Class 2 - Handling Tabular Data

1. Data Frames

- A **data frame** in R is the format used to store tables, particularly if it has mixed data types (e.g. character (words) and numeric data).
- Another common format to store data is a **matrix**, which should be only used to store data in a single form (character or numeric).
- If this conditions are guarenteed one can change the format of a given table from dataframe to matrix (or vice-versa) using these functions:
  - as.data.frame()
  - as.matrix()

1.1. Importing data to the R environment

R has multiple functions to import tabular data from a file (?read.table), that differ on the deafult values of the arguments:

```
- read.table() : more generic, some default arguments need to be changed
- read.csv() : for .csv files with values separated by ',' and '.' as decimal sign and header (default)
- read.csv2() : for .csv files with values separated by ';' and ',' as decimal sign and header (default
- read.delim() : for text files with 'tab' as separator and '.' as decimal sign and header (default)
- read.delim2() : for text files with 'tab' as separator and ',' as decimal sign and header (default)
```

```r
# open root_length.csv using the wordpad and choose the best function to import the file
root_l <- read.csv2("data/root_length.csv")

head(root_l)
```

```
##    line length.cm..1 length.cm..2 length.cm..3 Lat_roots
## 1 Col-0        5.968        5.941        5.912        10
## 2 Col-0        5.334        5.266        5.271         5
## 3 Col-0        4.553        4.566        4.638         5
## 4 Col-0        5.144        5.196        5.153         6
## 5 Col-0        5.439        5.482        5.346         7
## 6 Col-0        5.843        5.918        5.864        11
```

Root_l contains main root lengths from 40 plants from two different genotypes, identified in column 1. Each root was measured three times, with lengths displayed in columns 2 to 4. Column 5 has the number of lateral roots observed for each root.

Examples of functions that may be used to explore a dataframe by…

…Size:

```
dim() - returns a vector with the number of rows in the first element, and the
  number of columns as the second element
nrow() - returns the number of rows
ncol() - returns the number of columns
```

…Content:

```
head() - shows the first 6 rows
tail() - shows the last 6 rows
```

…Names:

```
names() - returns the column names (synonym of colnames())
rownames() - returns the row names
```

…Summary:

```
str() - structure of the object and information about the class, length and content
  of each column
summary() - summary statistics for each column
```

