Exercise Sheet 1: Basic Data Management and Plotting in **R**

Exercise 1 (Import from Excel)**.** Download the following molecular weight and subcelluar targeting tables from the TAIR site, import the files into Excel and save them as tab delimited text files or comma separated files.
MolecularWeight_tair7.xls
TargetP_analysis_tair7.xls

(i) Import the tables into R-from your local directory and name the data frame `my_mw` and `my_target`.

(ii) Check tables with `head, tail`, calculate the tables' dimension and check the column and row names.

(iii) Rename *GeneName* and *Sequence.id* column to *ID*.

(iv) Merge the two tables based on common ID field using the function `merge`. Make sure that all elements from both tables are included in the merged data frame. Name the merged table `my_mw_target`.

(v) Now, extract from one table the first 100 rows before the merge and then merge the two files such that the merged data frame includes only the values from the shortened table (e.g. no non-matching rows (NAs) in the merged file).

(vi) Replace all NAs in the data frame `my_mw_target` with zeros.

(vii) Apply several combinatorial queries (filters) on the merged data. For example select those proteins with molecular weight larger than 100 000 and which are located in chloroplasts (C).

(viii) How many protein entries in the `my_mw_target` data frame have a MW of greater then 4000 and less then 5000. Subset the data frame accordingly and sort it by MW to check that your result is correct.

(ix) Use a regular expression in a substitute function to generate a new ID column that lacks the gene model extensions (the dots and numbers behind the dots).

(x) Retrieve those rows in where the Gene ID contains the following identifiers: `c('AT5G52930.1', 'AT4G18950.1', 'AT1G15385.1', 'AT4G36500.1', 'AT1G67530.1')`. For example, you can use the `in()` function for this query. As an alternative approach, you can assign the Gene ID column to the row index of the data frame and then select the rows using the row index.

(xi) Calculate the number of gene models per gene with the `table()` function. Use the new ID column generated in (ix). Sort the data frame by the number of gene models such that those genes with the most gene models are on the top of the table.

(xii) Perform a variety calculations across columns: calculate the means, standard deviations and medians for all numeric columns. Calculate the two-sample t-test for the molecular weight between proteins with only one gene models and proteins with more than gene models.

(xiii) Export the data frame into a CSV file.

Exercise 2 (Plots)**.** Create some basic plots.

(i) Generate a plot with 2 boxplots (dark red and dark blue) for molecular weight and amino acid counts. Change the labels to "MW" and "AA" and create an appropriate title of the plot.

(ii) Plot the histogram and density of the molecular weight in two different colors (easily distinguishable), the density line should be a dashed line.

(iii) Create a barplot of molecular weights for the first five genes-set an appropriate limit of the vertical axis and select a nice color scheme for the barplots.

Exercise 3 (Venn diagram)**.** Use a venn diagram function to create fancy venn diagrams.

Download or import the venn diagram function and generate 5 sets of random IDs. Plot a 5-way venn diagram for the random ID sets and save it on your computer

```
source("http://faculty.ucr.edu/~tgirke/Documents/R_BioCond/My_R_Scripts/overLapper.R") # Imports the venn diagram function.
setlist <- list(A=sample(my_mw_target4[,1], 1000), B=sample(my_mw_target4[,1], 1100), C=sample(my_mw_target4[,1], 1200), D=sample(my_mw_target4[,1], 1300), E=sample(my_mw_target4[,1], 1400))
# Generates 5 sets of random IDs.
OLlist <- overLapper(setlist=setlist, sep="_", type="vennsets")
counts <- sapply(OLlist$Venn_List, length);
vennPlot(counts=counts, ccol=c(rep(1,30),2), lcex=1.5, ccex=c(rep(1.5,5), rep(0.6,25),1.5)) # Plots a 5-way venn diagram for the random ID sets.
OLexport <- as.matrix(unlist(sapply(OLlist[[4]], paste, collapse=" ")))
write.table(OLexport, file="OLexport.txt", col.names=F, quote=F, sep="\t") # Exports intersect data in tabular format to a file.
pdf("venn_plot.pdf");
vennPlot(counts=counts);
dev.off() # Saves the plot image to a PDF file.
```

(i) Generate two key lists each with 4 random sample sets. Compute their overlap counts and plot the results for both lists in one venn diagram.

(ii) Write all commands from the previous exercises into an R script (exerciseRbasics.R) and execute it with the `source()` function like this: `source('exerciseRbasics.R')`. This will execute all of the above commands and generates the corresponding output files in the current working directory.