Trait-based approach in ecology

Aurele Toussaint (aurele.toussaint@cnrs.fr)

Autumn 2024

Contents

Scripts and data in R

Working directory

In which folder (working directory) you have your files?

```
wd=getwd() # get working directory
wd
## [1] "/Users/aurele/Library/CloudStorage/Dropbox/courses/M2TULIP/R_script/Data"
setwd("/Users/aurele/Library/CloudStorage/Dropbox/courses/M2TULIP/R_script/Data") # change to your favo
dir() # lists files
##  [1] "AVONET.xlsx"       "basal_area.R"      "bry.tab.txt"
##  [4] "climate.txt"       "clusters.rda"      "community.rda"
##  [7] "coordinates.txt"   "Data.Rproj"        "droughtnet.txt"
## [10] "eesti reljeef.DAT"
##  [ reached getOption("max.print") -- omitted 32 entries ]
```

Sometimes it is needed to give double backslashes: `"c:\\mp\\doc\\"`

Keeping track of objects in the memory, saving R objects

You can check what objects you currently have and remove unnecessary.

```
# Making some objects
a=3
diversity=c(18,27,10,22,25,8,12)
soil.ph=c(4.5,6.4,3.9,5.3,5.9,3.2,4.2)
soil.type=c("mineral","mineral","peat","mineral","mineral","peat","peat")

ls()
## [1] "a"         "diversity" "soil.ph"   "soil.type" "wd"
ls.str() # more information
## a :  num 3
## diversity :  num [1:7] 18 27 10 22 25 8 12
## soil.ph :  num [1:7] 4.5 6.4 3.9 5.3 5.9 3.2 4.2
## soil.type :  chr [1:7] "mineral" "mineral" "peat" "mineral" "mineral" "peat" "peat"
## wd :  chr "/Users/aurele/Library/CloudStorage/Dropbox/courses/M2TULIP/R_script/Data"

save(diversity,soil.ph,soil.type,file="my.data.Rda") # saving so tahat R can use later
## NB! Other software cannot read these files!

rm(diversity,soil.ph,soil.type) # removes objects not needed
ls()
## [1] "a"  "wd"
load("my.data.Rda") # loads what has been saved in R
ls()
## [1] "a"         "diversity" "soil.ph"   "soil.type" "wd"
```

Explore with `?rm` how to remove all objects in the memory. WARNING, use carefully!

Answer

```
# currently we can still use this. Sometimes it is good to start your session with that expression
rm(list=ls())

# We now need to load our saved data again:
load("my.data.Rda")
```

Creating own scripts

Scripts are just text files with codes. In RStudio you can add new R Script file. When saving, use extension `.R`, for example `my_first_script.R`

Add comments for your own, and for others with `#` mark

Run your script by highlighting a part and then Cntr+Enter (=run!)

```
## My script,

# a is a variable ....
a = 5

print("Hello! This is my first script")
## [1] "Hello! This is my first script"

## end of my script
```

Save the text above to a text file with name "my_first_script.R".

Now we can call the same script.

```
a=NA
a
## [1] NA

source("my_first_script.R") # runs your script from the file
## Warning in file(filename, "r", encoding = encoding): cannot open file
## 'my_first_script.R': No such file or directory
## Error in file(filename, "r", encoding = encoding): cannot open the connection

a # a has been defined in the script!
## [1] NA
```

Conditions in scripts

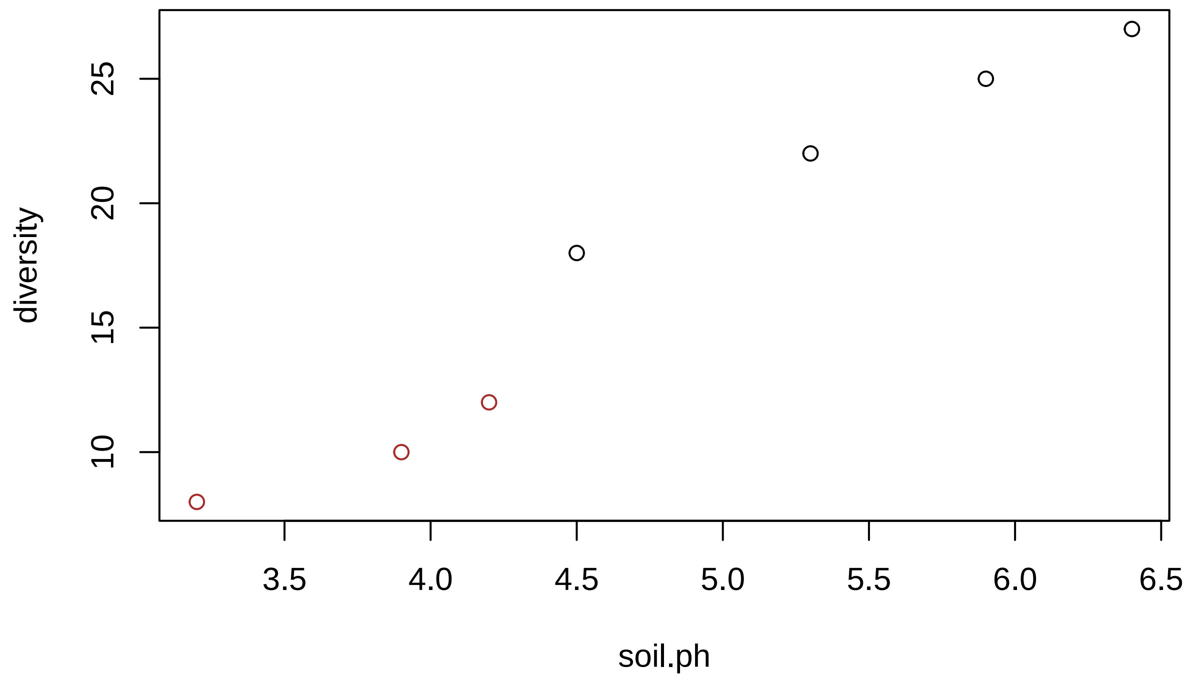
Does something only if conditions are fulfilled. The proportion of script for which the condition applies can be given between special parentheses `{` and `}`

```
soil="peat" # try to change to "mineral"

if (soil=="peat") {       # commands for conditions are between {}
  print("Has been wet")
}
## [1] "Has been wet"

answer=ifelse(soil.type=="peat","wet","dry") # conditions, if true, if not
print(answer)
## [1] "dry" "dry" "wet" "dry" "dry" "wet" "wet"
```

> Use function `ifelse` with `plot` to have scatterplot between `soil.ph` and `diversity` and select symbol colors depending on `soil.type`

Answer

```
plot(soil.ph,diversity,col=ifelse(soil.type=="peat","brown","black"))
```



Loops in scripts

Loops will make a similar thing n times, for example with each element of a vector. The part of script which is repeated should be marked with `{` and `}`. Actually in R often you can avoid loops since you can have logical questions to whole vectors.

```
for (i in 1:7) { # i is index the loop is between {}
  print(soil.type[i]) # within loop you need to add print to get output
}
## [1] "mineral"
## [1] "mineral"
## [1] "peat"
## [1] "mineral"
## [1] "mineral"
## [1] "peat"
## [1] "peat"

for (i in soil.type) { # now i is an element of the vector
  print(i)
}
## [1] "mineral"
## [1] "mineral"
## [1] "peat"
## [1] "mineral"
## [1] "mineral"
## [1] "peat"
## [1] "peat"

soils.high.div=NULL   # NULL is empty object, but we can define it!
for (i in 1:length(diversity)) {
  if (diversity[i]>20) soils.high.div=c(soils.high.div,soil.type[i])
}
soils.high.div
## [1] "mineral" "mineral" "mineral"
```

Please make `soils.high.div` without loop.

Answer

```
soils.high.div=soil.type[diversity>20] # if possible, avoid loops!
```

Data frames

We can have a single table which can include different types of data (e.g. numeric and character). We can also specify a grouping variable factor. When createing a data.frame, most non-numerical components are defined as factors. You can access individual columns of data.frame by name or by number or defining with data_frame_name$column_name

```
my.data=data.frame(soil.ph,soil.type,diversity) # creating data frame
my.data
##   soil.ph soil.type diversity
## 1     4.5   mineral        18
## 2     6.4   mineral        27
## 3     3.9      peat        10
## 4     5.3   mineral        22
## 5     5.9   mineral        25
## 6     3.2      peat         8
## 7     4.2      peat        12
str(my.data) # Note that in data-frame texts are transformed to factors with numeric levels!
## 'data.frame':    7 obs. of  3 variables:
##  $ soil.ph  : num  4.5 6.4 3.9 5.3 5.9 3.2 4.2
##  $ soil.type: chr  "mineral" "mineral" "peat" "mineral" ...
##  $ diversity: num  18 27 10 22 25 8 12

my.data[,2] # accessing by rows and columns
## [1] "mineral" "mineral" "peat"    "mineral" "mineral" "peat"    "peat"

names(my.data)
## [1] "soil.ph"   "soil.type" "diversity"
colnames(my.data) # same!
## [1] "soil.ph"   "soil.type" "diversity"
my.data[,"soil.type"]  # accessing a column by name
## [1] "mineral" "mineral" "peat"    "mineral" "mineral" "peat"    "peat"
my.data$soil.type # accessing by $ symbol
## [1] "mineral" "mineral" "peat"    "mineral" "mineral" "peat"    "peat"

as.character(my.data$soil.type) # transforming factors to characters
## [1] "mineral" "mineral" "peat"    "mineral" "mineral" "peat"    "peat"
as.numeric(my.data$soil.type) # to text: factor levels!
## Warning: NAs introduced by coercion
## [1] NA NA NA NA NA NA NA

my.data$high.diversity=my.data$diversity>20 # adds a new logical column
my.data
##   soil.ph soil.type diversity high.diversity
## 1     4.5   mineral        18          FALSE
## 2     6.4   mineral        27           TRUE
## 3     3.9      peat        10          FALSE
## 4     5.3   mineral        22           TRUE
## 5     5.9   mineral        25           TRUE
## 6     3.2      peat         8          FALSE
## 7     4.2      peat        12          FALSE
```

Reading and exploring data from text file

We generally read data from files. We can read text files by setting column limits (space, comma, semicolon etc.), defining headers (column titles), column for row names etc. etc.

