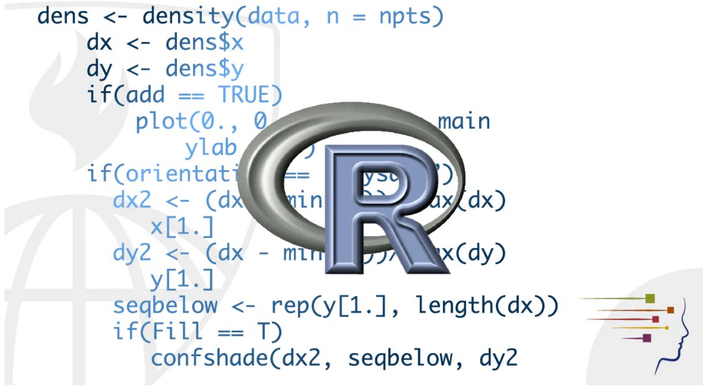
Introduction to R

Anders K. Krabberød (University of Oslo)

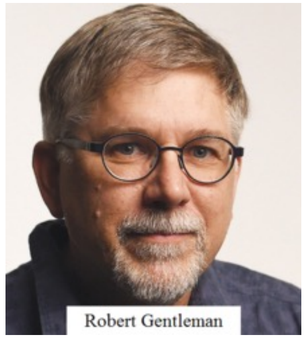
Based on slides by Ramiro Logares (ICM-CSIC, Barcelona)



History of R

John Chambers

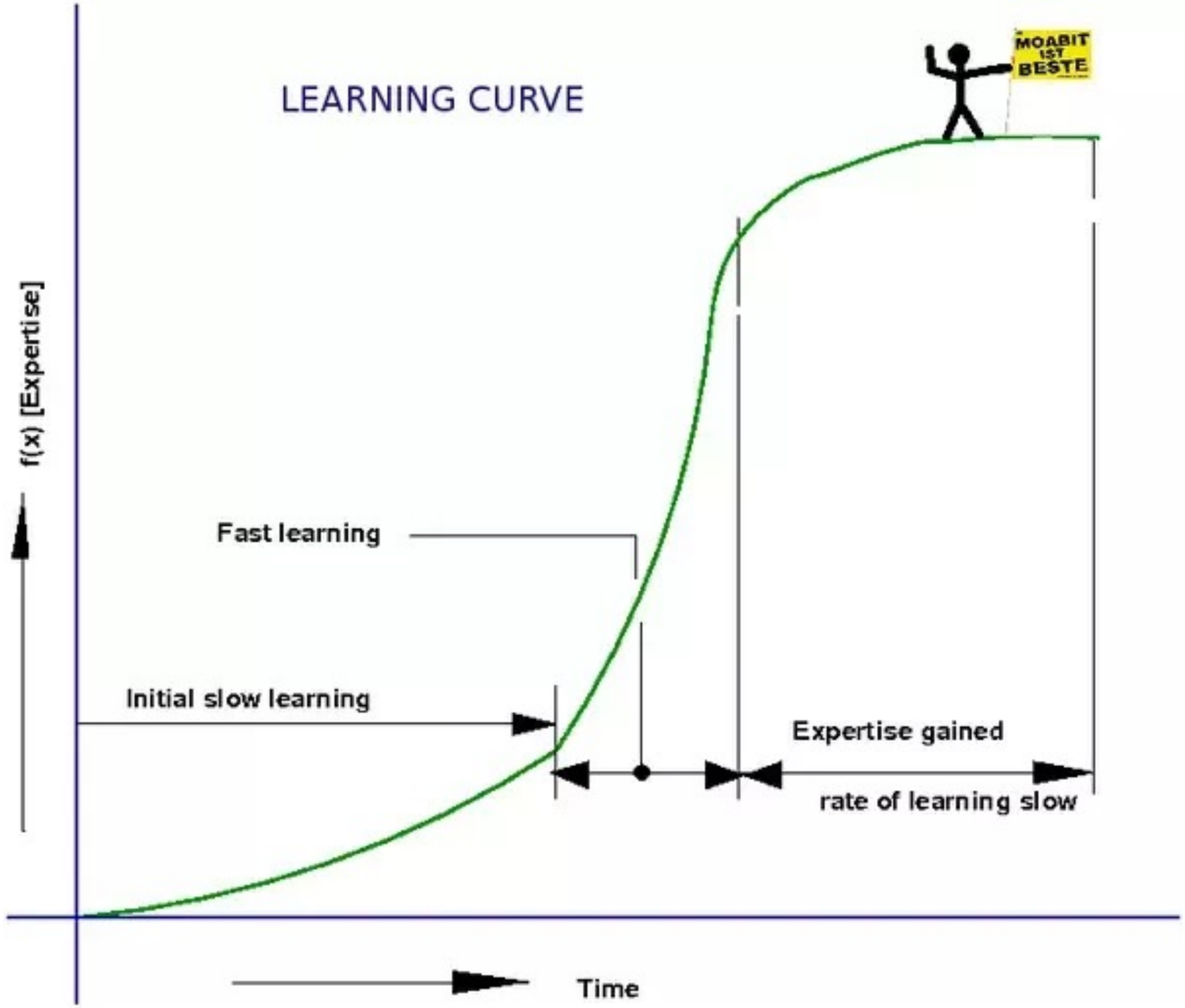
- Originated from S
  - Statistical programming language developed by John Chambers at Bell Labs in the 70s
  - Developed at the same times as Unix
  - Closed source
- First version of R: developed by Robert Gentleman and Ross Ihaka in the mid-1990s
  - Aimed for better statistics software in their Mac teaching labs
  - Open Source alternative
  - R 1.0.0 released in 2000
- Current version 4.2.2
- Developers: international R-core developing team



Ross Ihaka

Why learn R?

- Language and environment for statistical computing and graphics
- Open Source (free)
- Cross-platform compatibility
- Community supported
- Great flexibility to do what you want
- Many packages available: ecology, metabarcoding, networks
- Amazing publication quality graphs



Installing R

https://cran.uib.no

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux (Debian, Fedora/Redhat, Ubuntu)
- Download R for macOS
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2022-06-23, Funny-Looking Kid) R-4.2.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

