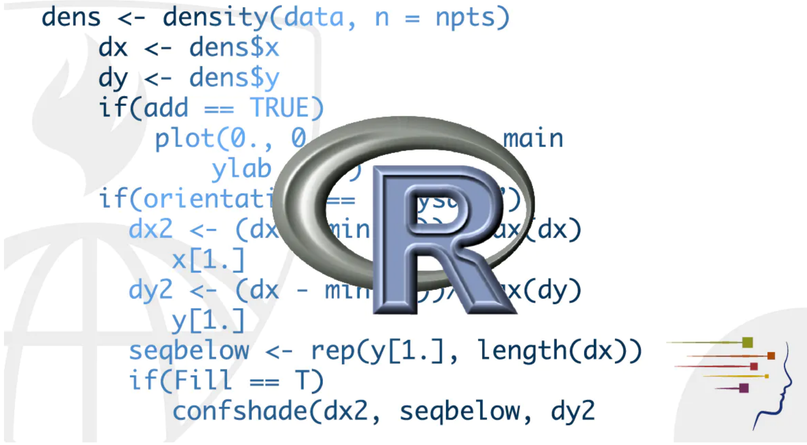


Introduction to R

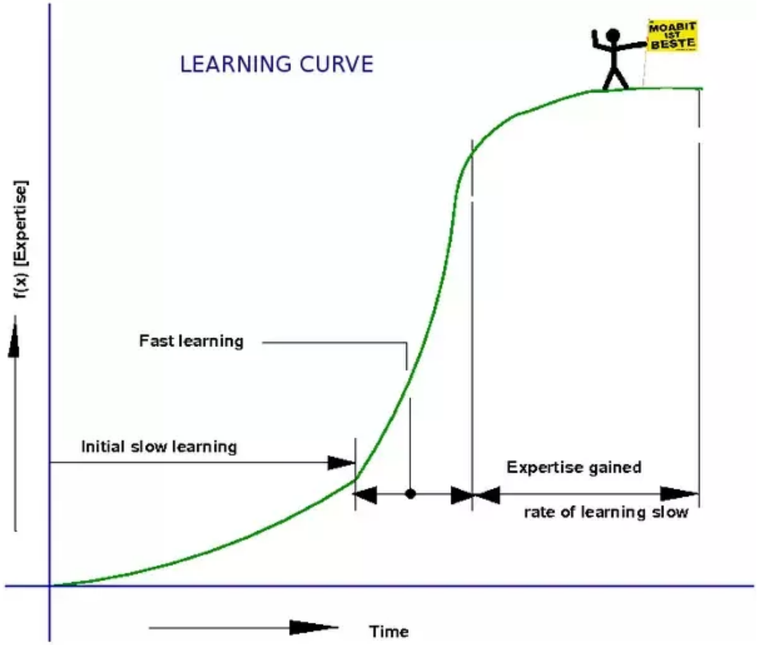
Ramiro Logares (ICM-CSIC, Barcelona)

History

- Originated from S
  - A statistical programming language developed by John Chambers at Bell Labs in the 70s
  - Developed at the same time as Unix
  - Closed source
- The 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 4.4.3
- Developers: international R-core developing team

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
- Publication quality graphs



okay, I want R

- https://cran.uib.no: Linux, Mac, Windows…
- We use 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
  - Allows having several open sessions
  - Includes a Unix terminal

Installing R

R version 4.4.3 (2025-02-28) -- “Trophy Case”

CRAN
Mirrors
What's new?
Search
CRAN Team

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Task Views
Other

Documentation
Manuals
FAQs
Contributed

Donations
Donate

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 (2025-02-28, Trophy Case) R-4.4.3.tar.gz, read what's new in the latest version.
- The CRAN directory src/base-prerelease contains R alpha, beta, and rc releases as daily snapshots in time periods before a planned release.
- Between releases, the same directory src/base-prerelease contains snapshots of current patched and development versions. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Alternatively, daily snapshots are available here.
- 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.

