Session 1: Basics, writing scripts, read/write data

Data types

Assigning values with destinctive data types to variables

In order to handle your data right it is very essential to know which type your data is, as different data types have different properties and allow different operations. There are various different data types available in R (e.g. there are several different date formats in R that behave slightly different and are explicitly required for some functions). As a start in R the most essential types to know are:

- **Logical** These only have two states either TRUE or FALSE:

```
# Logical variables
x_true <- TRUE
x_false <- FALSE
```

- **Numeric** These are integer or floating point (double) numbers:

```
# Numeric variables (either integer or double (floating point number)):
x_int <- 1
# if you want to set a variabe explicitly to integer:
x_int <- 1L

x_double <- 1.54
```

- **Character** Text strings of any length. Indicated by apostrophs:

```
# Character variables
x_chr <- "Text"
y_chr <- "Text 1"
z_chr <- "12.426"
```

- **Factor** Very critical data type. It is often mistaken with character or numeric. When it is not explicitly needed in any function (e.g. ANOVA requires factors for the model) factors should be avoided (see data.frames). These are used for categorical data and can only take defined values ("labels") predefined in e.g. a vector. The example below should illustrate the difference between a character vector and the case when the character vector is converted to a factor vector:

```
x_chr <- c("R1", "R2", "R4", "R1", "R1", "R2")
x_fct <- as.factor(x_chr)
x_chr
```

```
## [1] "R1" "R2" "R4" "R1" "R1" "R2"
```

```
x_fct
```

```
## [1] R1 R2 R4 R1 R1 R2
## Levels: R1 R2 R4
```

- **Date formats** As mentioned above there are several date formats available. Here the standard date format is mentioned, as it is the easiest to handle in the beginning. The standard date structure that is automatically recognized by the *as.Date()* function is YYYY-MM-DD. If the date structure in your data differs, you as a user have to give the function the date format as well. This is illustrated in the examples below. How to give the date format to the function can be found in the help of *strptime*:

```
date_1 <- as.Date("2010-12-25")
date_2 <- as.Date("2010-08-05")

# Examples for dates if the date format is not the standard format If the date
# (e.g. in your data) is not given in the standard format as in the two examples
# above, the user has to tell the function as.Date in which format the date is
# given. See the two examples below:
date_3 <- as.Date("25.12.2010", "%d.%m.%Y")

date_chr <- "12:25 05.08.10"
date_4 <- as.Date(date_chr, "%H:%M %d.%m.%y")
```

Querying the data type

To ask which type your data is simply use the function *typeof()*. The outcome for different data types is illustrated below:

```
typeof(x_true)
```

```
## [1] "logical"
```

```
typeof(x_int)
```

```
## [1] "integer"
```

```
typeof(x_double)
```

```
## [1] "double"
```

```
typeof(x_chr)
```

```
## [1] "character"
```

```
typeof(x_fct)
```

```
## [1] "integer"
```

```
typeof(date_1)
```

```
## [1] "double"
```

Conversion between different data types

Sometimes it is required to convert your data from one data type to another. Examples are when loading data; R interprets dates in a csv file as characters but the user wants them explicitly as dates, or if for any analysis categorical data is given as numeric or character, but a function requires them explicitly as factors. To convert the data type see following examples:

```
x <- 1.23456
x_chr <- as.character(x)
x_factor <- as.factor(x_chr)
x_logic  <- as.logical(x)
```

Data structures