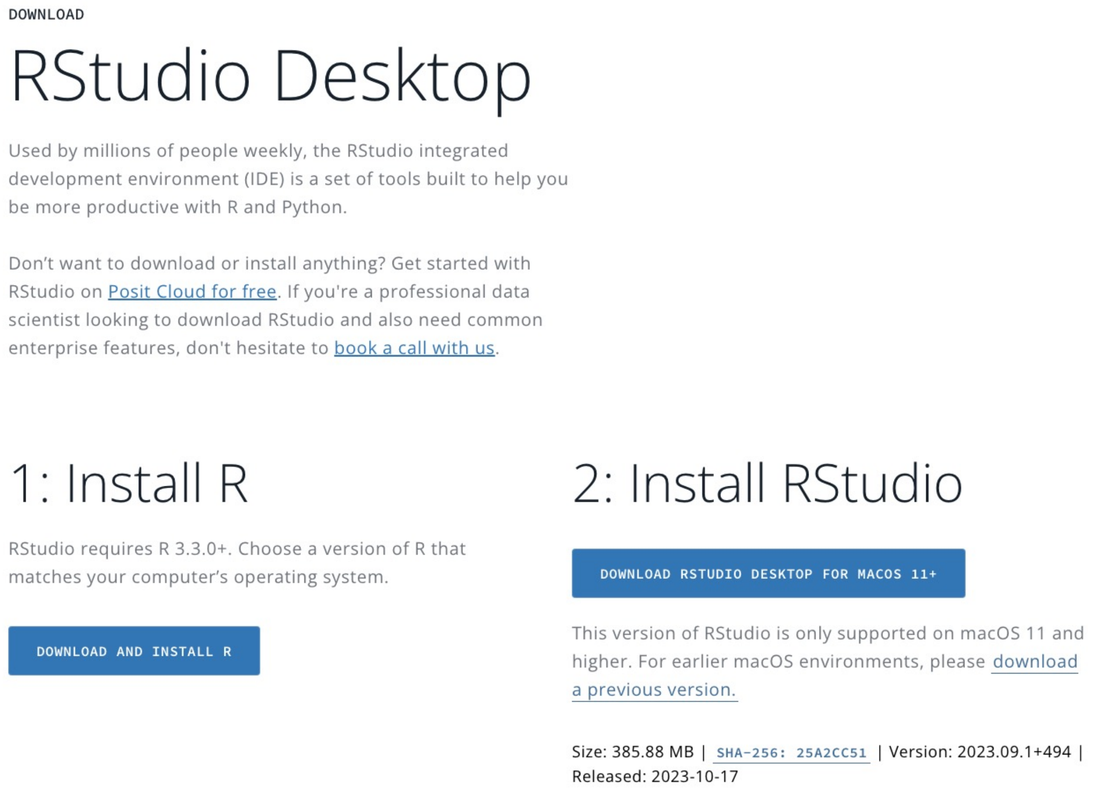
Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

Installing R-Studio

- An Integrated Development Environment (IDE): ***R-Studio***
  - Set of tools designed to help and be more productive with R
  - Includes a console and syntax-highlighting editor that supports code execution
  - Can have several open sessions (aka. Projects)
  - Includes a Unix terminal
  - Can run other programming languages (python, bash)



https://posit.co/download/rstudio-desktop/

RStudio

Editor

```
## BIO9905MERG1_V21

## Community ecology exercises

# Install packages

install.packages("vegan")     # Community ecology functions

# Read dada2 otuput

otu.tab<-read.table("https://github.com/path/")


#Library Vegan
```

Unix terminal

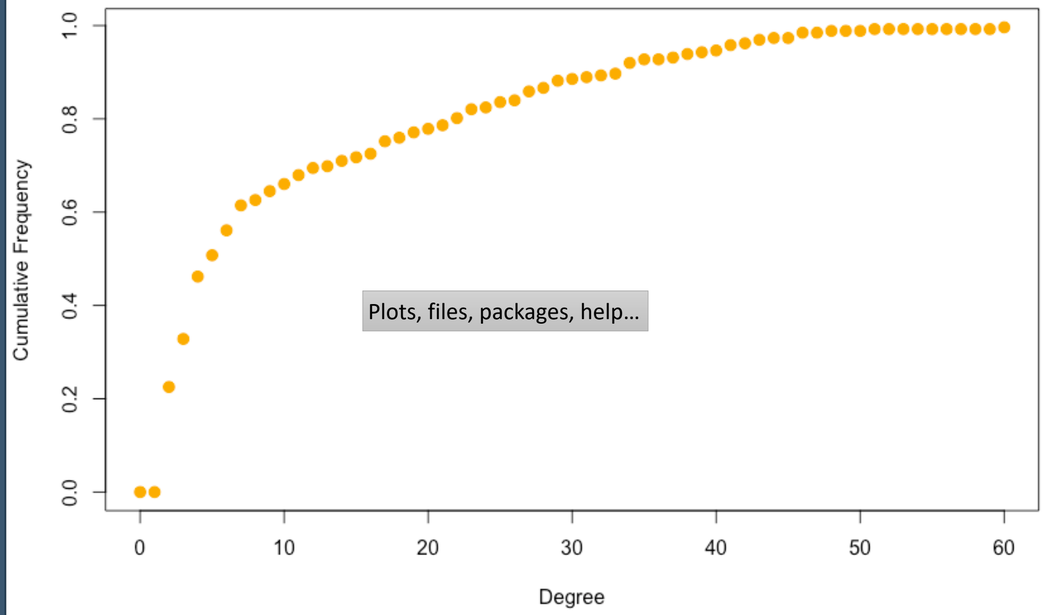
Console

```
bp_OTU_00594 . . . . . .
bp_OTU_000127 . . . . . .
bp_OTU_000003 . . . . . .
ep_OTU_00353 . . . . . .
bn_OTU_000265 . . . . . .
en_OTU_00329 . . . . . .
en_OTU_01088 . . . . . .
> bbmo.core.nw.deg.dist <- degree_distribution(bbmo.core.nw, cumulative=T, mode="all")
> plot( x=0:max(bbmo.core.nw.degree), y=1-bbmo.core.nw.deg.dist , pch=19, cex=1.2, col="orange",
+
+       xlab="Degree", ylab="Cumulative Frequency")
> plot(bbmo.core.nw)
> plot( x=0:max(bbmo.core.nw.degree), y=1-bbmo.core.nw.deg.dist , pch=19, cex=1.2, col="orange",
+
+       xlab="Degree", ylab="Cumulative Frequency")
>
```

Variables, etc

tara_mine_otu_otu_18S_MIC_mi…	50853 obs. of 8 variables
temp.daylength.merged	371 obs. of 11 variables
temp57	8 obs. of 3 variables
varpart.abiotic.biotic	List of 6
winter.vertex.mat	int [1:156, 1] 1 2 3 4 5 6 7 8 9 10 ...
world	253 obs. of 52 variables
Values	
apl	1.34615384615385
autumn.vertex	'igraph.vs' Named int [1:26] 1 2 3 4 5 6 7 8 9 10 ...
autumn.vertex.tab	'igraph.vs' Named int [1:26] 1 2 3 4 5 6 7 8 9 10 ...
avg.degree	2.11764705882353
bbmo.core.niche.val.signific…	chr [1:371] "bn_ASV_000001" "bp_ASV_000001" "ep_ASV_00001" "ep_ASV_00002" "en_ASV…
bbmo.core.nw.deg.dist	num [1:61] 1 1 0.775 0.672 0.538 ...

Plots, files, packages, help…



Presentation format

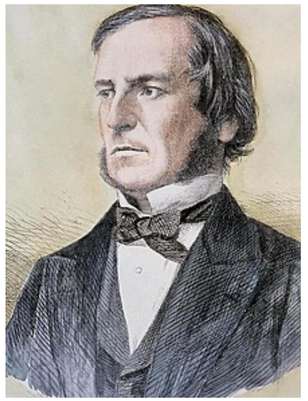
- The following slides shows some typical R commands and the important data structures
- The idea is that you become familiar with reading code as you will do when working with R-Studio
- After this introduction, you should be able to follow other sections of the course using R

Intro to R

- Basic operations.

```
6+6
6-6
6*6
6/6
log(6)
log2(6)
log10(6)
log(6,10)
log(6,3)
6^6
sin(pi/2)
cos(pi/2)
# …ETC
```