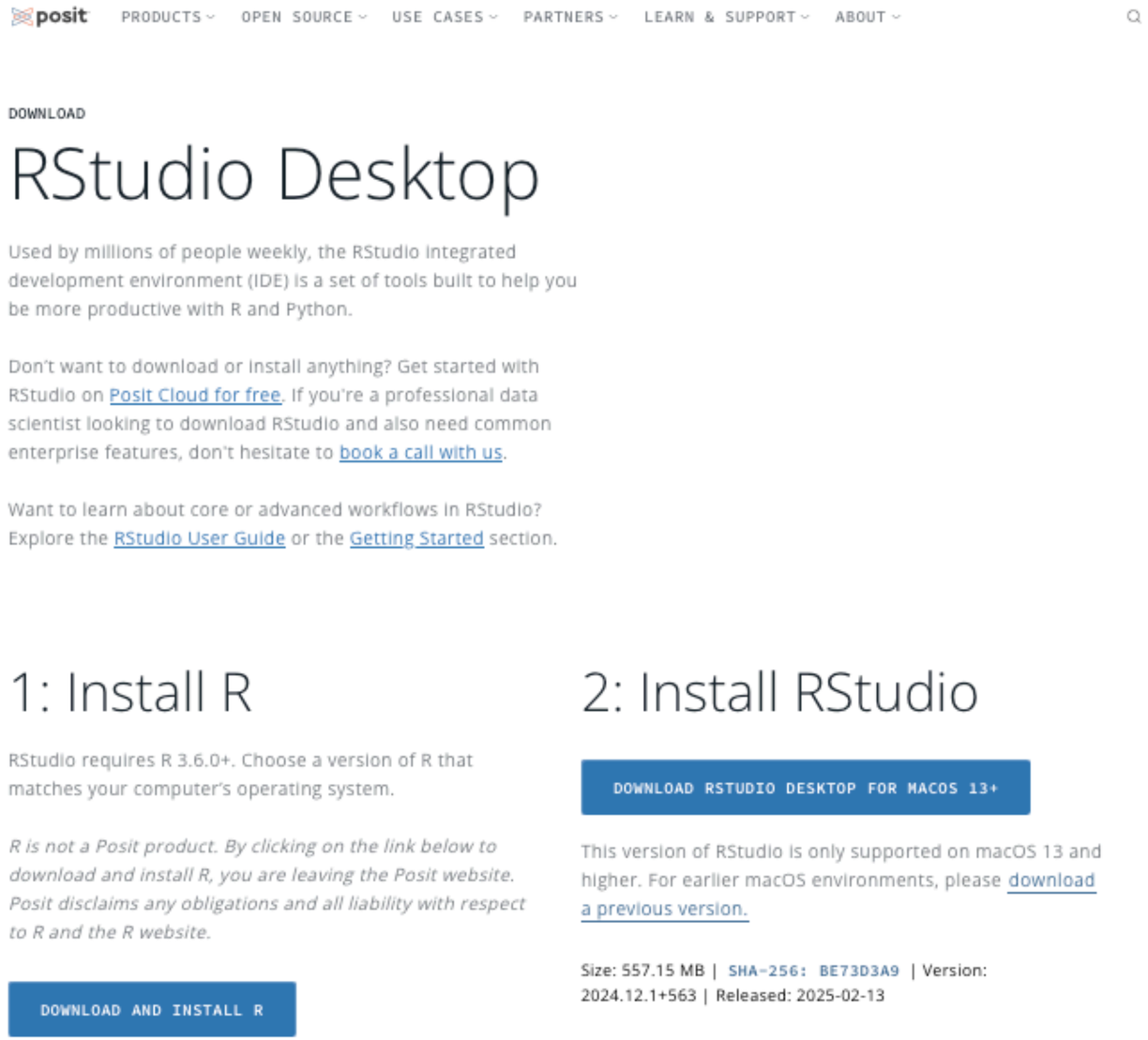
Supporting CRAN

- CRAN operations, most importantly hosting, checking, distributing, and archiving of R add-on packages for various platforms, crucially rely on technical, emotional, and financial support by the R community.

  Please consider making financial contributions to the R Foundation for Statistical Computing.

