn't it ?

Next, we want to print all persons with an age, which is below the average age of all persons.

To get the average age, we have to type:

```
mean(ages)
```

Which datasets are affected?

```
ages < mean(ages)
```

So, when we put all together, we have:

```
names[ages < mean(ages)]
```

Next, we want to print the names of all persons, ordered by their age (descending). What we need is the function `order`. Type `help(order)` to get more information. Then type

```
order(ages)
```

What does this output mean?

To reorder the elements in a vector, whe can do the following:

```
names[c(7,6,5,4,3,2,1)]
```

This command outputs the names in reverse order (starting with index position 7, than 6, ...)

We can combine these two techniques and get the following command:

```
names[order(ages)]
```

That's it - Marie is the youngest, followed by Thomas ...

An alternative way to get this information can be realized with named vectors.

So, first of all, we attach the (unique) names with the ages vector:

```
names(ages)<-names
```

`names(...)` is a function to get or set names for an object (a vector in this case). Do not get confused by the vector 'names' in our example, which has coincidentally the same name. Type

```
ages
sort(ages)
```

After we have accessed names to our vector `ages`, we not only can access the ages by index, but also by name. Try the following two statements, which should return the same value:

```
ages[4]
ages["Thomas"]
```

To get the names of a named object, type

```
names(ages)
```

The names of the persons, ordered by their age:

```
names(sort(ages))
```