1.2. Manipulating/Mining data frames

We need to calculate the mean of the three measurements and the lateral root density (# lateral roots / mean length)

```
# Calculate mean of values in 3 columns

rowMeans(root_l[,2:4])
```

```
##  [1] 5.940333 5.290333 4.585667 5.164333 5.422333 5.875000 6.238333
##  [8] 3.790333 4.378333 6.239000 5.620333 5.134667 5.547000 5.960000
## [15] 5.631000 5.435333 5.953667 4.887000 5.391333 6.414667 5.628000
## [22] 5.835667 5.598667 6.494333 4.647333 6.462667 5.563333 6.311667
## [29] 5.763000 5.879667 5.693667 5.824333 5.625667 6.500000 4.588667
## [36] 6.422000 5.572667 6.331667 5.717667 5.816667
```

```
# To add this value to the data frame there are two options
# Using cbind() function

root_l1 <- cbind(root_l, mean_length = rowMeans(root_l[,2:4]))

head(root_l1)
```

```
##    line length.cm..1 length.cm..2 length.cm..3 Lat_roots mean_length
## 1 Col-0        5.968        5.941        5.912        10    5.940333
## 2 Col-0        5.334        5.266        5.271         5    5.290333
## 3 Col-0        4.553        4.566        4.638         5    4.585667
## 4 Col-0        5.144        5.196        5.153         6    5.164333
## 5 Col-0        5.439        5.482        5.346         7    5.422333
## 6 Col-0        5.843        5.918        5.864        11    5.875000
```

```
# Using the $ operator Add a new column containing the mean of the three length measurements

root_l$mean_length <- rowMeans(root_l[,2:4])
head(root_l1)
```

```
##    line length.cm..1 length.cm..2 length.cm..3 Lat_roots mean_length
## 1 Col-0        5.968        5.941        5.912        10    5.940333
## 2 Col-0        5.334        5.266        5.271         5    5.290333
## 3 Col-0        4.553        4.566        4.638         5    4.585667
## 4 Col-0        5.144        5.196        5.153         6    5.164333
## 5 Col-0        5.439        5.482        5.346         7    5.422333
## 6 Col-0        5.843        5.918        5.864        11    5.875000
```

```
# to add a new line below the last line of the table, you can use rbind() function.
# you only need to guarantee that the new line has the same length and structure has the other rows.
```

```
#e.g.
#newline <- c("Col-0", 5.000, 5.000, 5.1000, 10, 5.0500)
#root_l2 <- rbind(root_l1, newline)
```

```
# the elements of each column in root_l are in fact one dimension vectors of size 40
# to calculate lateral root density just divide one column 5 by column 6

root_l$Lat_roots / root_l$mean_length
```

```
##  [1] 1.6834072 0.9451200 1.0903540 1.1618150 1.2909572 1.8723404 2.0838899
##  [8] 1.0553162 0.6851922 1.4425389 1.2454777 0.5842638 1.6224986 1.6778523
## [15] 0.3551767 0.7359254 0.5038912 1.4323716 1.4838630 1.7148202 1.4214641
## [22] 2.3990404 1.0716837 2.7716471 1.0758858 2.0115535 2.1569802 3.1687351
## [29] 2.6028110 2.0409320 1.4050700 2.4037086 1.0665403 2.7692308 1.0896411
## [36] 2.0242915 2.1533676 3.1587260 2.6234478 2.0630372
```

```
# this can be added to another column

root_l$lat_root_density <- root_l$Lat_roots / root_l$mean_length
head(root_l)
```

```
##    line length.cm..1 length.cm..2 length.cm..3 Lat_roots mean_length
## 1 Col-0        5.968        5.941        5.912        10    5.940333
## 2 Col-0        5.334        5.266        5.271         5    5.290333
## 3 Col-0        4.553        4.566        4.638         5    4.585667
## 4 Col-0        5.144        5.196        5.153         6    5.164333
## 5 Col-0        5.439        5.482        5.346         7    5.422333
## 6 Col-0        5.843        5.918        5.864        11    5.875000
##   lat_root_density
## 1         1.683407
## 2         0.945120
## 3         1.090354
## 4         1.161815
## 5         1.290957
## 6         1.872340
```

```
# to save this new data frame in another file
# type ?write.table for more options

write.csv(root_l,file="data/new_root_length.csv")
```