Save file ´envir.txt´to your R working directory!

```
envir=read.table("envir.txt",header=T, sep=" ",row.names = 1)

# Let's explore the file!

ncol(envir) # number of columns
## [1] 12
nrow(envir) # ... rows
## [1] 30
```

```
dim(envir)  # dimensions
## [1] 30 12
names(envir) # names of columns (colnames works as well)
##  [1] "lon"          "lat"          "forest.types" "Jrk..nr."     "Proovi.nimi"
##  [6] "pH.KCl"       "N.."          "P.mg.kg"      "K.mg.kg"      "Ca.mg.kg"
## [11] "Mg.mg.kg"     "OA.."
row.names(envir) # if there are row names
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15"
## [16] "16" "17" "18" "19" "20" "21" "26" "27" "28" "31" "32" "33" "36" "44" "73"
str(envir)
## 'data.frame':    30 obs. of  12 variables:
##  $ lon         : num  26.5 26.4 26.4 26.3 26.2 ...
##  $ lat         : num  58.2 58.2 58.2 58.2 58.2 ...
##  $ forest.types: int  1152 1131 1153 1512 1132 1132 1161 1162 1132 1161 ...
##  $ Jrk..nr.    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Proovi.nimi : chr  "Vapramae" "Illi" "Vitipalu" "Konguta" ...
##  $ pH.KCl      : num  3.72 3.65 4.25 4.4 2.62 3.28 5.77 4.01 2.51 4.04 ...
##  $ N..         : num  0.425 0.136 0.25 2.319 0.37 ...
##  $ P.mg.kg     : num  31.6 22.3 58 63 25.6 ...
##  $ K.mg.kg     : num  79 42.7 60.7 131.2 131.6 ...
##  $ Ca.mg.kg    : num  370 154 649 11208 282 ...
##  $ Mg.mg.kg    : num  43.9 21.6 83.3 601.5 74.6 ...
##  $ OA..        : num  11.4 5.24 7.78 70.85 24.83 ...
head(envir) # just upper rows
##        lon      lat forest.types Jrk..nr. Proovi.nimi pH.KCl       N..  P.mg.kg
## 1 26.45699 58.24387         1152        1    Vapramae   3.72 0.4246261 31.57133
## 2 26.43730 58.19846         1131        2        Illi   3.65 0.1364021 22.26459
## 3 26.41474 58.15658         1153        3    Vitipalu   4.25 0.2500752 58.04788
## 4 26.27875 58.21060         1512        4     Konguta   4.40 2.3192906 63.00445
## 5 26.15339 58.22432         1132        5     Vehendi   2.62 0.3698321 25.62339
## 6 26.32119 58.28426         1132        6      Erumae   3.28 0.2426881 16.21594
##     K.mg.kg   Ca.mg.kg  Mg.mg.kg      OA..
## 1  79.04454   370.4051  43.87278 11.397929
## 2  42.73691   154.3332  21.64893  5.237742
## 3  60.71264   648.9929  83.34118  7.777811
## 4 131.18302 11208.0226 601.45503 70.845348
## 5 131.56542   281.8415  74.61169 24.833289
## 6  66.16427   260.2084  25.95761  7.988277
summary(envir) # short summary of each variable
##       lon             lat         forest.types     Jrk..nr.
##  Min.   :26.11   Min.   :58.09   Min.   :1131   Min.   : 1.00
##  1st Qu.:26.44   1st Qu.:58.17   1st Qu.:1132   1st Qu.: 8.25
##  Median :26.78   Median :58.24   Median :1147   Median :15.50
##  Mean   :26.77   Mean   :58.22   Mean   :1194   Mean   :18.70
##  3rd Qu.:27.02   3rd Qu.:58.27   3rd Qu.:1161   3rd Qu.:26.75
##  Max.   :27.42   Max.   :58.36   Max.   :1512   Max.   :73.00
##  Proovi.nimi            pH.KCl           N..              P.mg.kg
##  Length:30          Min.   :2.510   Min.   :0.1189   Min.   : 8.215
##  Class :character   1st Qu.:2.965   1st Qu.:0.2207   1st Qu.:12.889
##  Mode  :character   Median :3.525   Median :0.2575   Median :17.712
##                     Mean   :3.593   Mean   :0.3812   Mean   :25.488
##                     3rd Qu.:4.010   3rd Qu.:0.3751   3rd Qu.:31.160
##                     Max.   :5.770   Max.   :2.3193   Max.   :65.774
##      K.mg.kg          Ca.mg.kg           Mg.mg.kg           OA..
##  Min.   : 32.93   Min.   :   80.24   Min.   : 18.71   Min.   : 4.871
##  1st Qu.: 66.69   1st Qu.:  225.05   1st Qu.: 43.89   1st Qu.: 7.045
##  Median : 80.91   Median :  430.02   Median : 76.38   Median : 8.408
##  Mean   : 87.97   Mean   :  817.52   Mean   :101.06   Mean   :12.627
##  3rd Qu.: 98.67   3rd Qu.:  617.13   3rd Qu.:105.18   3rd Qu.:11.693
##  Max.   :167.19   Max.   :11208.02   Max.   :601.46   Max.   :70.845

envir[1:4,6:12] # short subset
##   pH.KCl       N..  P.mg.kg   K.mg.kg    Ca.mg.kg  Mg.mg.kg      OA..
## 1   3.72 0.4246261 31.57133  79.04454   370.4051  43.87278 11.397929
## 2   3.65 0.1364021 22.26459  42.73691   154.3332  21.64893  5.237742
## 3   4.25 0.2500752 58.04788  60.71264   648.9929  83.34118  7.777811
## 4   4.40 2.3192906 63.00445 131.18302 11208.0226 601.45503 70.845348

my.cols=c(6:9,5)  # pH, N, P, K and site name
soil.data=envir[,my.cols] # subset by selected columns to new data.frame
head(envir[,-c(1:2,4,6:12)]) ## leaving out some columns (not "-" before "c")
##   forest.types Proovi.nimi
## 1         1152    Vapramae
## 2         1131        Illi
## 3         1153    Vitipalu
## 4         1512     Konguta
## 5         1132     Vehendi
## 6         1132      Erumae

my.rows=envir$pH.KCl>median(envir$pH.KCl) # logical vector
my.rows
##  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE
## [13]  TRUE FALSE  TRUE FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE
## [25]  TRUE FALSE  TRUE FALSE FALSE  TRUE
high.ph.soils=soil.data[my.rows,]
low.ph.soils=soil.data[!my.rows,] # note !

high.ph.soils.1=subset(soil.data, pH.KCl>median(pH.KCl)) # subset function

high.ph.soils==high.ph.soils.1 # comparing all values individually
##    pH.KCl  N.. P.mg.kg K.mg.kg Proovi.nimi
## 1    TRUE TRUE    TRUE    TRUE        TRUE
## 2    TRUE TRUE    TRUE    TRUE        TRUE
## 3    TRUE TRUE    TRUE    TRUE        TRUE
## 4    TRUE TRUE    TRUE    TRUE        TRUE
## 7    TRUE TRUE    TRUE    TRUE        TRUE
## 8    TRUE TRUE    TRUE    TRUE        TRUE
## 10   TRUE TRUE    TRUE    TRUE        TRUE
## 13   TRUE TRUE    TRUE    TRUE        TRUE
## 15   TRUE TRUE    TRUE    TRUE        TRUE
## 18   TRUE TRUE    TRUE    TRUE        TRUE
## 20   TRUE TRUE    TRUE    TRUE        TRUE
## 26   TRUE TRUE    TRUE    TRUE        TRUE
## 31   TRUE TRUE    TRUE    TRUE        TRUE
## 33   TRUE TRUE    TRUE    TRUE        TRUE
## 73   TRUE TRUE    TRUE    TRUE        TRUE
```

```
all.equal(high.ph.soils,high.ph.soils.1) # comparing two objects
## [1] TRUE

sample(nrow(soil.data)) # random order
##  [1] 29 26 19 18  8 22  7 15  6 13 17 30 23  9  1 24  2 25  5 28 10 14 21  4 20
## [26] 27 11 12  3 16

rand.soils=soil.data[sample(nrow(soil.data)),]
```

> make a separate data.frame xy for coordinates lon and lat and site name.

Answer

```
xy=envir[,c(1,2,5)]
```

Writing data tables to text files

```
write.table(xy,file="coordinates.txt") # default settings

write.table(soil.data,file="soil.data.csv", sep=";", row.names = F) # own settings
```

R packages

R has some basic packages but you need to install packages for special tasks. Let's install package "readxl" which allows to read excel files.

```
# install.packages("readxl") # if you do not have this, please omit # and install

library(readxl) # even if installed, you should call the package library each R session with the functi

test=read_excel("excel.xlsx")
test # a bit strange format
## # A tibble: 7 x 3
##   soil.ph soil.type diversity
##     <dbl> <chr>         <dbl>
## 1     4.5 mineral          18
## 2     6.4 mineral          27
## 3     3.9 peat             10
## 4     5.3 mineral          22
## 5     5.9 mineral          25
## 6     3.2 peat              8
## 7     4.2 peat             12
test=as.data.frame(test) # making a siple data.frame
test
##   soil.ph soil.type diversity
## 1     4.5   mineral        18
## 2     6.4   mineral        27
## 3     3.9      peat        10
## 4     5.3   mineral        22
## 5     5.9   mineral        25
## 6     3.2      peat         8
## 7     4.2      peat        12
```