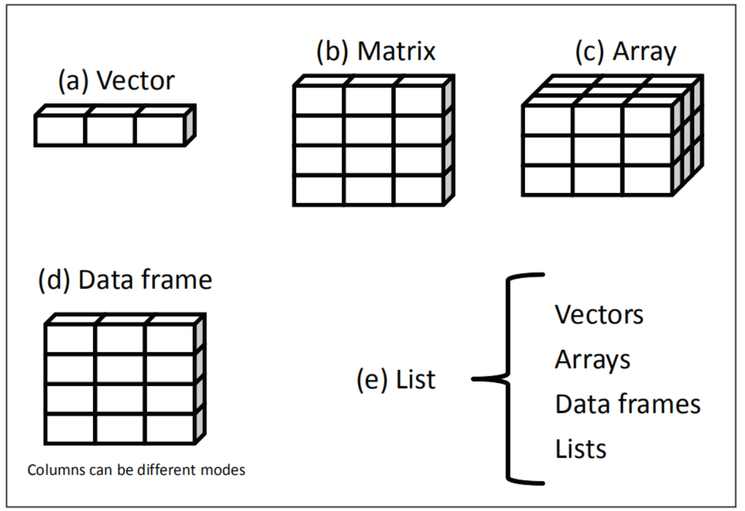
```r
#Let's have a look to basic datatypes on which R objects are built

#Numeric: numbers with decimals
mynumber <- 66.6
print(mynumber)
# [1] 66.6
class(mynumber) # use it to know what is the data type
# [1] "numeric"

#Integer: numbers with no decimals
mynumber.int <- as.integer(mynumber)
# [1] 66
class(mynumber.int)
# [1] "integer"

#Character: can be a letter or a combination of letters enclosed by quotes
mychar <- "bioinfo course"
print(mychar)
# [1] "bioinfo course"
class(mychar)
#[1] "character"

#Logical: a variable that can be TRUE or FALSE (boolean)
im.true <- TRUE
print(im.true)
#[1] TRUE
class(im.true)
# [1] "logical"
```



George Boole

Objects and data-types

- Fundamental structures in R
- Objects: Vectors, Lists, Matrices, Arrays, Factors, Data frames
- Data types: numeric, integer, character, logical



Vectors

Objects that are used to store values or other information of the same data type
They are created with the function "c()" that will generate a 1D array

```
species <- c(123,434,655,877,986) # we create a numeric vector
class(species)
#[1] "numeric"
length(species) # number of elements in the vector
#[1] 5
species[5] # accessing the fifth element in the vector
#[1] 986
species[1:3]
#[1] 123 434 655
species.names <- c("dog","lion","human","pig","cow") # we create a character vector
class(species.names)
# [1] "character"
```

```r
species <- c(123,434,655,877,986) # we create a numeric vector
class(species)
#[1] "numeric"
length(species) # number of elements in the vector
#[1] 5
species[5] # accessing the fifth element in the vector
#[1] 986
species[1:3]
#[1] 123 434 655
species.names <- c("dog","lion","human","pig","cow") # we create a character vector
class(species.names)
# [1] "character"
```

Factors

```
#Factor: used to refer to a qualitative relationship.

# to generate a factor, we'll use a vector defined with the function c()
myfactor <- factor(c("good", "bad", "ugly", "good", "good", "bad", "ugly"))
print(myfactor)
#[1] good bad  ugly good good bad  ugly
#Levels: bad good ugly  <- NB: levels of the factor
class(myfactor)
#[1] "factor"
levels(myfactor) # this can be used to check the levels of a factor
# [1] "bad"  "good" "ugly"
nlevels(myfactor)
# [1] 3
class(levels(myfactor))
# [1] "character"
```

List

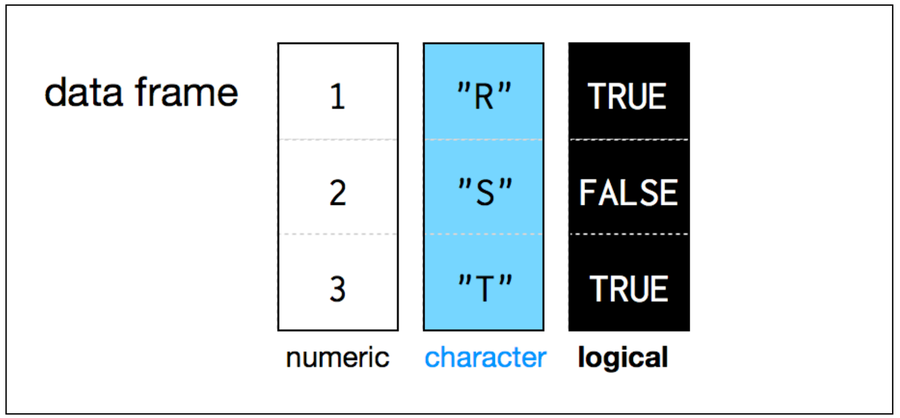
```
# List
# It can contain elements of various data types (e.g.vectors,functions,matrices,another list)
# Example of vectors with three different data types in one list
list1 <- c(1:5) # integer vector
#[1] 1 2 3 4 5
list2 <- factor(1:5) # factor vector
# [1] 1 2 3 4 5
# Levels: 1 2 3 4 5
list3 <- letters[1:5]
# [1] "a" "b" "c" "d" "e"
grouped.lists <- list(list1,list2,list3)
#[[1]]
#[1] 1 2 3 4 5

#[[2]]
#[1] 1 2 3 4 5
#Levels: 1 2 3 4 5

#[[3]]
#[1] "a" "b" "c" "d" "e"

#Accessing elements of a list
grouped.lists[[1]] # accessing the first vector
# [1] 1 2 3 4 5
```

Matrix

```
#Matrix
#Like a vector, a matrix stores information of the same data type, but different from a vector, it has 2 dimensions.

#syntax:  mymatrix <- matrix(vector, nrow=r, ncol=c, byrow=FALSE, dimnames=list(vector_rownames, vector_colnames))

# byrow=F indicates that the matrix should be filled by columns

mymatrix <- matrix(seq(1:100), nrow=10, ncol=10, byrow=FALSE, dimnames=list(c(1:10), letters[1:10]))
print(mymatrix)
#      a  b  c  d  e  f  g  h  i   j
# 1    1 11 21 31 41 51 61 71 81  91
# 2    2 12 22 32 42 52 62 72 82  92
# 3    3 13 23 33 43 53 63 73 83  93
# 4    4 14 24 34 44 54 64 74 84  94
# 5    5 15 25 35 45 55 65 75 85  95
# 6    6 16 26 36 46 56 66 76 86  96
# 7    7 17 27 37 47 57 67 77 87  97
# 8    8 18 28 38 48 58 68 78 88  98
# 9    9 19 29 39 49 59 69 79 89  99
# 10 10 20 30 40 50 60 70 80 90 100
```

Dataframes

```
#Dataframes
# More general than a matrix and can contain different data types
# Variables or features are in columns, while observations are in rows
# =>NB: this is one of the most common objects in metabarcoding analyses<=
# Generated with the data.frame() function

my.data.frame<-data.frame(
Name=c("Game of Thrones","MrRobot","WestWorld"),
Budget=c(344,59,122),
Seasons=c(8,4,3),
Audience=c(300,14,80),
Actors=c(221,56, 90)

print(my.data.frame)
#              Name Budget Seasons Audience Actors
#1 Game of Thrones    344       8      300    221
#2          MrRobot     59       4       14     56
#3       WestWorld    122       3       80     90

row.names(my.data.frame) <- my.data.frame[,1] # Assign to the row names the names in the first
column
my.data.frame <- my.data.frame[,-1] # Remove the fisrt column
print(my.data.frame)
#                 Budget Seasons Audience Actors
# Game of Thrones    344       8      300    221
# MrRobot             59       4       14     56
# WestWorld          122       3       80     90
```



