E12-1: Cleaning data frames

NAs

NA values generally stand for “not available” values i.e. missing values. In R, NA can also mean that the value is “not a number” which would be the case for e.g. a division by zero. While the actual meaning of NA might be of interest since a missing value might have already been “existed” in the source data set while “not a number” might be the result of a wrong piece of code, the result is always the same: one can not do anything with NAs except finding, replacing or to some extend ignoring them.

The main function for handling NAs is `is.na`. It returns a boolean vector or matrix with the value TRUE at each position which is NA in the data set under question.

Once identified, one can replace the NA values by any other value. However, the most frequent workflow will be the other way round: assign NA to a specific value or character sequence.

Have a look at C03-1 Missing values now for a hands-on introduction.

Cleaning data frames

In general, the goal of cleaning a data frames is to derive a technically correct data frame. This implies that there is only a single information within each cell and that cells with no values are identified by a specific constant (e.g. NA, -999).

Cleaning data frames can be quite complex and time consuming. In addition, each data set might be different so there is not much room for operational scripting (if one is restricted to simple scripts). Let's have a look at an example given by the following data set:

```
Region        Number
   R11      50 (10%)
   R12      200 (7%)
```

The column Region will become a character/factor data type in R's data frame. The column Number will also become a character because of the character symbols and the doubling of the numbers. In this example, one aim could be the splitting of the column Number in a column for the absolute values and a column for the relative ones. The latter could include the “%” or not. In the former case it would be a character, in the later a numeric data type.

While the actual workflow is tightly related to the individual data set, four tasks have to be used quite frequently:

Method	Situation	R function
Removing parts of a cell value	A typically numeric value is stored as character/factor because of some kind of additional string sequence in the cell. Example: “(205)”, “205%”, “205*”	substring

Method	Situation	R function
Splitting the content of a cell	A column contains two types of information Example: “1000 (5%)”, “1000 chairs”	substring
Converting data types (coercing)	A column is defined as data type x but you need y (and the values are appropriate for that	as.<data type>
Setting/changing NA values	Cleaning might result in NAs which should be interpreted as 0.0 in the following or simliar	is.na

As you notice, the four task need only three different functions. Hence, the main problem won't be related to the selection of functions but to understand what you actually have to do to clean your individual data set.

Have a look at E03-2 Extracting substrings and E03-3 Coercing data types now for more information on splitting strings and data type conversion.

Subsetting or combining data frames

So far we have used functions to add columns to an existing data frame or to change the values of individual cells. But this is often only the first step in cleaning data sets. In the end, you might only be interested in a subset of the information or you want to combine one data frame with another.

Subsetting

Subsetting implies that you remove certain rows and/or columns from a data frame to reduce the actual data set to what is needed for your analysis. The two types are realized with partially different manners:

- subsetting by selecting the rows and columns you want in your final data frame
- subsetting by removing the rows and columns you want in your final data frame

Both can easily be done using the indexing methods of the data types already introduced in C00-1 and C00-2. Alternatively, one could use the subset function but in some cases, it may fail (have a look at this discussion on stackoverflow if you feel up to it).

Combining

Combining implies that you add columns or rows from a second data frame to an existing one. The task is realized by two different work flows:

- for consecutively combining data frames by rows, use the rbind function and keep in mind that both data frames must have the same amount of columns and the same column names (but not the same order of columns)
- for merging data frames by values in a specific column, use the merge function.

Have a look at C12-1 Subsetting data frames now for more information on this subject.