Manipulating data tables

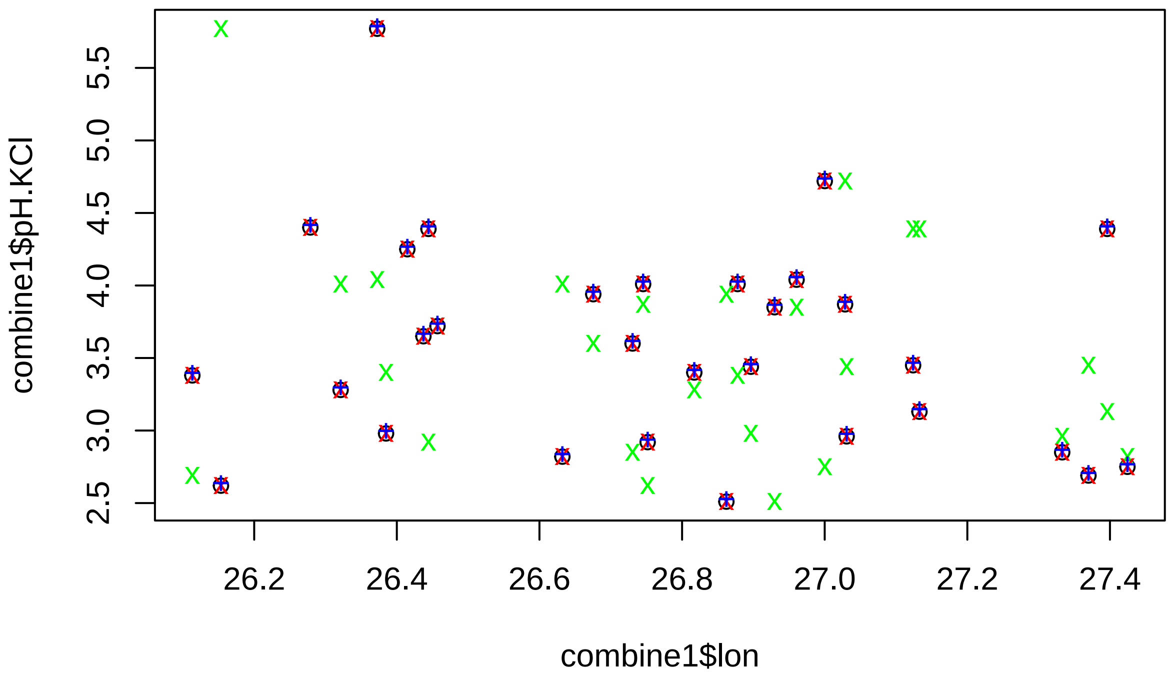
How to merge tables and how to split tables.

Merging data tables

Be very careful when merging data tables by combining them by columns: the order of the rows must be the same in both tables! It is a good practice to try to make some graphs to visualise and check manually some values to see that everything matches.

When reading files, be sure that these files are present in the working directory

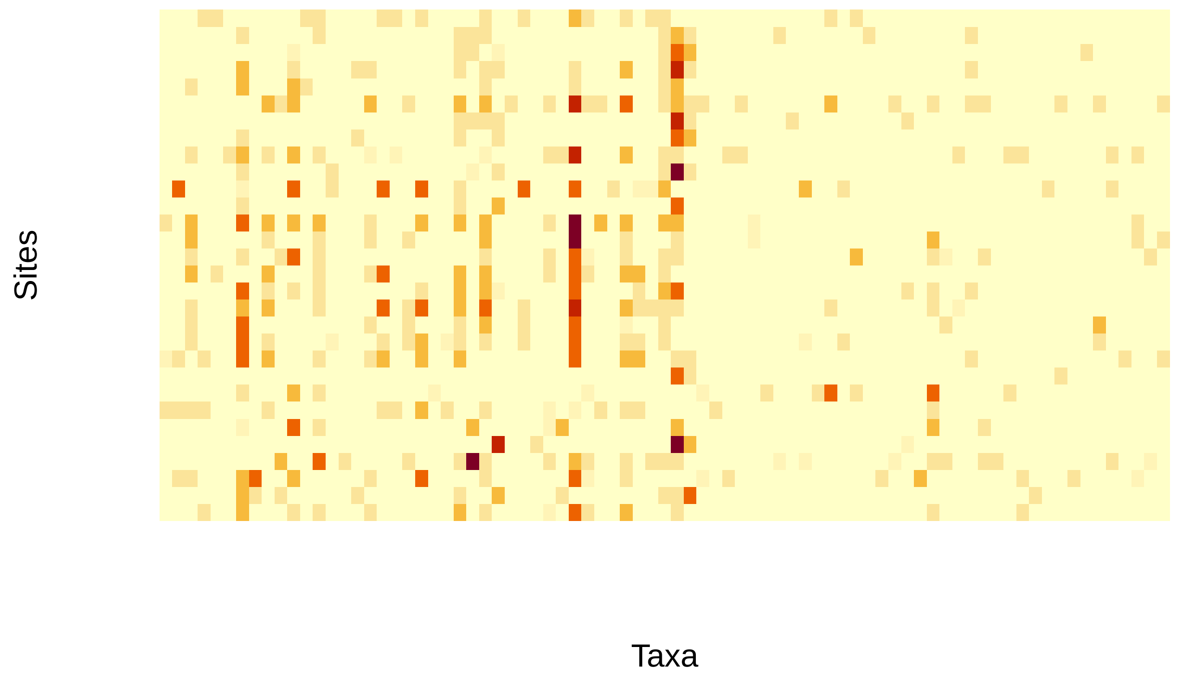
```
# Cleaning the memory before starting
rm(list=ls())

## Reading files we saved last time:
xy = read.table("coordinates.txt") # default settings
soil.data = read.table("soil.data.csv", sep = ";", header = T) # defining delimiting character (here ";

## As we did last time, let's split soil data into two using the function subset
high.ph.soils = subset(soil.data, pH.KCl > median(pH.KCl))
low.ph.soils = subset(soil.data, pH.KCl <= median(pH.KCl))

combine1 = cbind(xy, soil.data) # putting together columns
head(combine1)
##        lon      lat Proovi.nimi pH.KCl       N..  P.mg.kg   K.mg.kg Proovi.nimi
## 1 26.45699 58.24387    Vapramae   3.72 0.4246261 31.57133  79.04454    Vapramae
## 2 26.43730 58.19846        Illi   3.65 0.1364021 22.26459  42.73691        Illi
## 3 26.41474 58.15658    Vitipalu   4.25 0.2500752 58.04788  60.71264    Vitipalu
## 4 26.27875 58.21060     Konguta   4.40 2.3192906 63.00445 131.18302     Konguta
## 5 26.15339 58.22432     Vehendi   2.62 0.3698321 25.62339 131.56542     Vehendi
## 6 26.32119 58.28426      Erumae   3.28 0.2426881 16.21594  66.16427      Erumae

combine2 = rbind(high.ph.soils, low.ph.soils) # putting together rows
dim(combine2)
## [1] 30  5

combine3 = cbind(xy, combine2) # trying again, but ....
plot(combine1$lon, combine1$pH.KCl)
points(combine3$lon, combine3$pH.KCl, pch = "x", col = "green") # adding points, symbol "x"
head(combine3) # note that site names from xy and new soil data do not match!!
##        lon      lat Proovi.nimi pH.KCl       N..   P.mg.kg   K.mg.kg
## 1 26.45699 58.24387    Vapramae   3.72 0.4246261 31.571334  79.04454
## 2 26.43730 58.19846        Illi   3.65 0.1364021 22.264589  42.73691
## 3 26.41474 58.15658    Vitipalu   4.25 0.2500752 58.047875  60.71264
## 4 26.27875 58.21060     Konguta   4.40 2.3192906 63.004453 131.18302
## 5 26.15339 58.22432     Vehendi   5.77 0.2557405  8.214731  99.41711
## 6 26.32119 58.28426      Erumae   4.01 0.2749614 13.283927  64.78593
##   Proovi.nimi
## 1    Vapramae
## 2        Illi
## 3    Vitipalu
## 4     Konguta
## 5    Porgumae
## 6      Aardla

# The 'merge' function can help in merging two datasets that share a common column with unique identifi
combine3 = merge(xy, combine2, by = "Proovi.nimi") # merging by common column
head(combine3)
##   Proovi.nimi      lon      lat pH.KCl       N..    P.mg.kg   K.mg.kg
## 1      Aardla 26.74540 58.32910   4.01 0.2749614 13.283927  64.78593
## 2       Aravu 27.42453 58.25486   2.75 0.4787677 24.440573 107.38674
## 3      Erumae 26.32119 58.28426   3.28 0.2426881 16.215942  66.16427
## 4  Haavametsa 27.36984 58.26345   2.69 0.1794443  8.860196  65.97427
## 5      Ignase 26.81712 58.26367   3.40 0.2200201 37.057869  79.17404
## 6        Illi 26.43730 58.19846   3.65 0.1364021 22.264589  42.73691
points(combine3$lon, combine3$pH.KCl, pch = "x", col = "red") # now OK!

# an alternative with the function 'order':
combine4 = cbind(xy[order(xy$Proovi.nimi), ], combine2[order(combine2$Proovi.nimi), ])
points(combine4$lon, combine4$pH.KCl, pch = "+", col = "blue")
```



Community data: sites x taxa matrix

Community data is usually a matrix of **taxa x sites**. In R it is tradition to have rows as sites (samples) and columns as species (taxa). In the field often the opposite direction is used in books, so be careful with this when transcribing your data into R. Here we read a table of forest floor vascular plants from 10x10 m plots.

```
vas.plants = read.table("vascular.plants.txt")

dim(vas.plants)
## [1] 79 30

vas.plants[1:5, 1:5] # taxa as rows, sites as columns
##            X001Vapramae X002Illi X003Vitipalu X004Konguta X005Vehendi
## ACTAspic              0        0            0           0           0
## AEGOpoda              0        0            1           0           0
## ANEMnemo              0        0            1           0           0
## ASAReuro              1        0            0           0           0
## ATHYfil.fe            0        0            0           0           0

vas.plants = t(vas.plants) # transposing the matrix (replacing rows and columns)

vas.plants = as.data.frame(vas.plants) # transposing loose data.frame sturcture, putting back
vas.plants[1:5, 1:5]
##              ACTAspic AEGOpoda ANEMnemo ASAReuro ATHYfil.fe
## X001Vapramae        0        0        0        1          0
## X002Illi            0        0        0        0          0
## X003Vitipalu        0        1        1        0          0
## X004Konguta         0        0        0        0          0
## X005Vehendi         0        0        0        0          0

image(t(vas.plants),
      axes = F,
      ylab = "Sites",
      xlab = "Taxa") ## Quick visualisation of a table
```



```
apply(vas.plants, 1, FUN = max)  # a function can be applied to each row (1) or column (2). Here we fin
##   X001Vapramae        X002Illi   X003Vitipalu     X004Konguta      X005Vehendi
##              3               3              3               5                5
##     X006.Erumae    X007Porgumae       X008Aardla    X009Kurepalu   X010Sarakuste
##              3               2              3               3                3
##       X011Kannu       X012Vonnu       X013Rookse      X014Kammeri       X015Kambja
##              3               3              4               3                3
##       X016Reola      X017Ignase      X018Unikula X019Haavametsa X020Kruusahaua
##              3               5              5               3                3
##       X021Aravu    X026Pedajamae        X027Miti     X028Puhaste    X031Pimmelaan
##              5               4              3               4                4
##      X032Logina   X033Voorepalu X036Taevaskoja     X044Savimae X073Pausakunnu
##              2               4              3               2                2
```

Use function `apply` and calculate max values for each taxon in the object `vas.plants`.