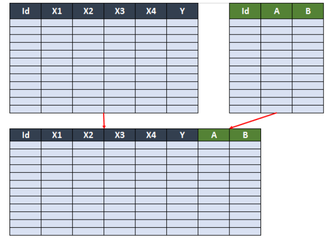
Dataframes

```
class(my.data.frame)
# [1] "data.frame"
ncol(my.data.frame) # Number of columns
# [1] 4
nrow(my.data.frame) # Number of rows
# [1] 3
colnames(my.data.frame) # Column names
# [1]  "Budget"   "Seasons"  "Audience" "Actors"
rownames(my.data.frame) # Name of rows
# "Game of Thrones"   "MrRobot"     "WestWorld"
colSums(my.data.frame) # Sum values in columns
# Budget  Seasons Audience   Actors
#   525       15      394      367
rowSums(my.data.frame) # We sum the values, even if they make no sense in the example
# Game of Thrones         MrRobot       WestWorld
 #             873             133             295
```

Dataframes

```
rbind(my.data.frame,my.data.frame) #  appends dataframes one below the other (column names identical)
#                   Budget Seasons Audience Actors
# Game of Thrones      344       8      300    221
# MrRobot               59       4       14     56
# WestWorld            122       3       80     90
# Game of Thrones1     344       8      300    221
# MrRobot1              59       4       14     56
# WestWorld1           122       3       80     90
```

```
cbind(my.data.frame,my.data.frame) # appends dataframes one next to the other (row names identical)
#                   Budget Seasons Audience Actors Budget Seasons Audience Actors
# Game of Thrones    344       8      300    221    344       8      300    221
# MrRobot             59       4       14     56     59       4       14     56
# WestWorld          122       3       80     90    122       3       80     90
```



```
head(my.data.frame, 2) # Useful to have a look to the beginning of the dataframe (specially useful in big tables)
# Here asking to print only 2 rows
#                 Budget Seasons Audience Actors
# Game of Thrones    344       8      300    221
# MrRobot             59       4       14     56
```

```
my.data.frame[1:2,2:4] # Useful to look at specific sections of the dataframe
#                 Seasons Audience Actors
# Game of Thrones       8      300    221
# MrRobot               4       14     56
```

Dataframes

```
# Let's generate a dataframe with different data types

my.data.frame.2<-data.frame(
Name=c("Game of Thrones","MrRobot","WestWorld", "Chernobyl"),
Rating=c("Excellent","Very Good","Excellent", "Very Good"),
Audience.Restriction=c(TRUE,FALSE,TRUE, FALSE)

print(my.data.frame.2)
#              Name    Rating  Audience.Restriction
# 1 Game of Thrones  Excellent                  TRUE
# 2         MrRobot  Very Good                 FALSE
# 3       WestWorld  Excellent                  TRUE
# 4       Chernobyl  Very Good                 FALSE
#Rename row names
row.names(my.data.frame.2) <- my.data.frame.2[,1]
my.data.frame.2<-my.data.frame.2[,-1] # Remove redundant column 1

#                   Rating Audience.Restriction
# Game of Thrones Excellent                  TRUE
# MrRobot         Very Good                 FALSE
# WestWorld       Excellent                  TRUE
# Chernobyl       Very Good                 FALSE

str(my.data.frame.2) # Let's look at the data types within this dataframe

# 'data.frame':4 obs. of  2 variables:
#  $ Rating              : chr  "Excellent" "Very Good" "Excellent" "Very Good"
#  $ Audience.Restriction: logi  TRUE FALSE TRUE FALSE

# Variables in this case are characters and logical (TRUE/FALSE)
```

Dataframes

```
#Merge two dataframes based on a pattern
# We will use the series names to merge these dataframes as this is what they have in common

data.frame.large<-merge(my.data.frame, my.data.frame.2, by="row.names") # "by" indicates the column used for merging

#         Row.names Budget Seasons Audience Actors    Rating Audience.Restriction
# 1 Game of Thrones    344       8      300    221 Excellent                 TRUE
# 2         MrRobot     59       4       14     56 Very Good                FALSE
# 3       WestWorld    122       3       80     90 Excellent                 TRUE
```

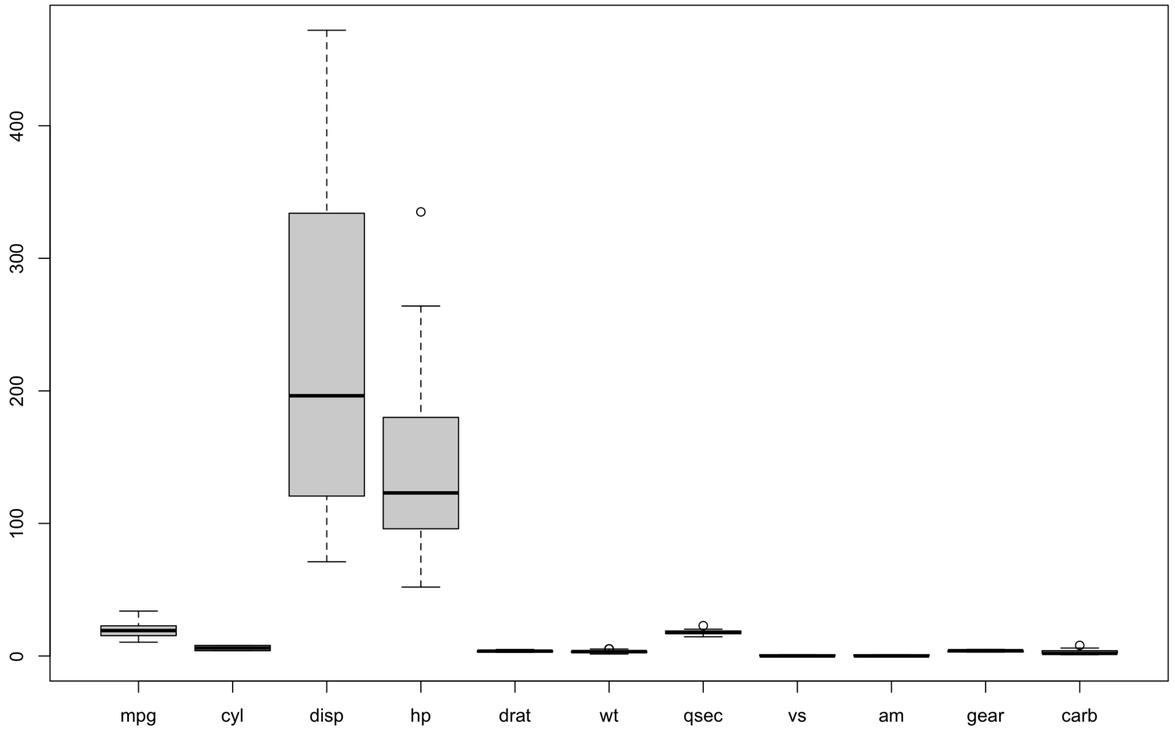
NB: “Chernobyl” was not used, as it was only present in one data frame, but this could be modified

