Session 1 - Part 2: Read/write/analyze data

Reading your data

Before starting with your analysis you have to get your data into R. R allows to work with many different data types, such as .csv, .xlsx, ASCII, .m, but also GIS raster or shape objects. Most of the time your data might be avilable in Excel spreadsheets. Here I recommend to save the data as .csv for the use in R. In the following two examples I will show how to load data in R. The first example demonstrates how easy reading data is, for the case of a already very clean and well prepared data set is. The second data set holds a very messy (again no offense to your data) data structure. To arrange the data for further analyses then requires some skills and some "creative" playing around with your data. Here one can consider to first prepare the data in Excel before loading it into R. If you gained already some experience in R it is much faster to rearrange your data in R.

The working directory

For many analyses it is advisable to share all your input data, the R script and the outputs you might produce in one working directory. Especially when starting with R this is the easiest way to work with R. To set a working directory and to maybe query what the current working directory is following commands are helpful:

```r
setwd("C:/Users/Christoph/Documents/Projects_R/RCourse")
getwd()
```

```
## [1] "C:/Users/Christoph/Documents/Projects_R/RCourse"
```

The path to your working directory you simply can copy paste from your explorer. As it is a text string it requires quoation marks. Caution here!: When working in Windows you have to replace all back slashes ("\") in the path with front slashes ("/"). Back slashes have a different purpose in R. With the command *setwd()* you set your path to the directory you are working in. Now when loading or writing a file you do not have to take care about the paths anymore. Simply give the file name and R will look directly under this path for it.

Reading/preparing a .csv file

To read and prepare a .csv file I will demonstrate on the data you provided me with. The first example is very clean data. In such a case there is nothing much to do but loading the data. The second example requires some tricks to bring it in the desired form for further analyses. If you want to follow the steps below, please save the .csv I sent you to a folder on your computer and set this as your working directory.

R offers many ways to load data. For starters the basic *read* family is the recommended way. When typing *read* into the help you will see that there are many different read commands. The most useful for you are the *read.table()* which is a more general read command and the *read.csv()* that is basically a *read.table()* but with default settings optimized for reading spreadsheets (e.g. seperator is ",", the first line is handled as a header: header = TRUE). A second command for .csv file might be important for you when you use european pc settings (e.g. seperator in csv files is ";" instead of ",", and "," is the decimal symbol instead of the decimal point ".") then the *read.csv2()* is the right choice as it uses european pc settings as default.

Example 1:

As the data is already well prepared we will use the default *read.csv()* command for loading the data.

```
# Setting the working directory to the folder where the data is located
setwd("C:/Users/Christoph/Desktop/R Course")

# Read in the csv file with the standard read.csv command
bio_data <- read.csv("Biomass_compiled.csv")
```

As you can see here I ommitted the *stringsAsfactors =* FALSE, allthough I mentioned one should be really careful with factors when for example loading the data. In the further analysis (ANOVA) however, categorical data (e.g.) plant variety or any treatment is explicitly required as factors.

Example 2

This example is just for demonstration. It is ok if you do not understand every step I did here. It should basically demonstrate that it is possible (and if you have some experience it is easy and quick as well) to prepare your data in R, but can be laborious when the data is "messy" (this does not mean that it is then neccessarily easier in other software such as Excel :-)).

The data set you provided me is a time series of soil moisture data in different depth for several plots and repetitions. In my example I only want to extract the time series data for the different plots. Following I explain the steps I had to take:

As the data is not simply columns of the same length with a header and values of the same data type I used the more flexible read function *read.table()*.

```
# Setting the working directory (it is already set, but just examplatory set again)
setwd("C:/Users/Christoph/Desktop/R Course")

# Read in the csv file now with the read.table command
soilwater_data <- read.table("wheat-soil_water.csv", skip = 8, nrows = 40,
                             sep = ",", fill = TRUE, stringsAsFactors = FALSE)
```

Here I skipped the first 8 rows using the option *skip = 8* and as I know from my data that I have 40 plots I only read 40 rows from then using the option *nrows = 40*. The option *fill* fills up blank fields in columns. This is neccessary here, as not for every date we have measurements on every plot.