Answer

```
apply(vas.plants, 2, FUN = max)
##   ACTAspic   AEGOpoda   ANEMnemo   ASAReuro ATHYfil.fe   BRACpinn   CALAarun
##        1.0        3.0        2.0        1.0        1.0        1.0        3.0
##   CALLvulg   CAREdigi   CIRCalpi   CONVmaja   DESCflex   DRYOcart   DRYOexpa
##        3.0        2.0        2.0        3.0        1.0        3.0        1.0
##   EPILangu   FESTovin   FRAGvesc   GALElute   GERAsylv   GYMNdryo   HEPAnobi
##        1.0        1.0        2.0        3.0        1.0        1.0        3.0
##   IMPAnoli   LATHvern   LUZUpilo   LYCOanno   MAIAbifo   MELAprat   MELInuta
##        0.5        1.0        2.0        5.0        3.0        4.0        1.0
##   MILIeffu   MOLIcaer   MYCEmura   ORTHsecu   OXALacet   PARIquad   PTERaqui
##        3.0        1.0        1.0        2.0        5.0        1.0        2.0
##   RANUcass   RUBUsaxa   STELholo   STELnemo   TRIEeuro   VACCmyrt VACCvit.id
##        1.0        3.0        2.0        1.0        2.0        5.0        3.0
##   VEROcham   VIOLmira   ANGEsylv   ANTHsylv   CAMPpers   CAREcane   CAREglob
##        1.0        1.0        1.0        1.0        0.5        1.0        1.0
##   CHIMumbe   CREPpalu   DESCcaes   DRYOfili   EQUIprat   EQUIsylv   GALEOPsp
##        1.0        2.0        1.0        3.0        1.0        2.0        1.0
##  GALIalbu.   GALIpalu   GOODrepe   HIERvulg   IMPAparv   LYSIvulg   MELAnemo
##        1.0        1.0        1.0        2.0        3.0        1.0        1.0
##   MELAsylv   MOERtrin   PHEGconn    POAnemo   POLYodor   POTEerec   PULMobsc
##        1.0        1.0        1.0        1.0        1.0        1.0        1.0
##   PYROmino  RANUauri.   SCORhumi   SOLIvirg   URTIdioi   VEROoffi   VIOLcani
##        1.0        1.0        1.0        2.0        1.0        1.0        1.0
##   VIOLepip  VIOLrivi.
##        1.0        1.0
```

Long data format

Data matrices might have too many zeros since most taxa are rare. Because of this, a "long" format, with columns indicating site, taxon and some abundance measure is often used. Here we read an example of long format from trees in forest sites from an excel file.

```
library(readxl) # package to read excel files

data = read_excel("trees.xlsx")
data = as.data.frame(data)      # to make the file data.frame
head(data)
##         ala nr rinne puuliik   D1   D2    H
## 1 008Aardla  1     1      PN   43 41.5 30.1
## 2 008Aardla  2     1      TA 30.5 31.5 21.5
## 3 008Aardla  3     1      HB   19 19.5 21.3
## 4 008Aardla  4     2      TM    7  8.0 10.4
## 5 008Aardla  5     2      TM  7.5  7.0 11.0
## 6 008Aardla  6     2      TM    8  7.0 12.4

# ala = site, puuliik = species (codes of tree species)

tree.counts = table(data[, c(1, 4)])  # counts of occurrences (number of trees)
tree.height = xtabs(H ~ ala + puuliik, data = data) # sum of H (height)

str(tree.counts) # from function table we obtain a bit special format, we better make it a simple data.
##  'table' int [1:30, 1:12] 0 0 0 0 0 0 0 3 0 0 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ ala    : chr [1:30] "001Vapramäe" "002Illi" "003Vitipalu" "004Konguta" ...
##   ..$ puuliik: chr [1:12] "HB" "KS" "KU" "LH" ...
tree.counts = as.data.frame.matrix(tree.counts)
str(tree.counts)
## 'data.frame':    30 obs. of  12 variables:
##  $ HB: int  0 0 0 0 0 0 0 3 0 0 ...
##  $ KS: int  0 1 3 0 0 0 0 1 0 2 ...
##  $ KU: int  3 1 8 8 0 8 5 0 3 6 ...
##  $ LH: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ MA: int  3 4 1 2 5 2 0 0 3 1 ...
##  $ MN: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ PI: int  0 0 0 0 0 0 1 0 0 0 ...
##  $ PN: int  0 0 0 0 0 0 0 1 0 0 ...
##  $ RE: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ TA: int  0 0 0 0 0 0 1 1 0 0 ...
##  $ TM: int  0 0 0 0 0 0 0 4 0 0 ...
##  $ VA: int  0 0 0 0 0 0 0 0 0 0 ...
```

NA in data

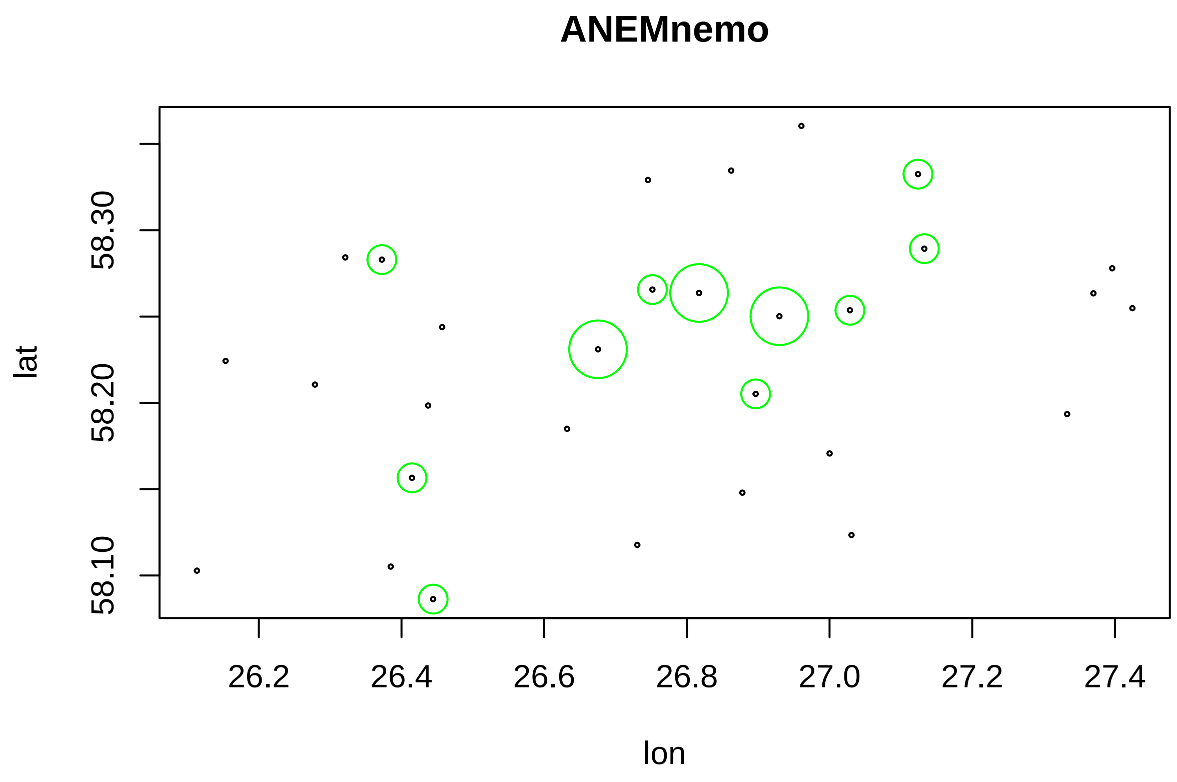
For missing values in a table, `NA` is automatically used.

```
summary(data) # Note D2
##      ala                  nr             rinne         puuliik
##  Length:239         Min.   : 1.000   Min.   :1.000   Length:239
##  Class :character   1st Qu.: 3.000   1st Qu.:1.000   Class :character
##  Mode  :character   Median : 5.000   Median :1.000   Mode  :character
##                     Mean   : 5.188   Mean   :1.439
##                     3rd Qu.: 7.000   3rd Qu.:2.000
##                     Max.   :16.000   Max.   :2.000
##
##       D1                  D2              H
##  Length:239         Min.   : 5.20   Min.   : 6.00
##  Class :character   1st Qu.:12.00   1st Qu.:13.10
##  Mode  :character   Median :24.50   Median :22.00
##                     Mean   :24.96   Mean   :22.40
##                     3rd Qu.:37.75   3rd Qu.:30.85
##                     Max.   :54.00   Max.   :44.30
##                     NA's   :9
complete.cases(data) # do we have real values for a row
##   [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
##  [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE
##  [25]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
##  [37]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
##  [49]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [61]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [73]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [97]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [109]  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## [133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [145]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [157]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [169]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [181]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [193]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [205]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [217]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
## [229]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE

data[!complete.cases(data), ] # not complete cases (note "!")
##               ala nr rinne puuliik     D1 D2    H
## 11    073Sulaoja  1     1      MA   ü177 NA 24.9
## 20    073Sulaoja 11     1      MA   ü189 NA 23.9
## 21     015Kambja  1     1      KU ü166.5 NA 35.0
## 24     015Kambja  5     1      KS   ü229 NA 36.4
## 36   007Põrgumäe  1     1      TA   ü174 NA 33.0
## 44      016Reola  4     1      KU   ü248 NA 34.6
## 77   003Vitipalu  9     1      MA    ü82 NA 36.8
## 113   028Puhaste  2     1      MA   ü203 NA 34.9
## 129 031Pimmelaan  2     1      MA   ü199 NA 36.4

is.na(data$D2) # where we have NA values?
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE  TRUE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

Combining community and site data

Before we analyse community and site data together, it is suggested to check if we have the same order in both tables. We can list rownames and site names side by side