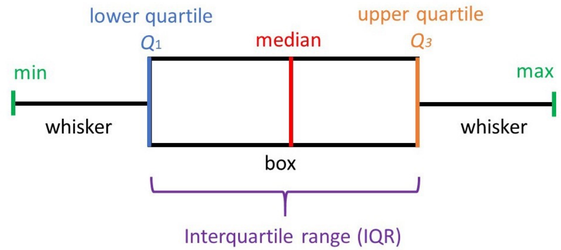
Working with tables or data frames

```
#Useful commands to work with tables or dataframes
getwd()             # get working directory
# [1] "/Users/admin"
setwd("path/to/my/directory") # set working directory

my.table<-read.table(file="table.tsv", sep="\t", header=T) # read table; several other options available
dim(my.table)       # Table dimensions
nrow(my.table)      # Number of rows
ncol(my.table)      # Number of columns
colnames(my.table) # Name of columns
rownames(my.table) # Name of rows
colSums(my.table)  # Sum of numeric values in columns
rowSums(my.table)  # Sum of numeric values in rows
head(my.table)      # See table header
t(my.table)         # Transpose table

#Table subsetting
# Format:  my.table[row, column]
my.table[1,2]                              # Get value from row 1, column 2
my.table[1,]                               # Get values from row 1 across all columns
my.table$column.name<-NULL                 # Remove column
my.table[-5,-2]                            # Remove row 5 and column 2
my.table[-(5:10),]                         # Remove rows 5 to 10, keep all columns
my.table[,-(which(colSums(my.table)==0))] # Remove columns that sum 0
```

Simple plots

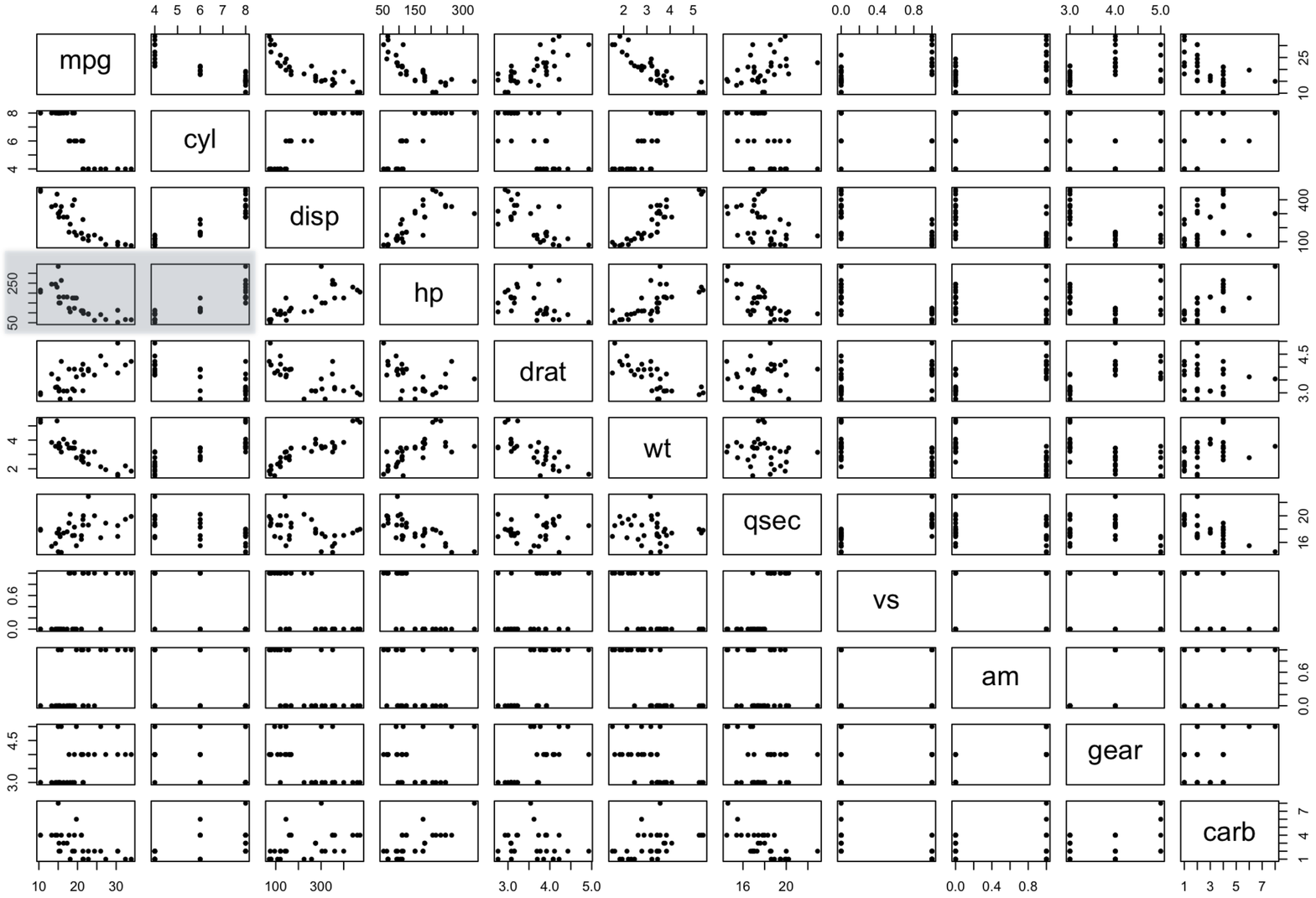
```
#Plotting
data("mtcars") # We load a dataset that comes with R
#The data was extracted from the 1974 Motor Trend US magazine, and comprises fuel consumption and 10 aspects
# of automobile design and performance for 32 automobiles (1973 & 74 models).

#Data structure
#                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
# Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
# Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
# Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
# Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
# ...

# [, 1] mpg  Miles/(US) gallon
# [, 2] cyl  Number of cylinders
# [, 3] disp Displacement (cu.in.)
# [, 4] hp Gross horsepower
# [, 5] drat Rear axle ratio
# [, 6] wt Weight (1000 lbs)
# [, 7] qsec 1/4 mile time
# [, 8] vs Engine (0 = V-shaped, 1 = straight)
# [, 9] am Transmission (0 = automatic, 1 = manual)
# [,10] gear Number of forward gears
# [,11] carb Number of carburetors
```





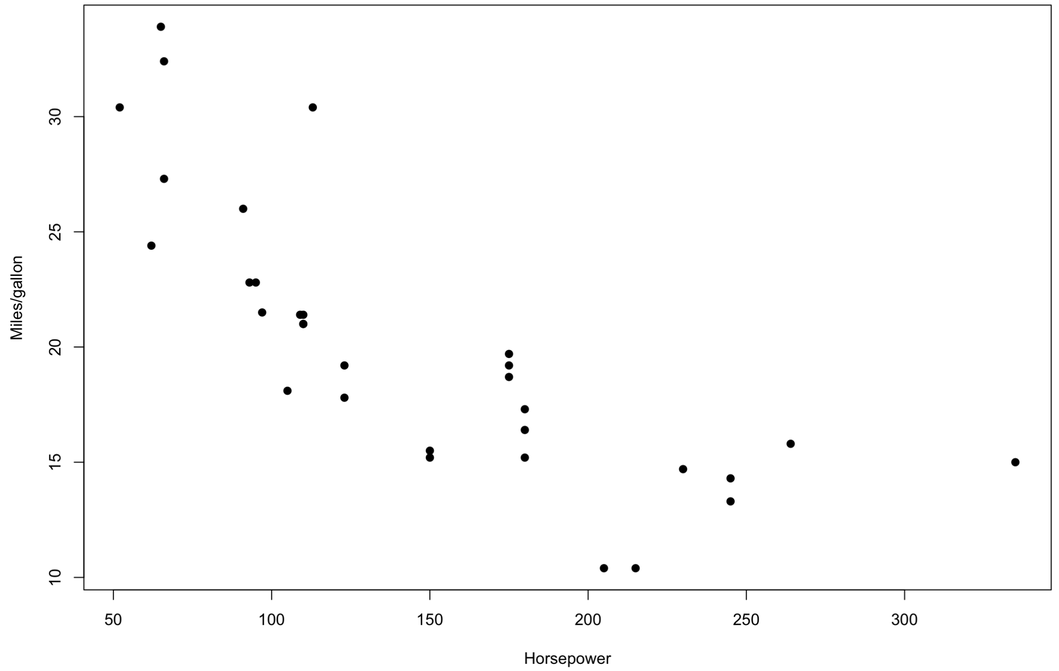

```
boxplot(mtcars) # make a boxplot of variables across car models
```