1.3. Working with a real/inconsistent dataset

```
# Files with extension _d1 are the same but saved in different formats:
# Ecoli_metadata_d1.txt : fields separated by tab '\t'
# Ecoli_metadata_d1.txt : fields separated by tab ','

# both these files were purposely damage to simulate a real case experience

# Both these commands will cause an error, why?

# Ecoli_metadata <- read.table("data/Ecoli_metadata_d1.txt",header = TRUE)

# Ecoli_metadata <- read.table("data/Ecoli_metadata_d1.txt",header = TRUE, sep = "\t")
# the space between "Genome" and "size" may be a problem
#   as a rule of thumb never add spaces between column names, use "." or "_"

# Another problem are the empty, invisible spaces: line 31 contains a random comment,
#and since the function tries to force a row with 7 columns, it fails when it reaches this line

# argument fill = TRUE solves this
# If TRUE then in case the rows have unequal length, blank fields are implicitly added.

Ecoli_metadata <- read.table("data/Ecoli_metadata_d1.txt",header = TRUE, sep = "\t", fill = TRUE)
tail(Ecoli_metadata)
```

```
##      sample generation clade strain   cit       run Genome.size  X
## 26    ZDB99      36000    C1 REL606 minus SRR098037        4.61 NA
## 27   ZDB107      38000  Cit+ REL606  plus SRR098038        4.79 NA
## 28   ZDB111      38000    C2 REL606 minus SRR098039        4.62 NA
## 29 REL10979      40000  Cit+ REL606  plus SRR098029        4.78 NA
## 30 REL10988      40000    C2 REL606 minus SRR098030        4.62 NA
## 31                  NA                                       NA NA
```

```
# read.delim already has this arguments as default
Ecoli_metadata <- read.delim("data/Ecoli_metadata_d1.txt")
tail(Ecoli_metadata)
```

```
##      sample                        generation clade strain   cit       run
## 26    ZDB99                             36000    C1 REL606 minus SRR098037
## 27   ZDB107                             38000  Cit+ REL606  plus SRR098038
## 28   ZDB111                             38000    C2 REL606 minus SRR098039
## 29 REL10979                             40000  Cit+ REL606  plus SRR098029
## 30 REL10988                             40000    C2 REL606 minus SRR098030
## 31          #This comment is highly important
##    Genome.size  X
## 26        4.61 NA
## 27        4.79 NA
## 28        4.62 NA
## 29        4.78 NA
## 30        4.62 NA
## 31          NA NA
```

```
# to ignore line 31, because it is only important for you, not for calculations,
#we can use the comment character
Ecoli_metadata <- read.delim("data/Ecoli_metadata_d1.txt", comment.char = '#')
tail(Ecoli_metadata)
```

```
##      sample generation clade strain   cit       run Genome.size  X
## 26    ZDB99      36000    C1 REL606 minus SRR098037        4.61 NA
## 27   ZDB107      38000  Cit+ REL606  plus SRR098038        4.79 NA
## 28   ZDB111      38000    C2 REL606 minus SRR098039        4.62 NA
## 29 REL10979      40000  Cit+ REL606  plus SRR098029        4.78 NA
## 30 REL10988      40000    C2 REL606 minus SRR098030        4.62 NA
## 31                  NA                                       NA NA
```

```
# Let's try the read.csv() version
Ecoli_metadata_csv <- read.csv("data/Ecoli_metadata_d1.csv", header = TRUE)
tail(Ecoli_metadata_csv)
```

```
##      sample                        generation clade strain   cit       run
## 26    ZDB99                             36000    C1 REL606 minus SRR098037
## 27   ZDB107                             38000  Cit+ REL606  plus SRR098038
## 28   ZDB111                             38000    C2 REL606 minus SRR098039
## 29 REL10979                             40000  Cit+ REL606  plus SRR098029
## 30 REL10988                             40000    C2 REL606 minus SRR098030
## 31          #This comment is highly important
##    Genome.size  X
## 26        4.61 NA
## 27        4.79 NA
## 28        4.62 NA
## 29        4.78 NA
## 30        4.62 NA
## 31          NA NA
```

```
Ecoli_metadata_csv <- read.csv("data/Ecoli_metadata_d1.csv", header = TRUE,
                               comment.char = '#', sep = ",")
tail(Ecoli_metadata_csv)
```

```
##      sample generation clade strain   cit       run Genome.size  X
## 26    ZDB99      36000    C1 REL606 minus SRR098037        4.61 NA
## 27   ZDB107      38000  Cit+ REL606  plus SRR098038        4.79 NA
## 28   ZDB111      38000    C2 REL606 minus SRR098039        4.62 NA
## 29 REL10979      40000  Cit+ REL606  plus SRR098029        4.78 NA
## 30 REL10988      40000    C2 REL606 minus SRR098030        4.62 NA
## 31                  NA                                       NA NA
```

Activity 2

a) More on importing files:

- a.1) Import Ecoli_metadata_d2.csv using read.csv() to a table named Ecoli_metadata_csv2. Print the first 8 rows of Ecoli_metadata_csv and Ecoli_metadata_csv2. Explain what happened.
- a.2) Ecoli_metadata_csv has one empty column and one empty and one empty row, why? Remove this column and row (use indexes)?
- a.3) Calculate the mean for variable Genome.size. How can you solve the problem found?
- a.4) With function grepl() you can search for patterns in a vector. How can you use this function to filter Ecoli_metadata_csv to get only the lines of samples starting with "ZDB"?
- *hint: grep("pattern", vector1) would retrieve the index of vector elements contaning the word "pattern)"*

Before moving forward, let's delete all working environment

```
# using RStudio, just click in the 'broom' button on the top right panel, or use ...

rm(list = ls())
```

1.4. Factors and missing data

```
feeders <- read.csv("data/Feeder_observations.csv", header=T, check.names = FALSE)
```

In this project with hummingbirds, authors aimed to study the effect of food provisioning by humans on animal nutrition and exposure to novel pathogens. By experimentally manipulating bird visitation to feeders, authoers compared microbial communities and sucrose solution characteristics over time and compared this to hummingbird bill and faecal samples, and to floral nectar.