Next I will extract the plot numbers as I use them later on for the headers of the columns and then I remove the first five columns as they do not hold any water content data. The read function also read many columns holding no data at all. These I will remove as well.

```
plot_nr <- soilwater_data[,2]
soilwater_data <- soilwater_data[,-(1:5)]
soilwater_data <- soilwater_data[,colSums(soilwater_data, na.rm = TRUE) > 0]
```

As you migth remember from Session 1 Part 1 a data frame is organized in columns. Therefore I have to "flip" the data set. This is done with the matrix operation *t()* that stands for transpose. Additionally I convert the matrix to a data.frame and add the plot numbers as headers using the function *colnames*. Here I add the word "plot" to each plot number using the function *paste*.

```
soilwater_data <- as.data.frame(t(soilwater_data), row.names = FALSE)
colnames(soilwater_data) <- paste("plot_", plot_nr, sep = "")
```

The same steps as above I will repeat with the date and depth data in the .csv file. In a further step I will merge the water content data with dates and depths using the column bind function *cbind()* in R.

```
# Setting the working directory (it is already set, but just examplatory set again)
setwd("C:/Users/Christoph/Desktop/R Course")

# Read in the csv file now with the read.table command
soilwater_datedepth <- read.table("wheat-soil_water.csv", skip = 4, nrows = 2,
                                  sep = ",", fill = TRUE, stringsAsFactors = FALSE)

soilwater_datedepth <- soilwater_datedepth[,-(1:5)]
soilwater_datedepth <- soilwater_datedepth[,!is.na(soilwater_datedepth[1,])]

soilwater_datedepth <- t(soilwater_datedepth)
soilwater_datedepth <- data.frame(date  = as.Date(soilwater_datedepth[,2], "%d.%m.%y"),
                                  depth = as.numeric(soilwater_datedepth[,1]))

soilwater_data <- cbind(soilwater_datedepth, soilwater_data)
```

As you can see when creating the date frame with dates and depths I used the data type conversion to dates and numerics as I showed in Part 1 of this session.

After all these transformations the data frame looks as follows in R (only a part of the data frame shown in the document):

```
head(soilwater_data[,1:10])
```

```
##         date depth plot_1 plot_3 plot_6 plot_8 plot_2 plot_4 plot_5 plot_7
## 1 2013-10-18     5   8.33  17.80  18.81  19.62  24.63  16.55   7.39  22.67
## 2 2013-10-18    15  19.44  30.15  28.38  28.82  25.10  26.95  19.73  38.50
## 3 2013-10-18    25  26.40  29.92  31.28  31.98  34.67  34.75  33.15  34.99
## 4 2013-10-18    35  28.02  25.30  27.52  29.18  25.57  35.07  28.16  30.60
## 5 2013-10-18    45  25.03  20.63  19.33  23.57  17.52  30.60  25.03  22.35
## 6 2013-10-18    55  21.42  18.92  19.97  15.87  14.86  23.64  27.17  23.96
```