Installing R-Studio

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

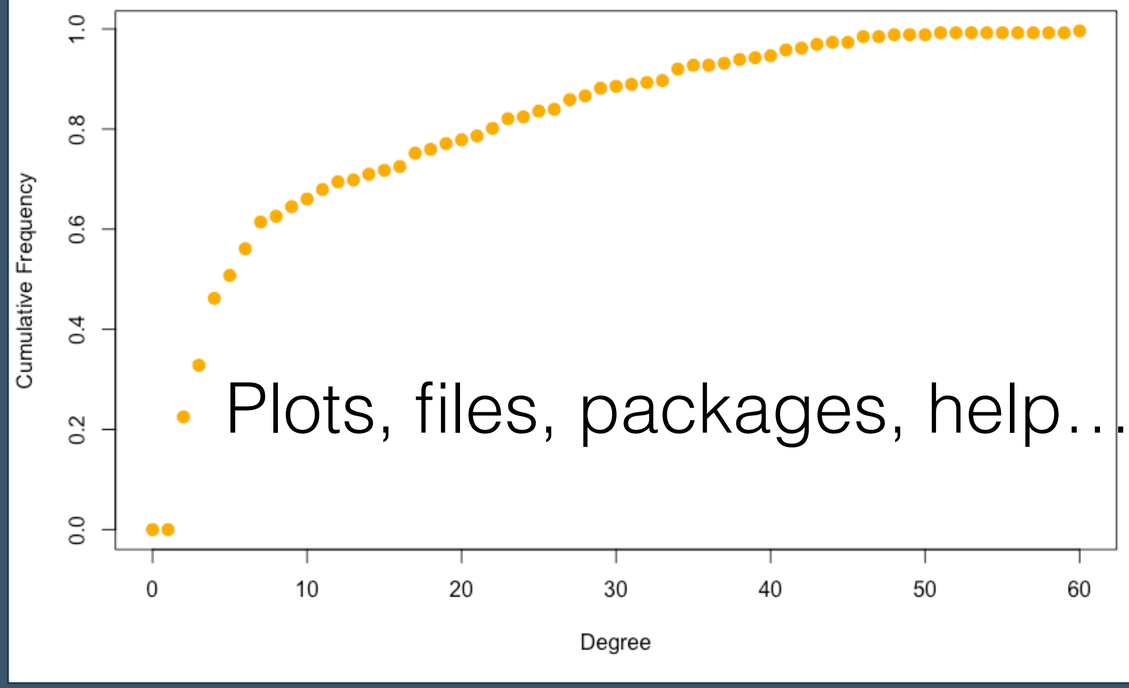
Console

```
en_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

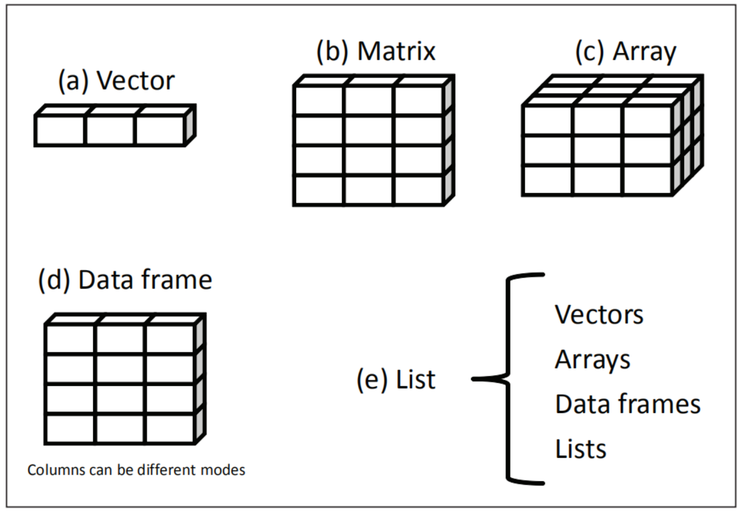
- The following slides come from R-Studio
- The idea is that you become familiar with reading code as you will do when working with R-Studio
- After this introduction, you should be able to follow other sections of the course using R (e.g., DADA2, community ecology)

```
## BIO9905MERG1_V25

# Intro to R

# <- use this symbol for lines that R should not interpret or as comments
    to yourself (highly recommended)

# Help
# Use "?" before a function. E.g. ?sum

# Execute from Editor: select the chunk of code to execute with the mouse
    and press then “control+enter"

# See the output in the console (+plot area)
```

```
#Basic operations


  6+6 # sums two numbers
# Result [1] 12

mysum <- 6+6 # sum two integers and assign it to a variable sum
mysum = 6+6 # same as above


# check the variable content by executing "mysum"
  head(mysum) # useful to see the beginning of a variable if it is too long


#Check variables defined
 ls() # this info is also shown in the "Environment" panel of RStudio