```
# Checking that the order is the same

cbind(row.names(vas.plants), as.character(xy$Proovi.nimi))
##        [,1]              [,2]
##   [1,] "X001Vapramae"    "Vapramae"
##   [2,] "X002Illi"        "Illi"
##   [3,] "X003Vitipalu"    "Vitipalu"
##   [4,] "X004Konguta"     "Konguta"
##   [5,] "X005Vehendi"     "Vehendi"
##   [6,] "X006.Erumae"     "Erumae"
##  [7,] "X007Porgumae"   "Porgumae"
##  [8,] "X008Aardla"     "Aardla"
##  [9,] "X009Kurepalu"   "Kurepalu"
## [10,] "X010Sarakuste"  "Sarakuste"
## [11,] "X011Kannu"      "Kannu"
## [12,] "X012Vonnu"      "Vonnu"
## [13,] "X013Rookse"     "Rookse"
## [14,] "X014Kammeri"    "Kammeri"
## [15,] "X015Kambja"     "Kambja"
## [16,] "X016Reola"      "Reola"
## [17,] "X017Ignase"     "Ignase"
## [18,] "X018Unikula"    "Unikula"
## [19,] "X019Haavametsa" "Haavametsa"
## [20,] "X020Kruusahaua" "Kruusahaua"
## [21,] "X021Aravu"      "Aravu"
## [22,] "X026Pedajamae"  "Pedajamae"
## [23,] "X027Miti"       "Miti"
## [24,] "X028Puhaste"    "Puhaste"
## [25,] "X031Pimmelaan"  "Pimmelaan"
## [26,] "X032Logina"     "Logina"
## [27,] "X033Voorepalu"  "Voorepalu"
## [28,] "X036Taevaskoja" "Taevaskoja"
## [29,] "X044Savimae"    "Savimae"
## [30,] "X073Pausakunnu" "Sulaoja"

# Species distribution maps

i=3  # id of a species
plot(xy[,1:2],cex=0.3) ## Coordinates of sites and small points
points(xy[,1:2], cex=vas.plants[,i]*2,col="green") # cex changes point size relative to 1
title(main=colnames(vas.plants)[i]) # adding title taking the i-th colname from the community data
```



Some text manipulations

If we have names of sites and species, we might also need some text manipulations to cut, paste, search and replace parts of texts.

```
sites1 = row.names(vas.plants)
sites1
##  [1] "X001Vapramae"   "X002Illi"       "X003Vitipalu"   "X004Konguta"
##  [5] "X005Vehendi"    "X006.Erumae"    "X007Porgumae"   "X008Aardla"
##  [9] "X009Kurepalu"   "X010Sarakuste"  "X011Kannu"      "X012Vonnu"
## [13] "X013Rookse"     "X014Kammeri"    "X015Kambja"     "X016Reola"
## [17] "X017Ignase"     "X018Unikula"    "X019Haavametsa" "X020Kruusahaua"
## [21] "X021Aravu"      "X026Pedajamae"  "X027Miti"       "X028Puhaste"
## [25] "X031Pimmelaan"  "X032Logina"     "X033Voorepalu"  "X036Taevaskoja"
## [29] "X044Savimae"    "X073Pausakunnu"

nchar(sites1) # how many characters (length of the text)
##  [1] 12  8 12 11 11 11 12 10 12 13  9  9 10 11 10  9 10 11 14 14  9 13  8 11 13
## [26] 10 13 14 11 14

sites2 = as.character(soil.data$Proovi.nimi) # Making character from a factor data type
sites2
##  [1] "Vapramae"   "Illi"       "Vitipalu"   "Konguta"    "Vehendi"
##  [6] "Erumae"     "Porgumae"   "Aardla"     "Kurepalu"   "Sarakuste"
## [11] "Kannu"      "Vonnu"      "Rookse"     "Kammeri"    "Kambja"
## [16] "Reola"      "Ignase"     "Unikula"    "Haavametsa" "Kruusahaua"
## [21] "Aravu"      "Pedajamae"  "Miti"       "Puhaste"    "Pimmelaan"
## [26] "Logina"     "Voorepalu"  "Taevaskoja" "Savimae"    "Sulaoja"

substring(sites1, 2, 4) # substring, from 2nd to 4th character
##  [1] "001" "002" "003" "004" "005" "006" "007" "008" "009" "010" "011" "012"
## [13] "013" "014" "015" "016" "017" "018" "019" "020" "021" "026" "027" "028"
## [25] "031" "032" "033" "036" "044" "073"

paste(1:30, sites2, sep = "-") # pasting different texts (non-text will be converted)
##  [1] "1-Vapramae"    "2-Illi"        "3-Vitipalu"    "4-Konguta"
##  [5] "5-Vehendi"     "6-Erumae"      "7-Porgumae"    "8-Aardla"
##  [9] "9-Kurepalu"    "10-Sarakuste"  "11-Kannu"      "12-Vonnu"
## [13] "13-Rookse"     "14-Kammeri"    "15-Kambja"     "16-Reola"
## [17] "17-Ignase"     "18-Unikula"    "19-Haavametsa" "20-Kruusahaua"
## [21] "21-Aravu"      "22-Pedajamae"  "23-Miti"       "24-Puhaste"
## [25] "25-Pimmelaan"  "26-Logina"     "27-Voorepalu"  "28-Taevaskoja"
## [29] "29-Savimae"    "30-Sulaoja"


grep("mae", sites1) # numbers of cases with a search string
## [1]  1  6  7 22 29
sites1[grep("mae", sites1)]
## [1] "X001Vapramae"  "X006.Erumae"   "X007Porgumae"  "X026Pedajamae"
## [5] "X044Savimae"
sites1[grep("\\.", sites1)] # some characters as "." "?" have a special meaning but we can search them
## [1] "X006.Erumae"

gsub("mae", "oru", sites1) # replacing a search string
##  [1] "X001Vapraoru"   "X002Illi"       "X003Vitipalu"   "X004Konguta"
##  [5] "X005Vehendi"    "X006.Eruoru"    "X007Porguoru"   "X008Aardla"
##  [9] "X009Kurepalu"   "X010Sarakuste"  "X011Kannu"      "X012Vonnu"
## [13] "X013Rookse"     "X014Kammeri"    "X015Kambja"     "X016Reola"
## [17] "X017Ignase"     "X018Unikula"    "X019Haavametsa" "X020Kruusahaua"
## [21] "X021Aravu"      "X026Pedajaoru"  "X027Miti"       "X028Puhaste"
## [25] "X031Pimmelaan"  "X032Logina"     "X033Voorepalu"  "X036Taevaskoja"
## [29] "X044Savioru"    "X073Pausakunnu"
```

Make a character vector from `sites1` without leading "X"

Answer

```
substring(sites1, 2, 9999)
##  [1] "001Vapramae"   "002Illi"       "003Vitipalu"   "004Konguta"
##  [5] "005Vehendi"    "006.Erumae"    "007Porgumae"   "008Aardla"
##  [9] "009Kurepalu"   "010Sarakuste"  "011Kannu"      "012Vonnu"
## [13] "013Rookse"     "014Kammeri"    "015Kambja"     "016Reola"
## [17] "017Ignase"     "018Unikula"    "019Haavametsa" "020Kruusahaua"
## [21] "021Aravu"      "026Pedajamae"  "027Miti"       "028Puhaste"
## [25] "031Pimmelaan"  "032Logina"     "033Voorepalu"  "036Taevaskoja"
## [29] "044Savimae"    "073Pausakunnu"
```

Manipulating sets

We make a set of forest type codes. In the database we have 4 digits but we use a broader classification of three digits. Then we can find how two sets differ.

```
forest.types = read.table("envir.txt")$forest.types
forest.types
##  [1] 1152 1131 1153 1512 1132 1132 1161 1162 1132 1161 1161 1142 1161 1141 1161
## [16] 1142 1142 1153 1132 1161 1511 1141 1132 1132 1161 1511 1132 1132 1511 1142

forest.types = as.character(forest.types) # numbers do not have meaning, just codes
forest.types
##  [1] "1152" "1131" "1153" "1512" "1132" "1132" "1161" "1162" "1132" "1161"
## [11] "1161" "1142" "1161" "1141" "1161" "1142" "1142" "1153" "1132" "1161"
## [21] "1511" "1141" "1132" "1132" "1161" "1511" "1132" "1132" "1511" "1142"

forest.types = substring(forest.types, 1, 3)
forest.types
##  [1] "115" "113" "115" "151" "113" "113" "116" "116" "113" "116" "116" "114"
## [13] "116" "114" "116" "114" "114" "115" "113" "116" "151" "114" "113" "113"
## [25] "116" "151" "113" "113" "151" "114"

unique(forest.types) # set of unique values
## [1] "115" "113" "151" "116" "114"
table(forest.types) # frequency table
## forest.types
## 113 114 115 116 151
##   9   6   3   8   4

# separating 2 groups according to soil pH
forest.types.1 = forest.types[soil.data$pH.KCl < 3]
forest.types.2 = forest.types[soil.data$pH.KCl >= 3]

forest.types.1
## [1] "113" "113" "114" "114" "113" "151" "113" "113" "151"
forest.types.2
##  [1] "115" "113" "115" "151" "113" "116" "116" "116" "116" "114" "116" "116"
## [13] "114" "115" "116" "114" "113" "116" "151" "113" "114"

union(forest.types.1, forest.types.2) # from two sets (here same as unique of all)
## [1] "113" "114" "151" "115" "116"
intersect(forest.types.1, forest.types.2) # in both
## [1] "113" "114" "151"
setdiff(forest.types.2, forest.types.1) # in set 2 but not in set 1
## [1] "115" "116"
setdiff(forest.types.1, forest.types.2) # None!
## character(0)

is.element(forest.types.2, forest.types.1) # asking which in set 2 are present in set 1
##  [1] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE  TRUE FALSE FALSE
## [13]  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