README_for_Feeder_observations.docx has the metadata for Feeder_observations.csv

```r
# re-check the elements in
str(feeders)
```

```
## 'data.frame':    411 obs. of  18 variables:
##  $ TRIAL #                   : int  2 2 2 2 2 2 2 2 2 1 ...
##  $ DATE                      : Factor w/ 17 levels "10-Jul-17","11-Jul-17",..: 12 12 13 13 14 14 17 1
##  $ TIME                      : Factor w/ 36 levels ".","17:32","18:05",..: 23 4 28 13 22 4 24 11 16 3
##  $ SITE                      : Factor w/ 3 levels ".","Glide","Rita": 2 2 2 2 2 2 2 2 2 3 ...
##  $ DURATION OF GROWTH (days): int  0 0 1 1 2 2 4 4 7 0 ...
##  $ FEEDER #                  : Factor w/ 21 levels ".","1","10","11",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ TREATMENT                 : Factor w/ 4 levels ".","cage","net",..: 2 2 2 2 2 2 2 2 2 3 ...
##  $ OBSERVER                  : Factor w/ 3 levels ".","CGL","TH": 3 3 3 2 3 3 3 3 3 2 ...
##  $ # HUMM                    : Factor w/ 43 levels ".","0","1","10",..: 2 2 2 2 2 2 2 2 1 2 ...
##  $ # INSECT                  : Factor w/ 2 levels ".","0": 2 2 2 2 2 2 2 2 1 2 ...
##  $ FEEDER WT (g)             : Factor w/ 159 levels "",".","1009",..: 122 2 120 2 109 2 92 1 69 84 ..
##  $ FEED TEMP                 : Factor w/ 62 levels "",".","52.7",..: 2 2 18 2 27 2 16 2 29 2 ...
##  $ ENV TEMP                  : Factor w/ 2 levels "",".": 1 1 1 1 1 1 1 1 1 1 ...
##  $ NOTES                     : Factor w/ 41 levels "",".","<2' left in feeder",..: 1 28 1 1 1 1 1 1 2
##  $ pH                        : Factor w/ 13 levels "",".","2.5","2.75",..: 2 2 9 2 9 2 8 2 7 2 ...
##  $ OD_1                      : Factor w/ 80 levels ".","0.033","0.0331",..: 1 1 49 1 29 1 41 1 62 1 .
##  $ OD_2                      : Factor w/ 82 levels ".","0.0334","0.0341",..: 1 1 25 1 20 1 33 1 46 1
##  $ OD_avg                    : Factor w/ 106 levels ".","0.0336","0.03385",..: 1 1 48 1 31 1 47 1 74
```

Most elements are classified as **Factors**:

Factors are used to represent categorical data. Factors can be ordered (e.g.; small, medium, large) or unordered (e.g.; blond, brunette, ginger) and are an important class for statistical analysis and for plotting.

Once created, factors can only contain a pre-defined set of values known as levels (stored as numeric and for instance, important for plots, regression analysis, . . .). By default, R always sorts levels in alphabetical order.