Getting a first overview of the data

Here I will continue with the data set from example 1. To get a first overview of your data the commands *str()* and *summary()* are very useful. *str()* gives you an overview of the structure of your data (e.g. data types of the variables and the first 10 values or the possible levels when the variable is a factor). *summary()* gives you basic statistical measures of the variables in your data (e.g. min, max, quantiles, mean). For bio data the results look as follows:

```
str(bio_data)
```

```
## 'data.frame':    70 obs. of  17 variables:
##  $ Water        : Factor w/ 2 levels "WS","WW": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Variety      : Factor w/ 7 levels "Caesar","Cardinal",..: 1 1 1 1 1 2 2 2 2 2 ...
##  $ Replication  : int  1 2 3 4 5 1 2 3 4 5 ...
##  $ Total_T      : num  1527 1698 1271 1370 1279 ...
##  $ Initial_DM   : num  1.29 1.08 1.33 2.4 1.79 ...
##  $ Shoot_DM     : num  3.71 4.9 3.37 2.73 1.93 2.56 1.48 2.43 4.06 2.59 ...
##  $ Leaf_DM      : num  4.91 6.18 4.02 3.38 2.51 ...
##  $ Tuber_FM     : num  36 43.7 26.7 36.4 32.8 ...
##  $ Tuber_DM     : num  9.5 11.93 7.27 8.73 7.4 ...
##  $ AG_TDM       : num  8.62 11.08 7.39 6.11 4.44 ...
##  $ Gain_AG_DM   : num  7.32 9.99 6.07 3.71 2.65 4.97 2.9 5.59 7.59 5.82 ...
##  $ UG_DM        : num  0.84 1.48 5.86 2.66 6.88 6.37 1.99 2.34 3.76 2.98 ...
##  $ Gain_TDM     : num  17.7 23.4 19.2 15.1 16.9 ...
##  $ TE_Gain_AG_DM: num  4.79 5.89 4.77 2.7 2.07 3.61 2.33 3.78 5.5 3.42 ...
##  $ TE_Gain_TDM  : num  11.6 13.8 15.1 11 13.2 ...
##  $ TE_Tuber_DM  : num  6.22 7.03 5.72 6.37 5.79 0.82 6 4.47 2.68 3.54 ...
##  $ X            : logi  NA NA NA NA NA NA ...
```

```
summary(bio_data[,1:6]) #To save space in the document only the first 6 columns used
```

```
##  Water       Variety     Replication    Total_T       Initial_DM   
##  WS:35   Caesar  :10   Min.   :1    Min.   :1228   Min.   :1.022  
##  WW:35   Cardinal:10   1st Qu.:2    1st Qu.:1577   1st Qu.:1.274  
##          Desiree :10   Median :3    Median :1921   Median :1.509  
##          Diamant :10   Mean   :3    Mean   :2416   Mean   :1.644  
##          Farida  :10   3rd Qu.:4    3rd Qu.:3429   3rd Qu.:1.953  
##          Mondial :10   Max.   :5    Max.   :4330   Max.   :2.752  
##          Spunta  :10                                              
##     Shoot_DM    
##  Min.   :1.480  
##  1st Qu.:3.065  
##  Median :3.800  
##  Mean   :4.004  
##  3rd Qu.:4.800  
##  Max.   :7.550  
## 
```

The first analysis reveals that the variable *Replication* is treated as a numeric variable. This is because of the read in. As the possible levels of replications are given as numbers R thinks these are actually numbers. Therefore we have to convert the variable *Replication* to factor as we have learned it in Part 1 of this session.

```
bio_data$Replication <- as.factor(bio_data$Replication)
```

To check whether the data is right now we again have a look on the structure of the data:

```
str(bio_data)
```

```
## 'data.frame':    70 obs. of  17 variables:
##  $ Water        : Factor w/ 2 levels "WS","WW": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Variety      : Factor w/ 7 levels "Caesar","Cardinal",..: 1 1 1 1 1 2 2 2 2 2 ...
##  $ Replication  : Factor w/ 5 levels "1","2","3","4",..: 1 2 3 4 5 1 2 3 4 5 ...
##  $ Total_T      : num  1527 1698 1271 1370 1279 ...
##  $ Initial_DM   : num  1.29 1.08 1.33 2.4 1.79 ...
##  $ Shoot_DM     : num  3.71 4.9 3.37 2.73 1.93 2.56 1.48 2.43 4.06 2.59 ...
##  $ Leaf_DM      : num  4.91 6.18 4.02 3.38 2.51 ...
##  $ Tuber_FM     : num  36 43.7 26.7 36.4 32.8 ...
##  $ Tuber_DM     : num  9.5 11.93 7.27 8.73 7.4 ...
##  $ AG_TDM       : num  8.62 11.08 7.39 6.11 4.44 ...
##  $ Gain_AG_DM   : num  7.32 9.99 6.07 3.71 2.65 4.97 2.9 5.59 7.59 5.82 ...
##  $ UG_DM        : num  0.84 1.48 5.86 2.66 6.88 6.37 1.99 2.34 3.76 2.98 ...
##  $ Gain_TDM     : num  17.7 23.4 19.2 15.1 16.9 ...
##  $ TE_Gain_AG_DM: num  4.79 5.89 4.77 2.7 2.07 3.61 2.33 3.78 5.5 3.42 ...
##  $ TE_Gain_TDM  : num  11.6 13.8 15.1 11 13.2 ...
##  $ TE_Tuber_DM  : num  6.22 7.03 5.72 6.37 5.79 0.82 6 4.47 2.68 3.54 ...
##  $ X            : logi  NA NA NA NA NA NA ...
```

Now you can see that *Replication* is a categorical variable as the numbers 1 to 5 are now levels of a factor variable.