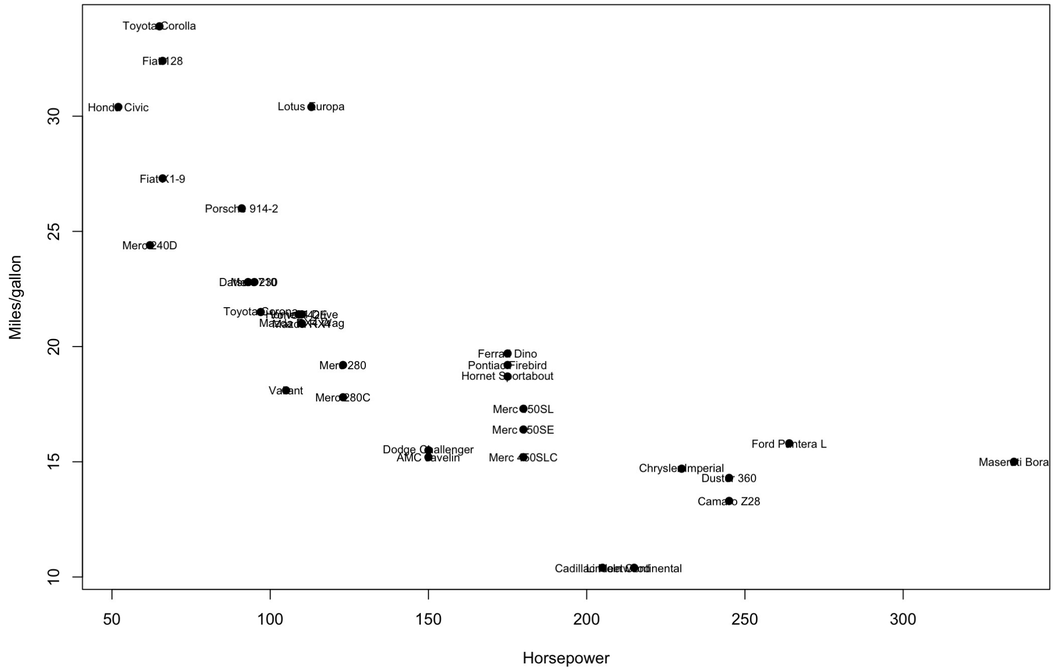
```
14 # [, 1]   mpg Miles/(US) gallon
15 # [, 2]   cyl Number of cylinders
16 # [, 3]   disp   Displacement (cu.in.)
17 # [, 4]   hp Gross horsepower
18 # [, 5]   drat   Rear axle ratio
19 # [, 6]   wt Weight (1000 lbs)
20 # [, 7]   qsec   1/4 mile time
21 # [, 8]   vs Engine (0 = V-shaped, 1 = straight)
22 # [, 9]   am Transmission (0 = automatic, 1 = manual)
23 # [,10]   gear   Number of forward gears
24 # [,11]   carb   Number of carburetors
```



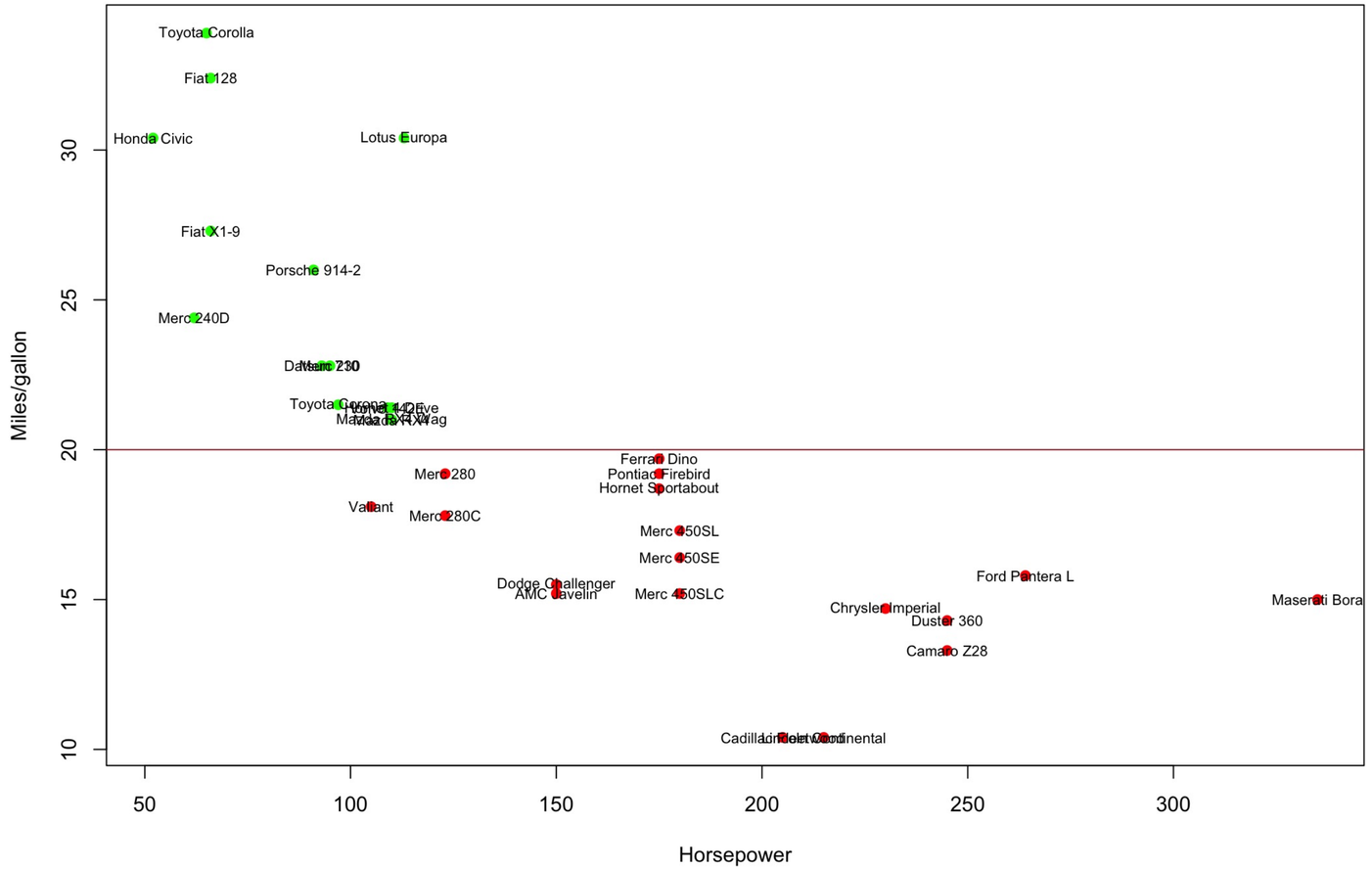

```
plot(mtcars, pch=19, cex=0.6) # make x-y plots for all variables
```

```
14 # [, 1]   mpg Miles/(US) gallon
15 # [, 2]   cyl Number of cylinders
16 # [, 3]   disp   Displacement (cu.in.)
17 # [, 4]   hp Gross horsepower
18 # [, 5]   drat   Rear axle ratio
19 # [, 6]   wt Weight (1000 lbs)
20 # [, 7]   qsec   1/4 mile time
21 # [, 8]   vs Engine (0 = V-shaped, 1 = straight)
22 # [, 9]   am Transmission (0 = automatic, 1 = manual)
23 # [,10]   gear   Number of forward gears
24 # [,11]   carb   Number of carburetors
```