Making our own function to combine two datasets

Sometimes we want to combine different files which include different sites but species composition do not match. I have not found a good function in R for that. But we can make our own functions!

```r
# We start with some artificial examples, just two species in both sets

t1 = data.frame(a = c(1, 0, 1), b = c(2, 2, 0))
t2 = data.frame(b = c(3, 0), c = c(0, 4))

t1
##   a b
## 1 1 2
## 2 0 2
## 3 1 0
t2
##   b c
## 1 3 0
## 2 0 4



t2[, setdiff(names(t1), names(t2))] = 0 # adding now species only found in table1 to table2 as zeros
t1[, setdiff(names(t2), names(t1))] = 0 # same for table1

t1
##   a b c
## 1 1 2 0
## 2 0 2 0
## 3 1 0 0
t2
##   b c a
## 1 3 0 0
## 2 0 4 0

rbind(t1, t2) # now mergind two tables (rbind looks names!)
##   a b c
## 1 1 2 0
## 2 0 2 0
## 3 1 0 0
## 4 0 3 0
## 5 0 0 4


# Let's make a function from script above!

tables.join = function(t1, t2) {
  # asking two tables which got names t1 and t2
  t2[, setdiff(names(t1), names(t2))] = 0
  t1[, setdiff(names(t2), names(t1))] = 0
  t12 = rbind(t1, t2)
  return(t12) # returns value from function
}
```

```
tables.join(t1, t2)
##   a b c
## 1 1 2 0
## 2 0 2 0
## 3 1 0 0
## 4 0 3 0
## 5 0 0 4

t12 # objects within a functions are not kept in memory!
## Error in eval(expr, envir, enclos): object 't12' not found
```

When using this function you can specify your input objects (let's imagine that these are called `a.table` and `b.table`) either by defining function parameters: `tables.join(t1=a.table,t2=b.table)`, or just by position `tables.join(a.table,b.table)`

Saving community data for the future

```
save(vas.plants,
     tree.counts,
     forest.types,
     xy,
     soil.data,
     tables.join,
     file = "community.rda")
```

Presenting data

Default plots in R are generally very basic and not very suitable for publication. However, one of the characteristics of R is that plots are highly customizable, so that you can basically create any plot you are able to think of *as long as you are able to write the necessary code*. In this lecture we are going to explore a few ways to make our plots look a bit better.

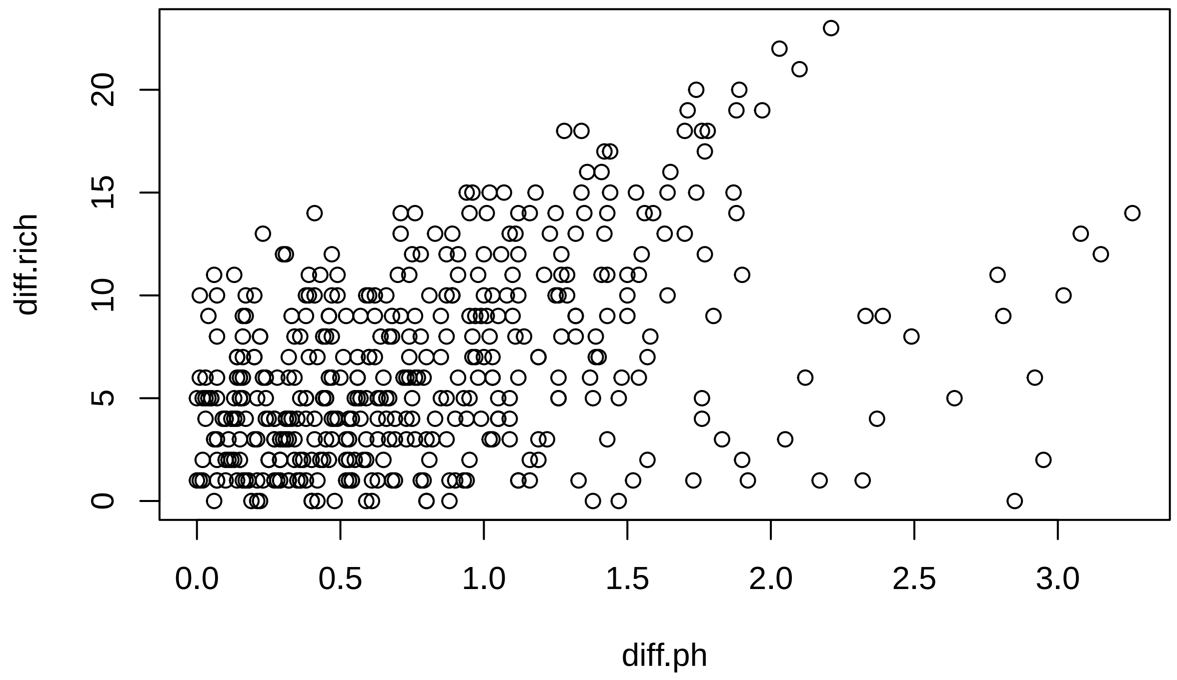
Copy to and from clipboard

```
load("community.rda")
soil.data[1:5,]
##   pH.KCl        N..  P.mg.kg   K.mg.kg Proovi.nimi
## 1   3.72 0.4246261 31.57133  79.04454    Vapramae
## 2   3.65 0.1364021 22.26459  42.73691        Illi
## 3   4.25 0.2500752 58.04788  60.71264    Vitipalu
## 4   4.40 2.3192906 63.00445 131.18302     Konguta
## 5   2.62 0.3698321 25.62339 131.56542     Vehendi
library(clipr)
```

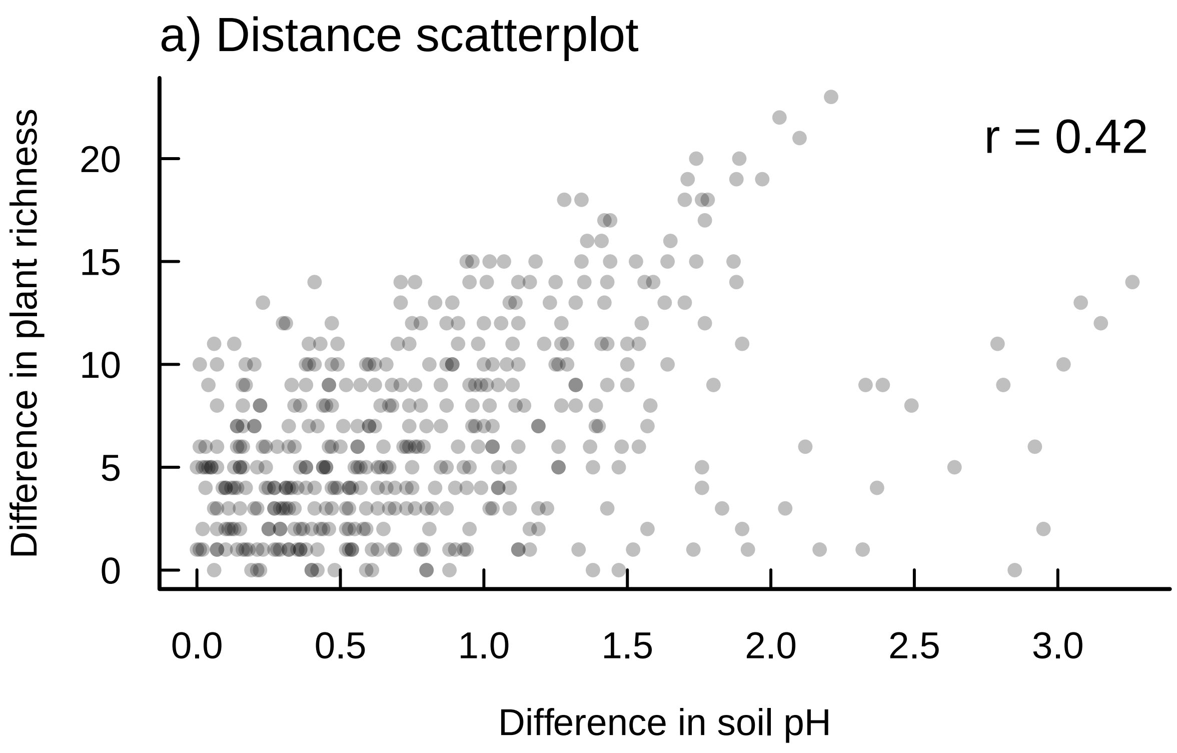
```
write_clip(soil.data[1:5,]) ## you can paste now to excel or word!

#While the write_clip alternative should work for all systems, Windows users can also try this (which d

# write.table(soil.data[1:5,], "clipboard", sep="\t")

#Now we can read from the memory (whatever it is in the computer clipboard):
read.table("clipboard")

o <- cor(soil.data[, -5])
o
write_clip(round(o, 3))

# Paste to your work now!
```

Some tricks and tips with graphs

```
diff.ph <- dist(soil.data$pH.KCl)
diff.rich <- dist(rowSums(vas.plants > 0))

plot(diff.ph, diff.rich) # an ugly plot.
```



```r
# The axis titles are not very informative, and the text is quite small
# Lots of overlapping points, it is hard to see where there is more density of points
# Units in y axis are written in vertical, but horizontal is a bit easier to read (and often required)
# Many times we dont want a full box around our plot
# It would be better to have a title
# We could include some summary statistic, for example the correlation

# Making it better step by step

plot(diff.ph,diff.rich,
     xlab="Difference in soil pH",
     ylab="Difference in plant richness",
     cex.lab=1.2,  # size of axis labels
     pch=16, # dots
     col=rgb(0,0,0,0.25), # grey semi-transparent color
     axes=F) # no axes, we will add them later

axis(side = 1, #Axis 1. 1 means bottom; 2 is left, 3 is top, 4 is right
     lwd = 1.5, #lwd controls axis thickness
     cex.axis = 1.2, # cex.axis change font size
     tcl = 0.5) # tcl determines size and orientation of tickmarks
axis(2, lwd = 1.5, cex.axis = 1.2, las = 1, tcl = 0.5) # same for y axis, las=1 give axis numbers horiz
box(lwd = 2, bty = "l")  # making axes lines

mtext(paste0(letters[1], ") Distance scatterplot"), # adding title (you could  use LETTERS for capital
      side = 3, # same as with axis: 3  is "top"
      cex = 1.5, # size of text
      line = 0.5, #how much above the plot we want it
      adj = 0) # left-justified (0.5  is centered, 1 is right)


r  <- round(cor(diff.ph, diff.rich), 2)

legend("topright", #you can also use x and y coordinates, but this is more  general
       legend = paste("r =", r),
       bty = "n", #we dont want a box around the legend
       cex = 1.5)
```