Visualization of the data

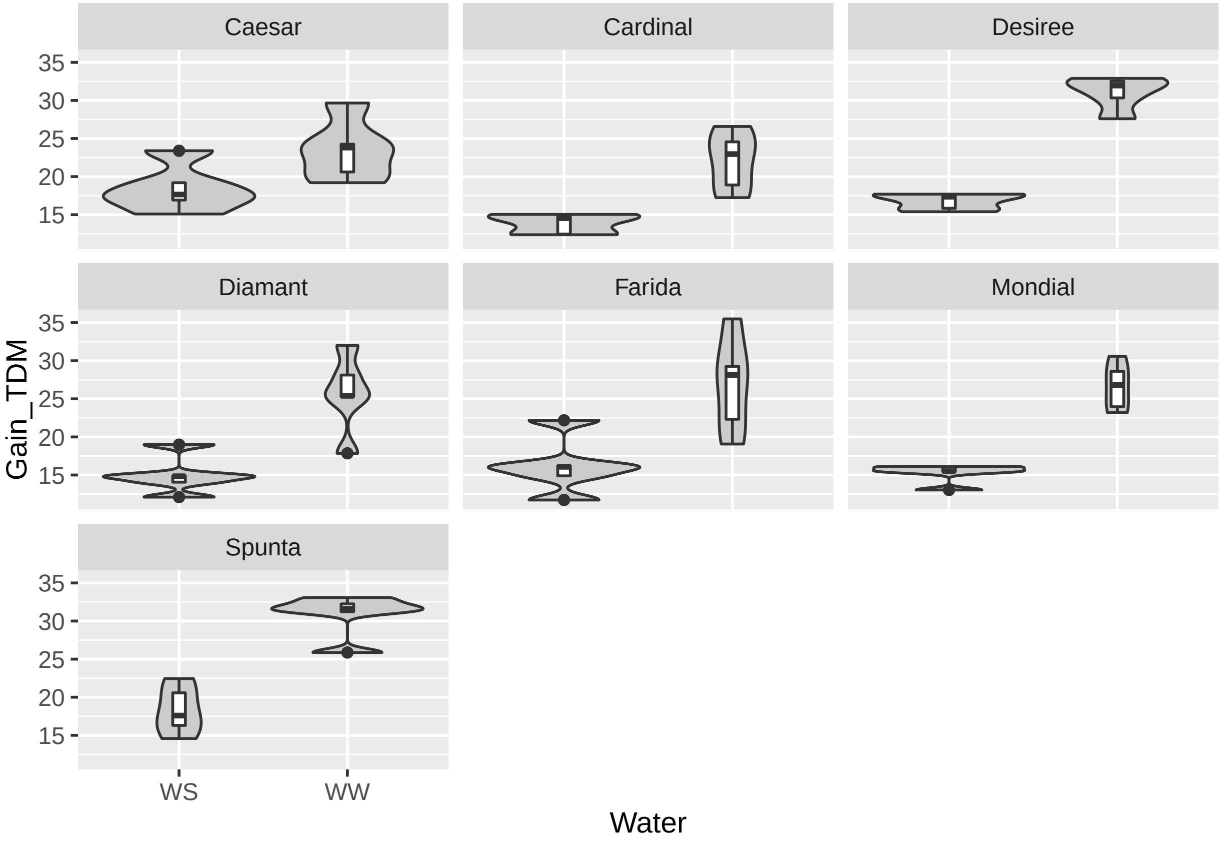
Visualization is done in Session 3 of this short course. But it is always essential to have a first look on the plotted data before starting an analysis. Here I just want to plot the data without explaining in detail the steps. This will be done in Session 3.