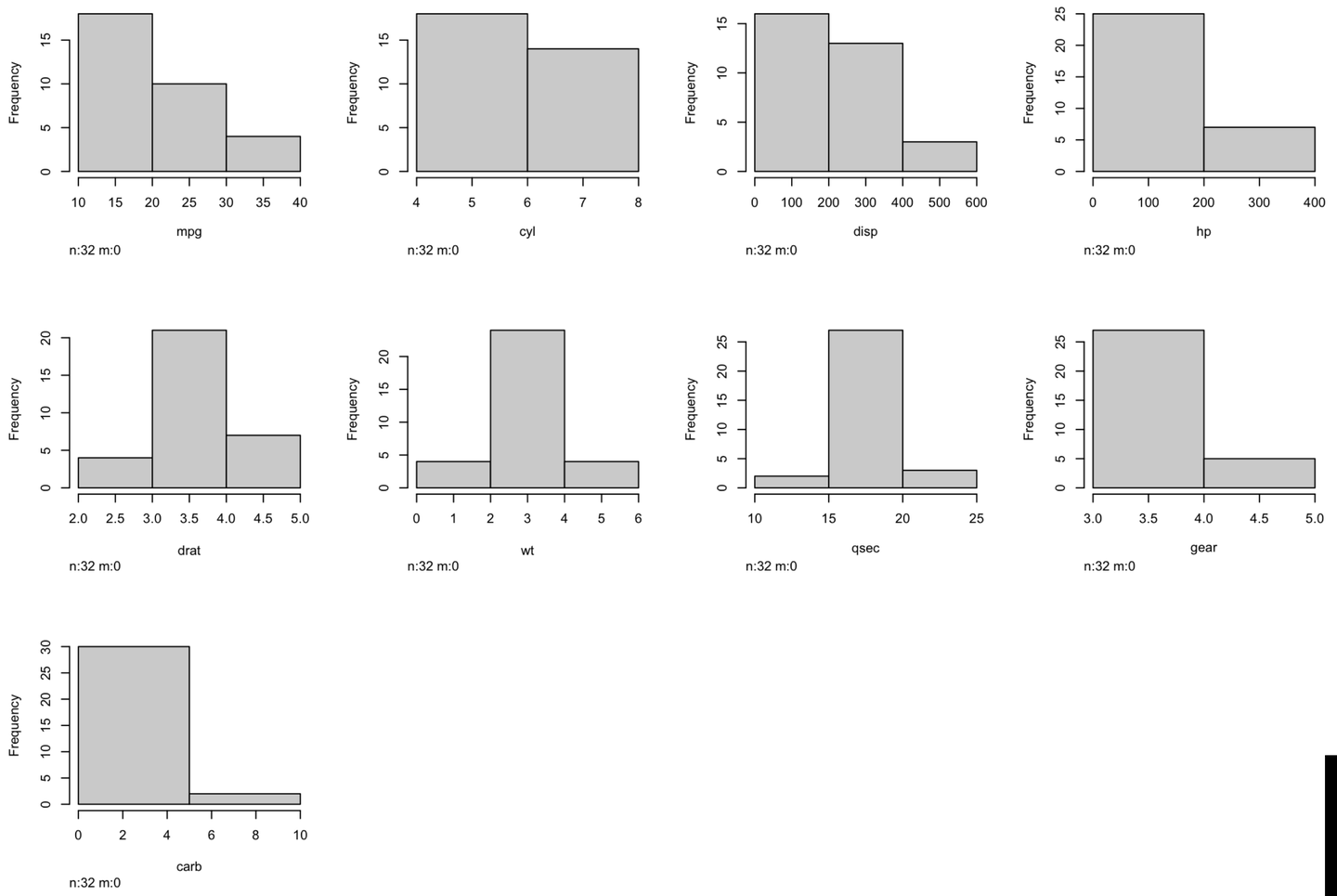





```
plot(mtcars$hp, mtcars$mpg, xlab="Horsepower", ylab="Miles/gallon", pch=19) # we plot horsepower vs. miles per gallon
text(mtcars$hp, mtcars$mpg, row.names(mtcars), cex=0.7) # we add the car model
```