Most columns containing character data are considered factors by default when imported in a table (use stringsAsFactors = FALSE to avoid this, if necessary)

```r
# first let's change the column names to less 'error-prone' names

names(feeders)
```

```
##  [1] "TRIAL #"                   "DATE"
##  [3] "TIME"                      "SITE"
##  [5] "DURATION OF GROWTH (days)" "FEEDER #"
##  [7] "TREATMENT"                 "OBSERVER"
##  [9] "# HUMM"                    "# INSECT"
## [11] "FEEDER WT (g)"             "FEED TEMP"
## [13] "ENV TEMP"                  "NOTES"
## [15] "pH"                        "OD_1"
## [17] "OD_2"                      "OD_avg"
```

```r
names(feeders)[c(1,5,6,9,10,11,12,13)] <- c("TRIAL","DurGROWTH","FEED.N","N.HUMM","N.INSECT",
                                            "FEED.WT","FEED.TEMP", "ENV.TEMP")

# when applying str() to the feeders we see a lot of columns with factors, but some should
# be numeric
```

```
summary(feeders$N.HUMM)
```

```
##   .   0   1  10  11  12  13  14  15  17  18  19   2  21  22  23  24  25
##  63 244  15   4   2   2   3   2   1   1   3   4  15   1   1   3   1   2
##  28  29   3  30  31  32  33  35  36  37  38  39   4  41  46  51  56  58
##   2   1   5   2   3   2   1   1   1   1   2   1   6   1   1   1   1   1
##  59   6  67   7   8  81   9
##   1   3   1   1   2   1   2
```

```
# to convert a factor to a numeric, first convert to character and then to numeric

feeders$N.HUMM <- as.numeric(as.character(feeders$N.HUMM))
```

```
## Warning: NAs introduced by coercion
```

```
summary(feeders$N.HUMM) # function summary for a numeric vector should give this output
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##   0.000   0.000   0.000   4.773   2.000  81.000      63
```

```
# other variables should be transformed too
feeders$FEED.TEMP <- as.numeric(as.character(feeders$FEED.TEMP))
```

```
## Warning: NAs introduced by coercion
```

```
feeders$pH <- as.numeric(as.character(feeders$pH))
```

```
## Warning: NAs introduced by coercion
```

```
feeders$OD_1 <- as.numeric(as.character(feeders$OD_1))
```

```
## Warning: NAs introduced by coercion
```

```
feeders$OD_2 <- as.numeric(as.character(feeders$OD_2))
```

```
## Warning: NAs introduced by coercion
```

```
feeders$OD_avg <- as.numeric(as.character(feeders$OD_avg))
```

```
## Warning: NAs introduced by coercion
```

```
feeders$FEED.WT <- as.numeric(as.character(feeders$FEED.WT))
```

```
## Warning: NAs introduced by coercion
```

```
feeders$NOTES <- as.character(feeders$NOTES)
feeders$DurGROWTH <- as.factor(feeders$DurGROWTH)

# when applying str() to the feeders the variable TREATMENT is described as
# a Factor w/ 4 levels: '.', 'cage', 'net', 'open'

levels(feeders$TREATMENT)
```

```
## [1] "."    "cage" "net"  "open"
```

```
# the '.' factor in this dataset is related to 'not available' data and should
# not be considered. Therefore we need to convert it to NA

levels(feeders$TREATMENT)[1]<- NA

# to re-order the factor and set a reference
```

```r
feeders$TREATMENT <- relevel(feeders$TREATMENT, ref="open")

#other factors need some revision too, due to the amount of missing data
levels(feeders$SITE)[1]<- NA
levels(feeders$OBSERVER)[1]<- NA
levels(feeders$N.INSECT)[1]<- NA
levels(feeders$TIME)[1]<- NA

# Create a new feeders table with data from both sites (Glinda and Rita) and 2 DurGrowth levels (1 and

feeders_select <- feeders[feeders$SITE %in% c("Glinda","Rita") & feeders$DurGROWTH %in% c(1,7), ]

# when filtering a data-set with factor, these need to be adjusted

levels(feeders_select$DurGROWTH)
```

```
## [1] "0" "1" "2" "3" "4" "7"
```

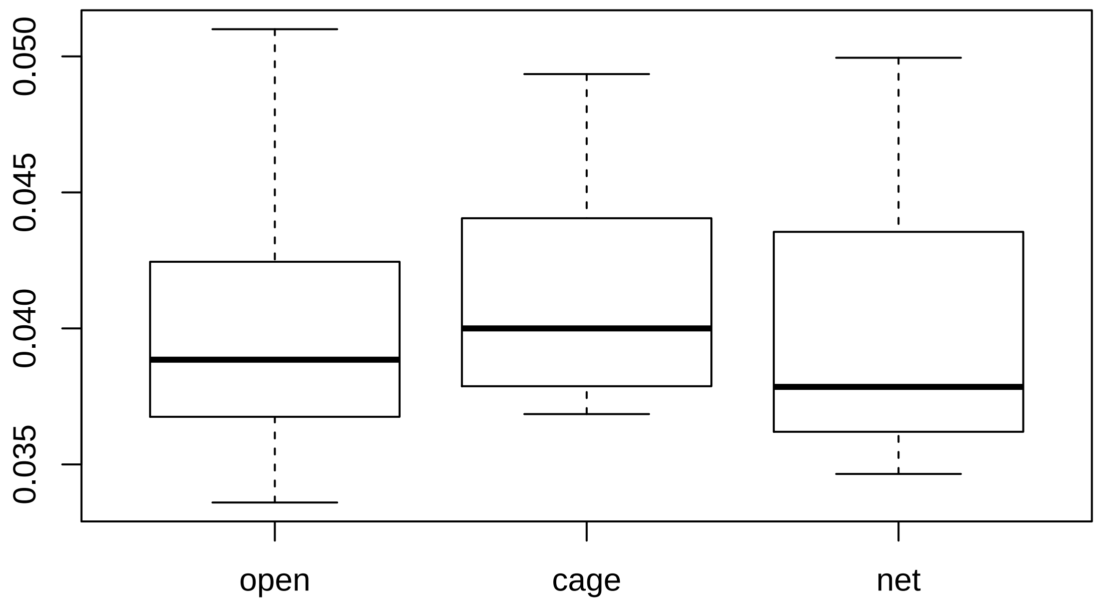
```r
levels(feeders_select$SITE)
```

```
## [1] "Glide" "Rita"
```