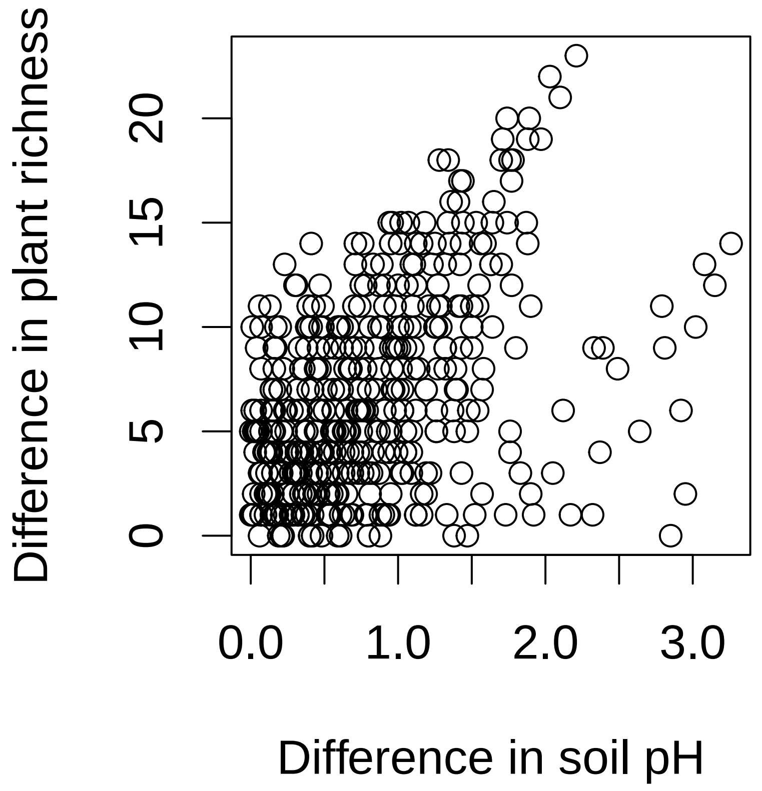
Identifying coordinates and points from graph

```
## Find coordinates by clicking on the graph (number is how many)
locator(1)

## Find a point (same things as in inital plot). Esc for ending
identify(dist(soil.data$pH.KCl), dist(rowSums(vas.plants > 0)))
```

Graph windowns size and margins

```
plot(diff.ph, diff.rich,
     xlab = "Difference in soil pH",
     ylab = "Difference in plant richness")


par(pty = "s") # square graph always

plot(diff.ph, diff.rich,
     xlab = "Difference in soil pH",
     ylab = "Difference in plant richness")
```

```r
par(mar = c(5, 3, 1, 1))

# graph margins in lines form c(bottom, left, top, right).
# Default is c(5, 4, 4, 2) + 0.1

par(cex=1.5) # general size value for all things

plot(diff.ph,diff.rich,
     xlab="Difference in soil pH",
     ylab="Difference in plant richness")
```



```r
par(pty="m") # not square, maximal plotting region

#You can reset graphic settings to default ones (and close the current plot) by executing dev.off()
dev.off()
```

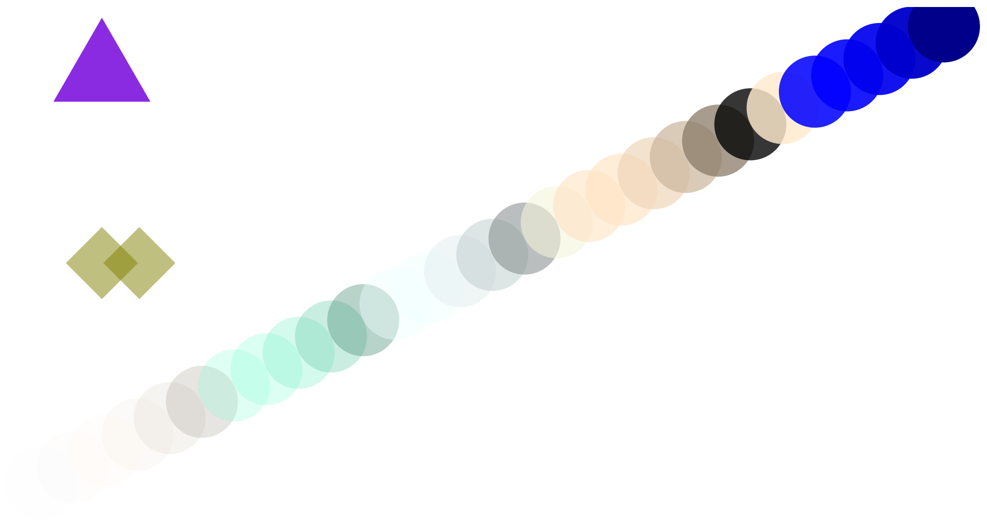
```
## null device
##           1
```

Colors

```r
plot(
  NA,
  xlim = c(0, 1),
  ylim = c(0, 1),
  axes = F,
  xlab = "",
  ylab = ""
) # An empty plot that we will fill
colors()[1:50] # color names
```

```
##  [1] "white"          "aliceblue"      "antiquewhite"   "antiquewhite1"
##  [5] "antiquewhite2"  "antiquewhite3"  "antiquewhite4"  "aquamarine"
##  [9] "aquamarine1"    "aquamarine2"    "aquamarine3"    "aquamarine4"
## [13] "azure"          "azure1"         "azure2"         "azure3"
## [17] "azure4"         "beige"          "bisque"         "bisque1"
## [21] "bisque2"        "bisque3"        "bisque4"        "black"
## [25] "blanchedalmond" "blue"           "blue1"          "blue2"
## [29] "blue3"          "blue4"          "blueviolet"     "brown"
## [33] "brown1"         "brown2"         "brown3"         "brown4"
## [37] "burlywood"      "burlywood1"     "burlywood2"     "burlywood3"
## [41] "burlywood4"     "cadetblue"      "cadetblue1"     "cadetblue2"
## [45] "cadetblue3"     "cadetblue4"     "chartreuse"     "chartreuse1"
## [49] "chartreuse2"    "chartreuse3"
```

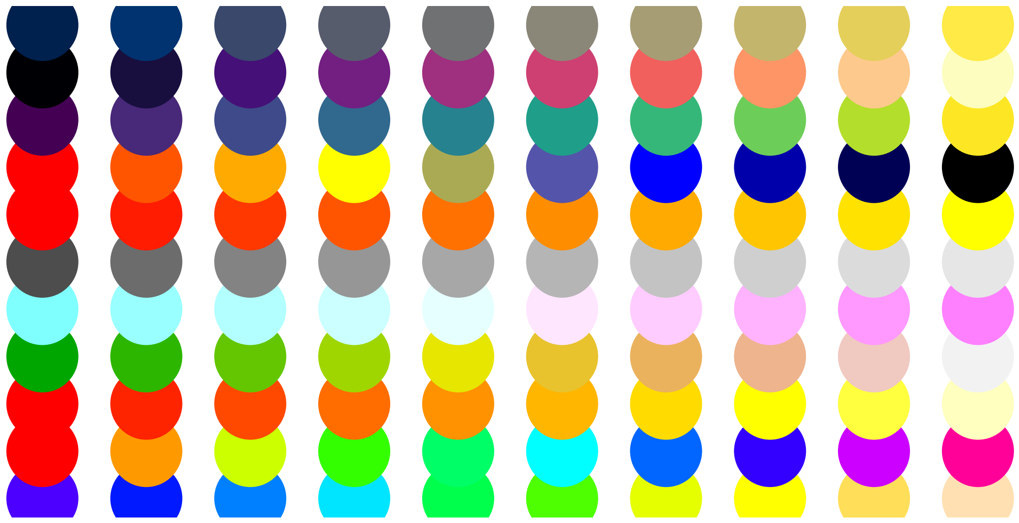
```
colUse <- "blueviolet"
points(0.1, 0.9, pch = 17, cex = 5, col = colUse)

# Transparency by using rgb (red, green, blue) and transparency measure alpha
colUse = rgb(0.5, 0.5, 0, alpha = 0.5)
points(c(0.1, 0.14), c(0.5, 0.5), pch = 18, cex = 5, col = colUse)

# Making a color name semi-transparent
index <- 1
for (i in seq(0, 1, length = 30)) {
  col.t = as.vector(col2rgb(colors()[index])) / 255
  col = rgb(col.t[1], col.t[2], col.t[3], alpha = i)  # alpha is defining transparency (0....1)
  points(i, i, pch = 16, cex = 5, col = col)
  index <- index + 1
}
```



```r
# Color palettes
plot(NA, xlim = c(1, 10), ylim = c(1, 11), axes = F, xlab = "", ylab = "")
topo.colors(10)
```

```
##  [1] "#4C00FF" "#0019FF" "#0080FF" "#00E5FF" "#00FF4D" "#4DFF00" "#E6FF00"
##  [8] "#FFFF00" "#FFDE59" "#FFE0B3"
```