#Remove a variable
 rm(mysum)
 rm(list=ls()) # remove all variables
```

Objects and data-types

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



```
#Let’s have a look at 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"
```

Vectors

```
#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] # accessing the first three elements in the vector
  #[1] 123 434 655

  sum(species) # sum the values in the vector
  #[1] 3075
```

Vectors

```
species.names<-c("dog","lion","human","pig","cow") # we create a character vector

class(species.names)

 # [1] "character"


seq.num<-c(1:100) # we create a sequence of numbers

 # [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25
        26  27  28  29  30  31  32 33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50
        51  52  53  54  55  56  57  58  59  60  61  62  63  64 65  66  67  68  69  70  71  72  73  74  75
        76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96 97  98  99 100


seq.num.by2<-seq(1,100, 2) # same sequence as above but taken by 2

 # [1]  1  3  5  7  9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65 67
        69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99


seq.num.by2[5:10] # we access the 5th to the 10th element

 # [1]  9 11 13 15 17 19
```

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"
```

List

```
#Accessing elements of a list

  grouped.lists[[1]] # accessing the first vector
  # [1] 1 2 3 4 5

  grouped.lists[[3]][5] # accessing the 5th element from the third vector
  # [1] "e"

  #Ungroup the list
   ungrouped.list<-unlist(grouped.lists)
  #  [1] "1" "2" "3" "4" "5" "1" "2" "3" "4" "5" "a" "b" "c" "d" “e"

   class(ungrouped.list)
  # [1] "character" # NB: the list becomes a character datatype

   length(ungrouped.list)
  # [1] 15
```

Matrix

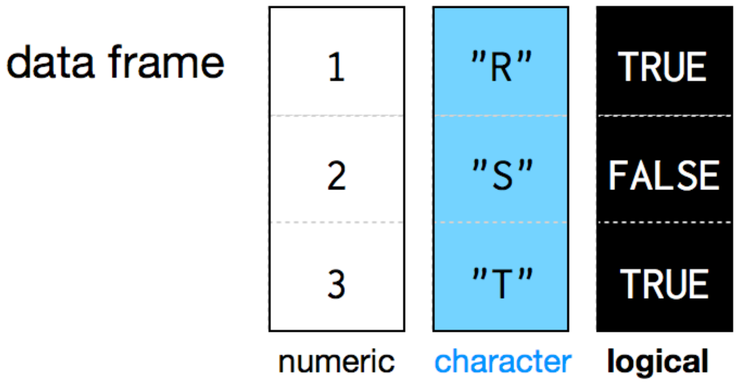
```
#Matrix

#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(char_vector_rownames,
 char_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
```

Matrix

```
mymatrix.rand <- matrix(sample (seq(1:100),100), nrow=10, ncol=10, byrow=FALSE,
  dimnames=list(c(1:10), letters[1:10]))

# We generate a matrix with random numbers

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


  mymatrix[3:6,1:3] # We select what sections of the matrix we want to look at
# Rows 3 to 6 and Columns 1 to 3

#   a  b  c
# 3 20 45 84
# 4 98 77 85
# 5 63 82 29
# 6 51 80 27
```

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)
  )
```



Dataframes

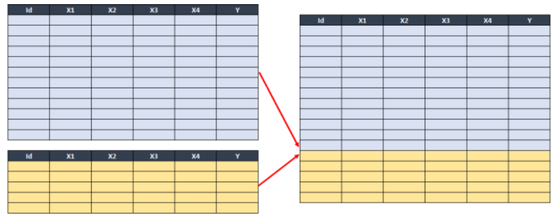
```
print(my.data.frame)
#              Name Budget Seasons Audience Actors
#1 Game of Thrones    344       8      300    221
#2          MrRobot     59       4       14     56
#3       WestWorld    122       3       80     90


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 first column

print(my.data.frame) # By clicking this object in the "Environment" panel
on the right, you'll see a window with the dataframe

#                  Budget Seasons Audience Actors
# Game of Thrones    344       8      300    221
# MrRobot             59       4       14     56
# WestWorld          122       3       80     90
```

Dataframes

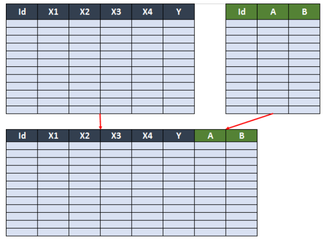
```
class(my.data.frame)
# [1] “data.frame"
ncol(my.data.frame) # Number of columns
# [1] 4
nrow(my.data.frame) # Number of rows
# [1] 3