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.2.4
```

```
# Visualization of the data. Just to show you. More about that in Session 3
ggplot(bio_data) +
  geom_violin(aes(x = Water, y = Gain_TDM), fill = "grey80") +
  geom_boxplot(aes(x = Water, y = Gain_TDM), width = 0.1) +
  facet_wrap(~Variety)
```



###Analysis of the data The two analyses shown here are just examplatory, as the possibilities to apply to the data are manifold. As examples a two way ANOVA and TukeyHSD test are shown.

When doing an analysis the first time in R you as a user face two problems. First you do not know what is the right command for the desired analysis and second sometimes it is not intuitive in the beginning how to apply the function (if you found the right one) to your data.

For both problems the simple answer is **post the right question in google**. Fortunately, the R community is really strong and most certainly someone faced the same problem already before you did. Here I want to highlight the community on **stackoverflow**, as here many good answeres are posted to many very practical problems. When you know already which function to apply the R help and in particular the examples at the bottom of every help file are helpful in many cases.

ANOVA

In our example we developed a two way ANOVA model using the function *aov()* that is implemented in the base package of R (so no installation of packages required). To get an idea of how to set up the model for the ANOVA the examples in the help file are useful. One of the examples is given as:

```
head(npk)
```

```
##   block N P K yield
## 1     1 0 1 1  49.5
## 2     1 1 1 0  62.8
## 3     1 0 0 0  46.8
## 4     1 1 0 1  57.0
## 5     2 1 0 0  59.8
## 6     2 1 1 1  58.5
```

```
aov(yield ~ block + N * P + K, npk)
```

```
## Call:
##    aov(formula = yield ~ block + N * P + K, data = npk)
##
## Terms:
##                    block        N        P        K      N:P Residuals
## Sum of Squares  343.2950 189.2817   8.4017  95.2017  21.2817  218.9033
## Deg. of Freedom        5        1        1        1        1        14
##
## Residual standard error: 3.954232
## Estimated effects may be unbalanced
```

Here *npk* is an example data set implemented in R (Just for your information, there are many example datasets implemented for testing your analysis). Transfering the concept of this example to our data gives following analysis:

```
# Applying two way ANNOVA to the data
bio_aov <- aov(Gain_TDM ~ Water + Variety + Water*Variety, data = bio_data)

# Getting the results of the ANNOVA with summary
summary(bio_aov)
```

```
##                Df Sum Sq Mean Sq F value  Pr(>F)    
## Water           1 1894.0  1894.0 158.805 < 2e-16 ***
## Variety         6  297.3    49.5   4.154 0.00161 ** 
## Water:Variety   6  134.9    22.5   1.885 0.09947 .  
## Residuals      56  667.9    11.9                    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The individual components of the model used in *aov()* are combined with *+* and ***, where a combinantion with *+* only gives the main effect of each variable and *** also gives the combined effect of the variables.

TukeyHSD

A similar way was chosen for applying the TukeyHSD test. Communities and help gave a good idea of how to implement the analysis for our data:

```r
# Applying the Tukey honest significant differences test to our ANNOVA model
bio_THSD <- TukeyHSD(bio_aov, ordered = TRUE)
```

For both analyses, I want to point out that the result were stored as lists. For accessing particular parts of the results, please remember how to access lists, as it was explained in Part 1 of this Session.

Writing data

Often you as a user want to export particular results of your analysis. The straight forward way to do this is to export data (either data.frames or matrices) as .csv files. To write .csv files use the function *write.csv()*. As example I write a subset of our TukeyHSD test to the .csv file "tukey_HSD.csv". I extract the subset of the combined effect of water and variety of this analysis (which is a table) and write it to the file:

```r
# Writing the results of our TukeyHSD test to a csv file in our working
# directory
write.csv(x = bio_THSD$`Water:Variety`, file = "tukey_HSD.csv")
```

If you want to customize the data export (e.g. ommiting the headers, disable the quatation of rownames etc.) look into the help for the function.