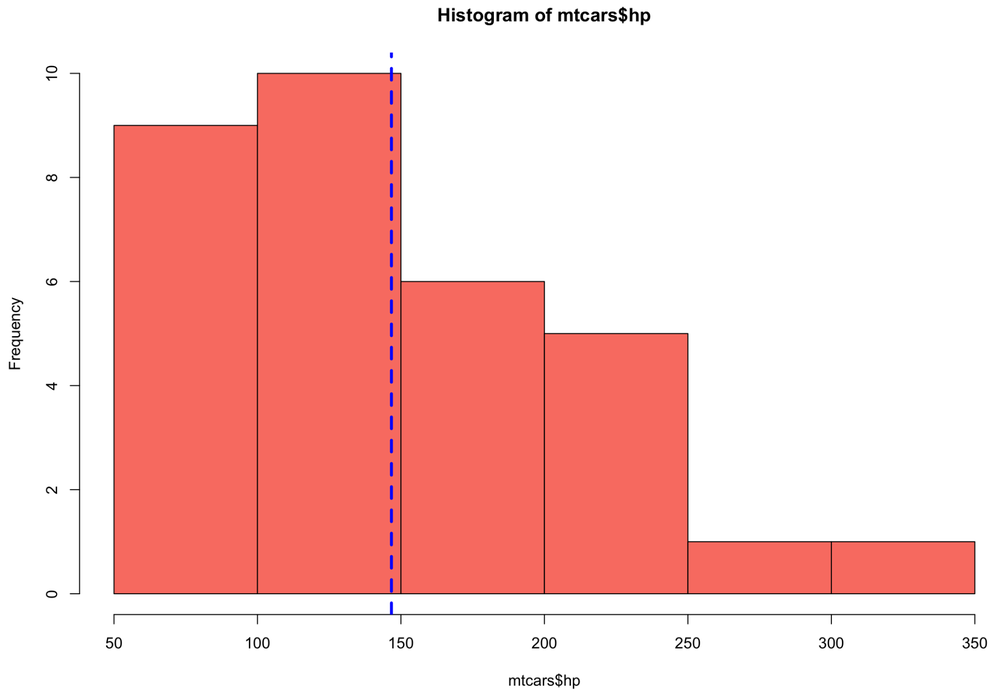


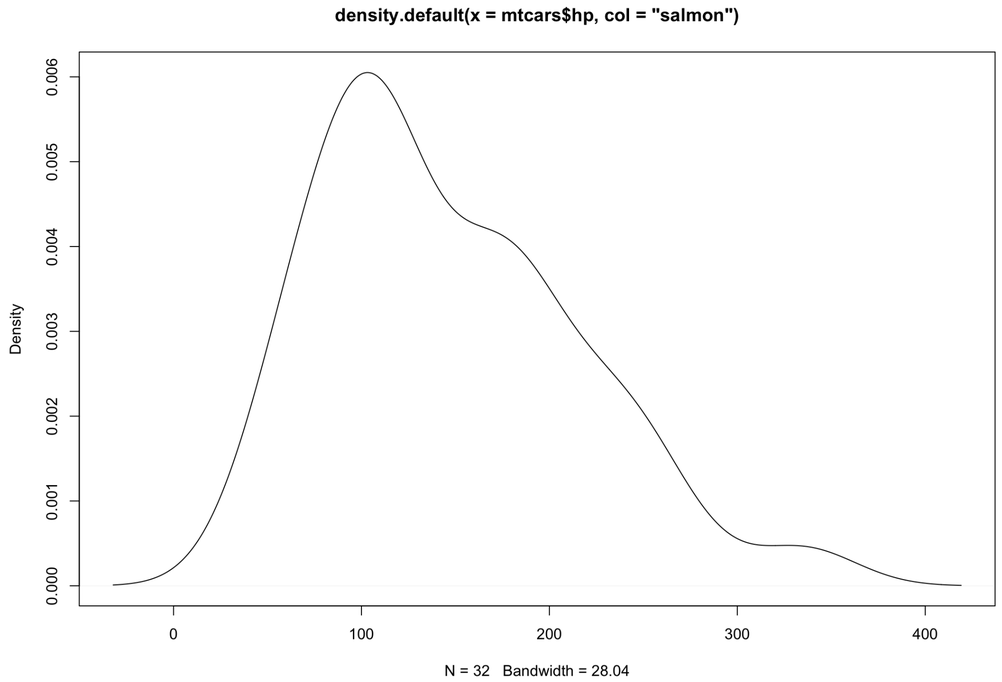

```
plot(mtcars$hp, mtcars$mpg, xlab="Horsepower", ylab="Miles/gallon", pch=19, col=ifelse(mtcars$mpg<20,"red", "green"))
# We color dots according to
      a condition (20<x<20 mpg)
text(mtcars$hp, mtcars$mpg, row.names(mtcars), cex=0.7) # we add the car model
abline(h=20, col="brown") # we add an horizontal line at "20"
```




```
hist(mtcars) # we plot an histogram for the different variables
```

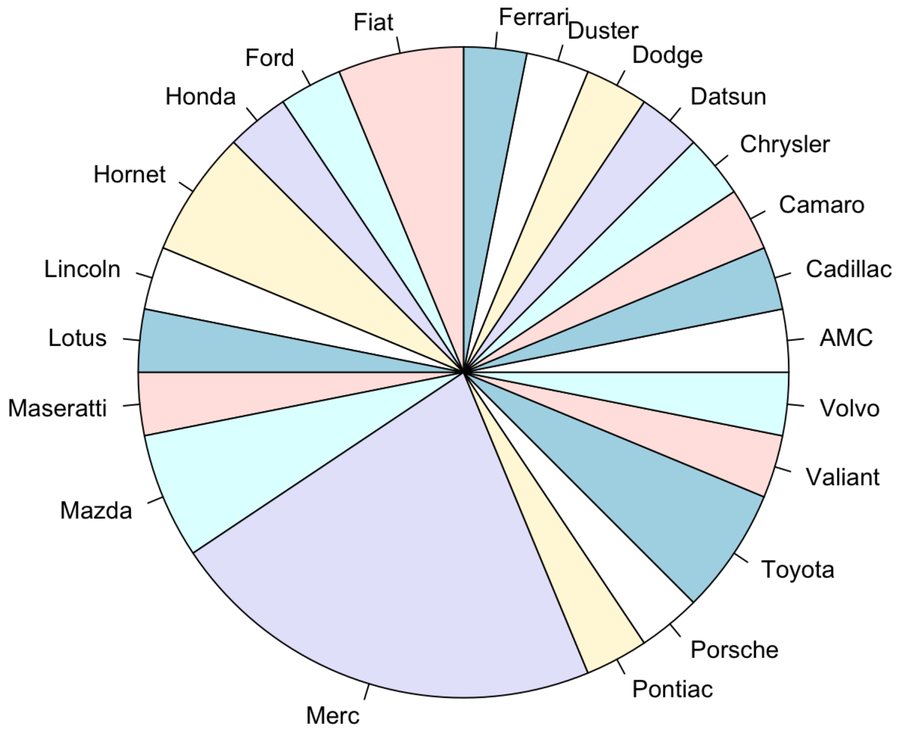
```
14 # [, 1]   mpg Miles/(US) gallon
15 # [, 2]   cyl Number of cylinders
16 # [, 3]   disp   Displacement (cu.in.)
17 # [, 4]   hp Gross horsepower
18 # [, 5]   drat   Rear axle ratio
19 # [, 6]   wt Weight (1000 lbs)
20 # [, 7]   qsec   1/4 mile time
21 # [, 8]   vs Engine (0 = V-shaped, 1 = straight)
22 # [, 9]   am Transmission (0 = automatic, 1 = manual)
23 # [,10]   gear   Number of forward gears
24 # [,11]   carb   Number of carburetors
```






```
hist(mtcars$hp, col="salmon")
abline(v=mean(mtcars$hp), col="blue", lwd=3, lty=2)
plot(density(mtcars$hp))
```

```r
brands<-c("Mazda", "Mazda","Datsun", "Hornet", "Hornet", "Valiant", "Duster", "Merc", "Merc", "Merc", "Merc", "Merc","Merc","Merc","Cadillac",
          "Lincoln","Chrysler","Fiat",
          "Honda", "Toyota", "Toyota","Dodge","AMC","Camaro","Pontiac", "Fiat","Porsche", "Lotus", "Ford","Ferrari", "Maseratti","Volvo")
mtcars$brand<-brands # we add an extra column with brands
mtcars[1:5,] # let's double check

#                   mpg cyl disp  hp drat    wt  qsec vs am gear carb  brand
# Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4  Mazda
# Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4  Mazda
# Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1 Datsun
# Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1 Hornet
# Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2 Hornet

pie(table(mtcars$brand)) # we make a piechart of the brands
```



Installing and loading packages

```r
#Installing packages

# R has a large repository of packages for different applications

install.packages("spaa")  # Installs the ecological package spaa
install.packages("vegan") # Installs the community ecology package Vegan with hundreds of functions
library("vegan") # load Vegan
#Loading required package: permute
# Loading required package: lattice
# This is vegan 2.5-7


# Other relevant packages

install.packages("readr")     # To read and write files
install.packages("readxl")    # To read excel files
install.packages("dplyr")     # To manipulate dataframes
install.packages("tibble")    # To work with data frames
install.packages("tidyr")     # To work with data frames
install.packages("stringr")   # To manipulate strings
install.packages("ggplot2")   # To do plots
install.packages("kableExtra")  # necessary for nice table formatting with knitr
install.packages("tidyverse")

if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
#BiocManager::install(version = "3.10")
BiocManager::install(c("dada2", "phyloseq","Biostrings"))

install.packages("devtools")
devtools::install_github("pr2database/pr2database") # Installs directly from github resources that are not in R repos
devtools::install_github("GuillemSalazar/EcolUtils") # Installs other tools for ecological analyses

#Load libraries
#### Load libraries ####

library("dada2")
library("phyloseq")
library("Biostrings")
library("ggplot2")
library("dplyr")
library("tidyr")
library("tibble")
library("readxl")
library("readr")
library("stringr")
library("kableExtra")
library("tidyverse")
#library("pr2database")

```

-The code is available in:

https://github.com/krabberod/UNIS_AB332_2023/