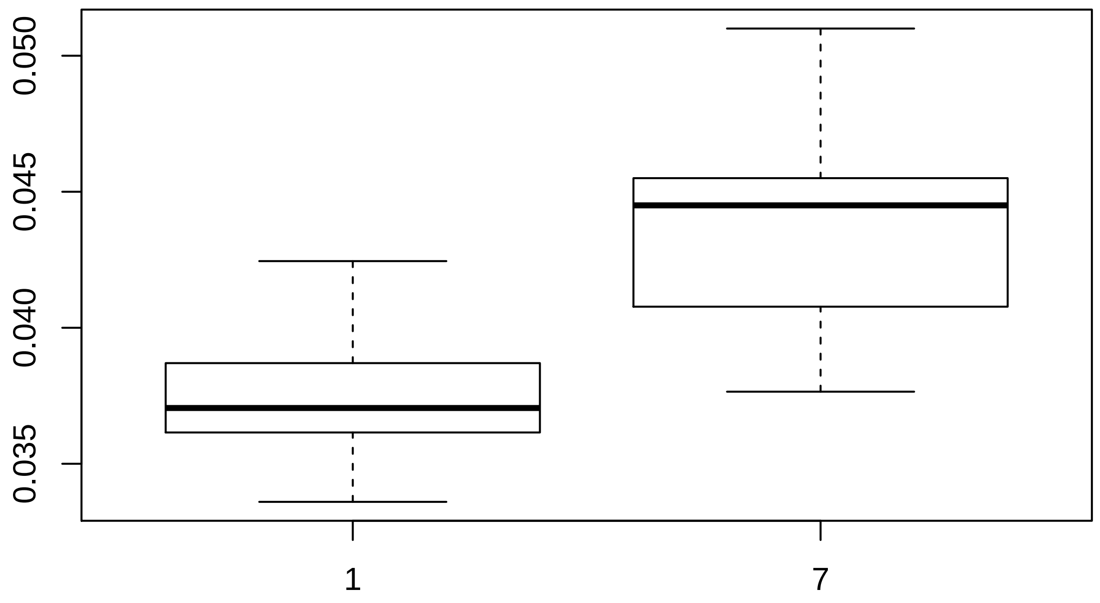
```r
levels(feeders_select$TREATMENT)
```

```
## [1] "open" "cage" "net"
```

