E12-1 Subsetting data frames

Subsetting e.g. a data frame can be realized using the indexing method which is also used to access individual cells or groups of cells like entire rows or columns. We will use the following data frame for the examples below.

```
x <- c(1, 2, 3, 4, 5)
y <- c(1.4, 2.5, 3.6, 4.0, 5.5)
z <- c("A", "B", "A", "A", "B")
df <- data.frame(X = x, Y = y, Z = z)
df
```

```
##   X   Y Z
## 1 1 1.4 A
## 2 2 2.5 B
## 3 3 3.6 A
## 4 4 4.0 A
## 5 5 5.5 B
```

In order to get a data frame which has only the first and third column of the original one, we can subset it by selecting columns 1 and 3 and store the result in a new variable (of course you can also overwrite the original one if you don’t need it any longer).

```
df_a1 <- df[,c(1,3)]
df_a1
```

```
##   X Z
## 1 1 A
## 2 2 B
## 3 3 A
## 4 4 A
## 5 5 B
```

Alternatively, we can subset the data frame by removing the second column.

```
df_a1 <- df[,-2]
df_a1
```

```
##   X Z
## 1 1 A
## 2 2 B
## 3 3 A
## 4 4 A
## 5 5 B
```

A third alternative (genreally preferable if one wants to overwrite the original variable anyway) would

set the respective columns to NULL.

```
df_a2 <- df
df_a2$Y <- NULL
df_a2
```

```
##   X Z
## 1 1 A
## 2 2 B
## 3 3 A
## 4 4 A
## 5 5 B
```

To subset a data frame by rows, the same logic as for the columns can be applied. Let's just have a look on one alternative to select rows one and five.

```
df_a1 <- df[c(1,5),]
df_a1
```

```
##   X   Y Z
## 1 1 1.4 A
## 5 5 5.5 B
```

While the above examples select columns and rows by their index range, a completely different approach can be used for rows, too.

Imagin that you want to select your rows not based on a index number but based on the content of one or more columns within that row. In this case, you can use logical operators to get the job done. For example, if you want to select all rows which have an A in column Z, just do the following:

```
df_a1 <- df[df$Z == "A",]
df_a1
```

```
##   X   Y Z
## 1 1 1.4 A
## 3 3 3.6 A
## 4 4 4.0 A
```

Of course this is equivalent to using the column index and not the column name.

```
df_a1 <- df[df[,3] == "A",]
df_a1
```

```
##   X   Y Z
## 1 1 1.4 A
## 3 3 3.6 A
## 4 4 4.0 A
```

As you already know, logical values can be combined using boolean operators. This makes it easy to

select all rows with value A which - at the same time - have a value of less or equal to three in column X.

```
df_a1 <- df[df$Z == "A" & df$X <= 3,]
df_a1
```

```
##   X   Y Z
## 1 1 1.4 A
## 3 3 3.6 A
```

Finally, you can of course combine your selections for rows and columns.

```
df_a1 <- df[df$Z == "A" & df$X <= 3, c(1,3)]
df_a1
```

```
##   X Z
## 1 1 A
## 3 3 A
```