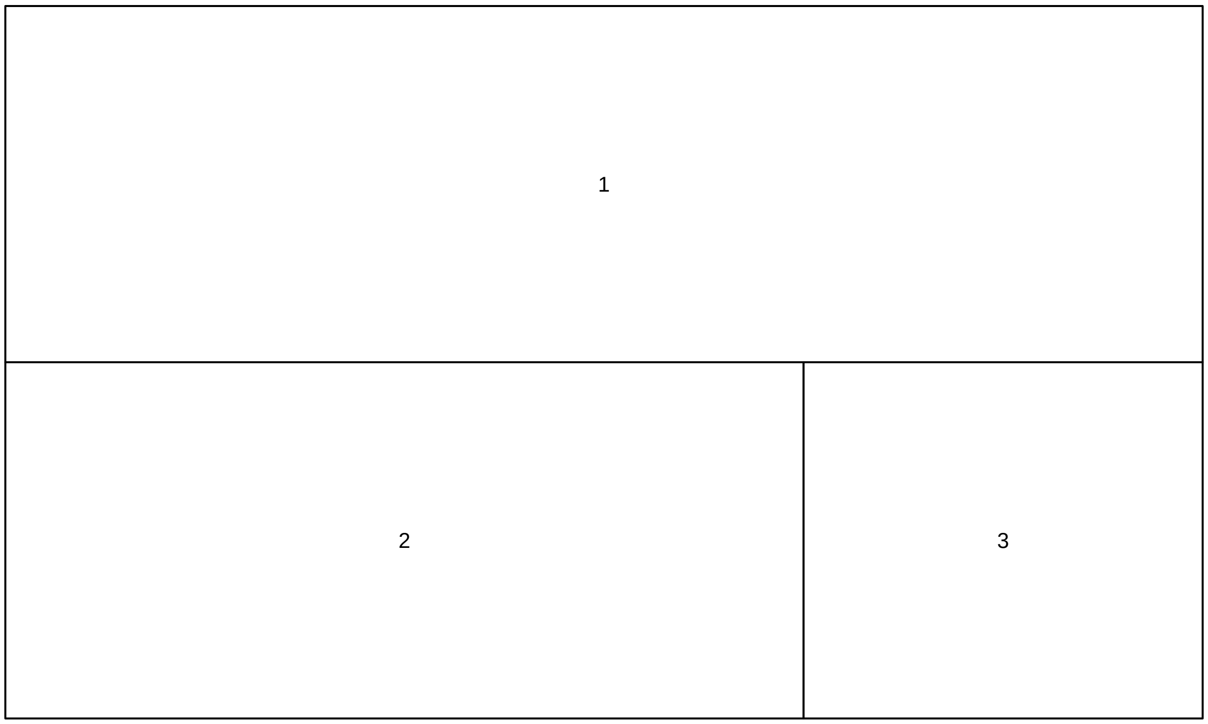
```r
points(1:10, rep(1, 10), col = topo.colors(10)[1:10], pch = 16, cex = 5)
points(1:10, rep(2, 10), col = rainbow(10)[1:10], pch = 16, cex = 5)
points(1:10, rep(3, 10), col = heat.colors(10)[1:10], pch = 16, cex = 5)
points(1:10, rep(4, 10), col = terrain.colors(10)[1:10], pch = 16, cex = 5)
points(1:10, rep(5, 10), col = cm.colors(10)[1:10], pch = 16, cex = 5)
points(1:10, rep(6, 10), col = gray.colors(10)[1:10], pch = 16, cex = 5)
# mixing own colors
points(1:10, rep(7, 10), col = colorRampPalette(c("red", "yellow"))(10)[1:10],
       pch = 16, cex = 5)

points(1:10, rep(8, 10), col = colorRampPalette(c("red", "yellow", "blue", "black"))(10)[1:10],
       pch = 16, cex = 5)

library(viridis)

# visible in grey tones as well! Seen by color-blind people!

points(1:10, rep(9, 10), col = viridis(10)[1:10], pch = 16, cex = 5)
points(1:10, rep(10, 10), col = magma(10)[1:10], pch = 16, cex = 5)
points(1:10, rep(11, 10), col = cividis(10)[1:10], pch = 16, cex = 5)
```



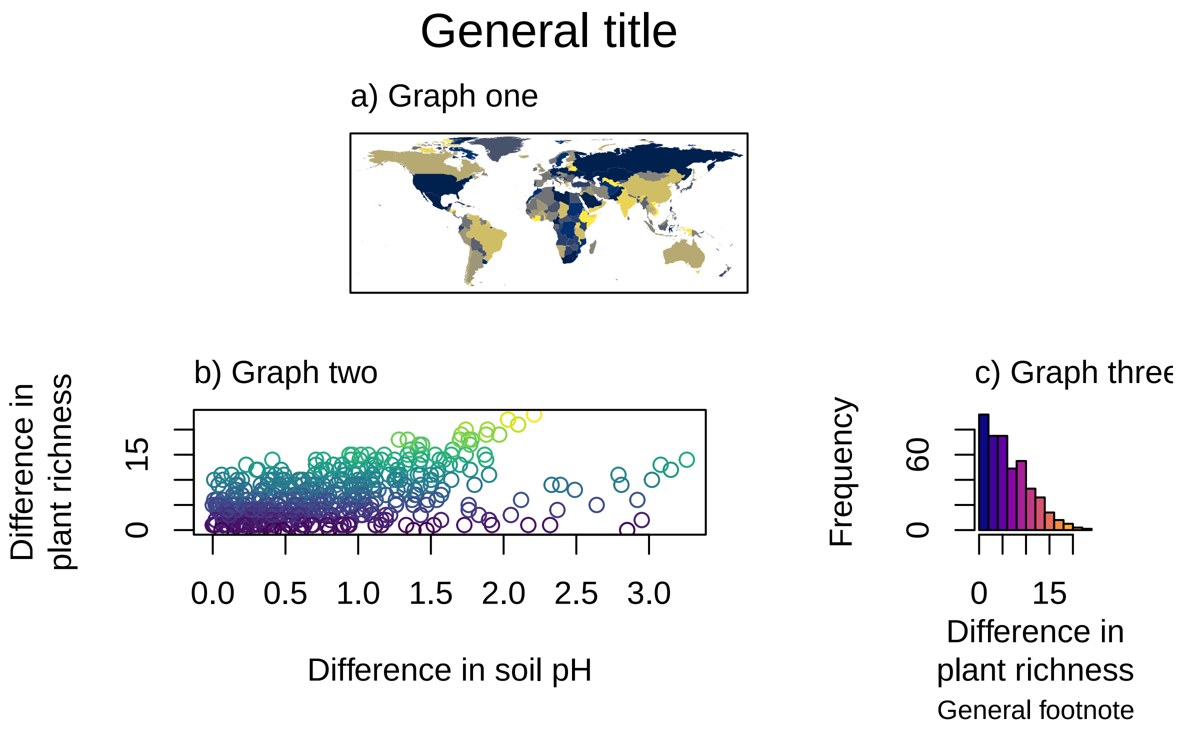
Several graphs together

We have already explored `par(mfrow = c(2, 2))` but sometimes we need more complicated designs.

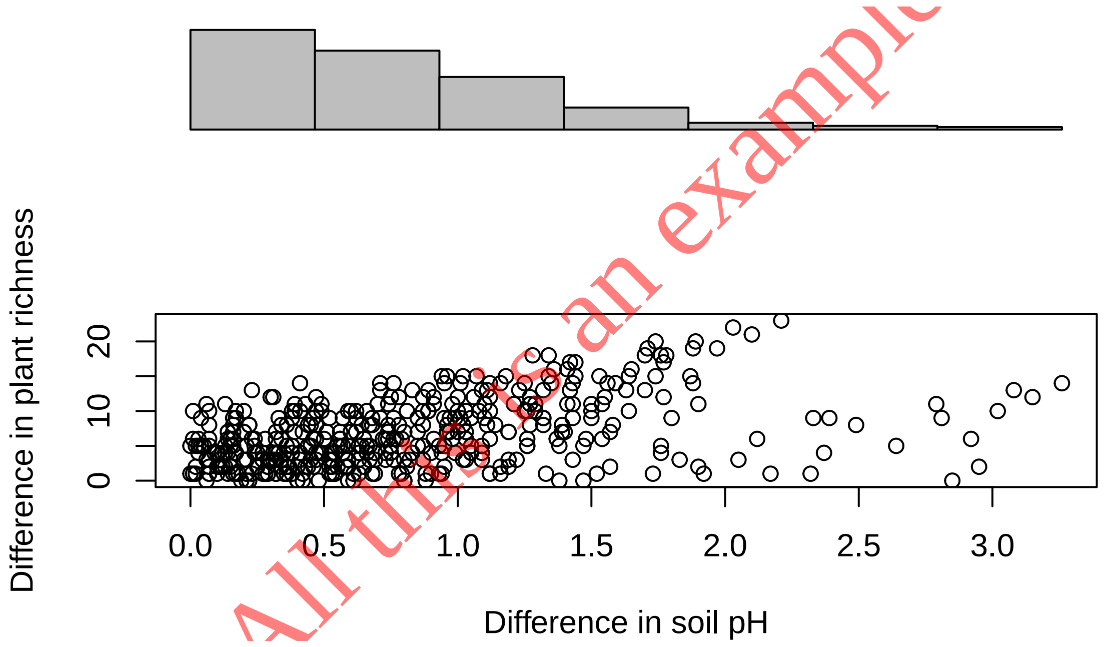
```r
par(oma = c(3, 1, 3, 1)) # margins outside the combined graph (bottom, left, top, right)

# design matrix

m = matrix(c(1, 1, 1,
             2, 2, 3),
           byrow = T,
           nrow = 2, ncol = 3)
m
##      [,1] [,2] [,3]
## [1,]    1    1    1
## [2,]    2    2    3

layout(m)
layout.show(3) # just checking
```



```r
par(cex = 1)   # Select general size factor
par(mar = c(0, 0, 0, 0)) # margins for each of the individual graphs (bottom, left, top, right)

library(maps) # global maps
map("world", fill = T,  col = cividis(12),  border = F,  ylim = c(-60, 85))
box()
mtext("a) Graph one", 3, adj = 0, line = 0.5)

par(mar=c(4,6,1,1)) # we give more space for axis titles
plot(diff.ph,diff.rich,
     xlab="Difference in soil pH",
     ylab="Difference in\nplant richness",
     col=viridis(24)[diff.rich+1])
mtext("b) Graph two",3,adj=0,line=0.5)

hist(diff.rich,main="",xlab="Difference in\nplant richness", col=plasma(12))
mtext("c) Graph three",3,adj=0,line=0.5)

mtext("General title", 3, outer = T, line = 0, cex = 1.5)
mtext("General footnote", 1, outer = T, line = 0, cex = 0.8, adj = 1)
```



```r
layout(1) # ending layout


# Overlapping graphs
par(mar = c(4, 4, 8, 2))
plot(diff.ph, diff.rich,
     xlab = "Difference in soil pH",
     ylab = "Difference in plant richness")

par(fig = c(0, 1, 0.8, 1), new = T) # in relative space of the previous window 0..1
par(mar = c(0, 4, 0.5, 2))
hist(diff.ph, axes = F, xlab = "", ylab = "", main = "", col = "grey")

par(fig = c(0, 1, 0, 1), new = T) # new graph over all

par(mar = c(0, 0, 0, 0)) # no edges
plot(c(0, 1), c(0, 1), ann = F, bty = 'n', type = 'n', xaxt = 'n', yaxt = 'n') # empty graph

text(x = 0.5, y = 0.5, "All this is an example", cex = 3.5, srt = 45,
     col = rgb(1, 0, 0, 0.5), family = "serif")
```



Saving graphs to files

```
pdf("my.graph.pdf", width = 12, height = 12)  # size in inches

## Copy your three graphs here

dev.off() # closing (saving) pdf file
```

```
## pdf
##   2
```

```
png("my.graph.png", width = 6, height = 6, units = "in", res = 600) # resolution dpi

## Copy graphs on top of each other here

dev.off() # closing
```

```
## pdf
##   2
```