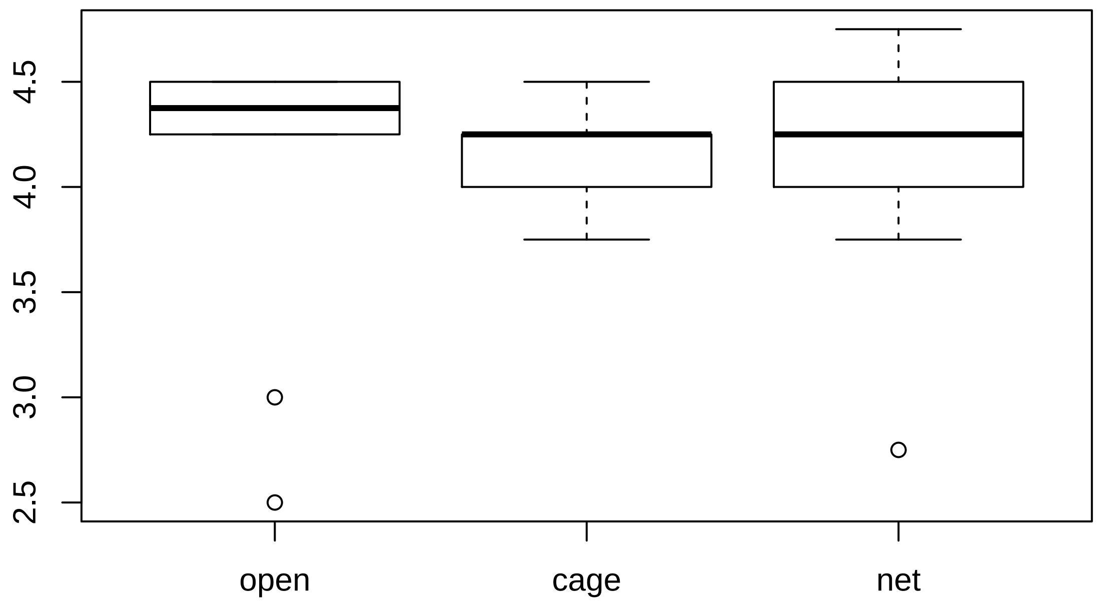
```r
# the new feeders_select only has two levels in DurGROWTH, yet 7 levels are shown.
# the new feeders_select only has three levels in TRE, yet 7 levels are shown.
# therefore we need to ajust the factors

feeders_select$DurGROWTH <- factor(feeders_select$DurGROWTH)
feeders_select$TREATMENT <- factor(feeders_select$TREATMENT)
# we could also adjust for other factors, but these are the only one
# we're interested for statistical analysis

# to save all R objects obtained so far
save.image(file = 'feeders.RData')

boxplot(formula = OD_avg ~ TREATMENT,data = feeders_select )
```



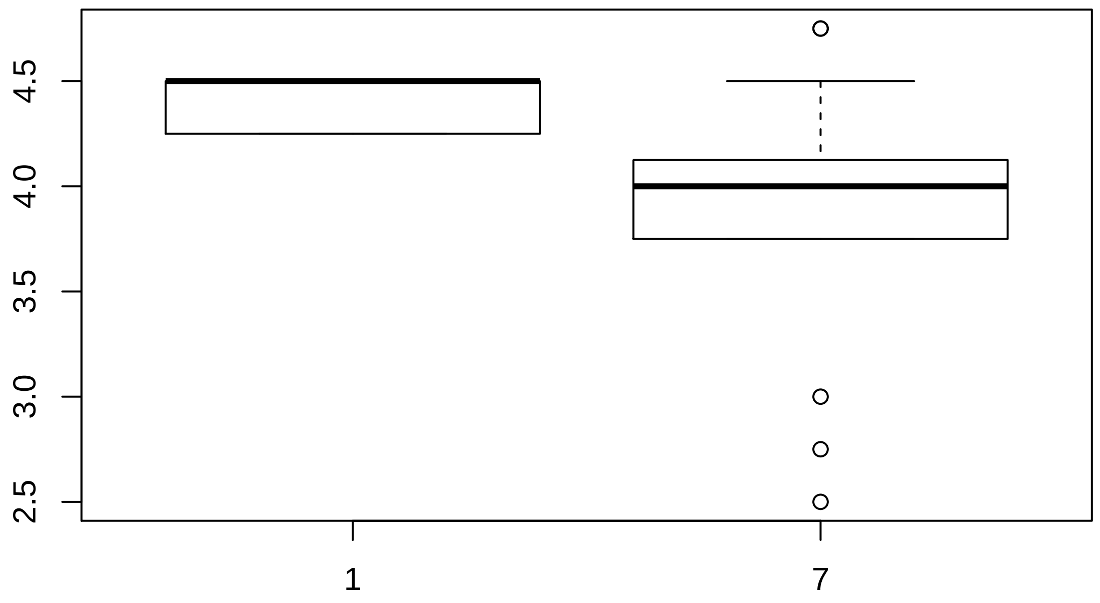
```r
boxplot(formula = OD_avg ~ DurGROWTH,data = feeders_select )
```



```r
boxplot(formula = pH ~ TREATMENT,data = feeders_select )
```



```
boxplot(formula = pH ~ DurGROWTH,data = feeders_select )
```


4.5
4.0
3.5
3.0
2.5
1
7