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
```

Dataframes

```
 head(my.data.frame, 2) # Useful to have a look to the beginning of the
 dataframe (especially 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
```

“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 are 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
```

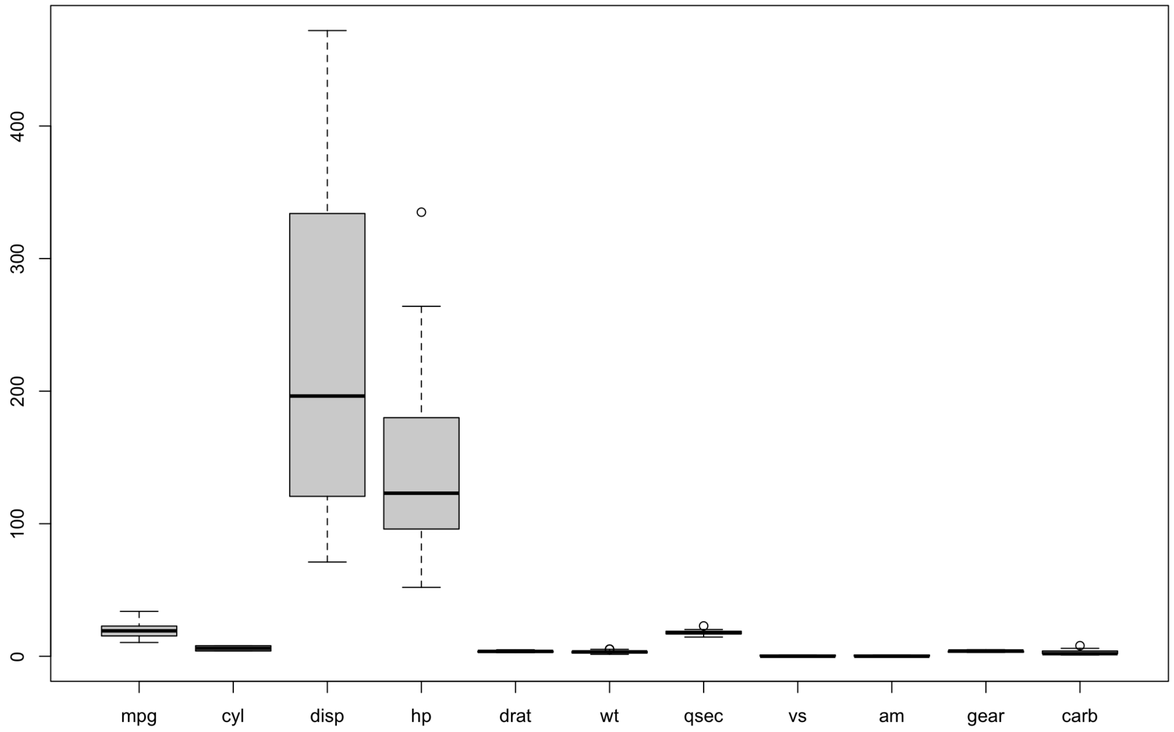
Working with tables or data frames

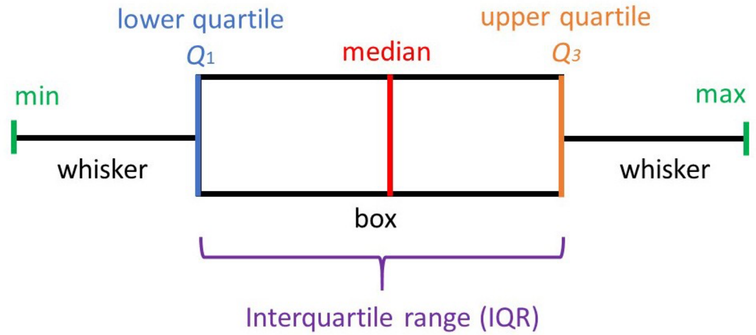
```
# Table subsetting

# 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
```

Plotting

```
#Plotting
data("mtcars") # We load a dataset that comes with R
    #The data was extracted from the 1974 Motor Trend US magazine

#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
```

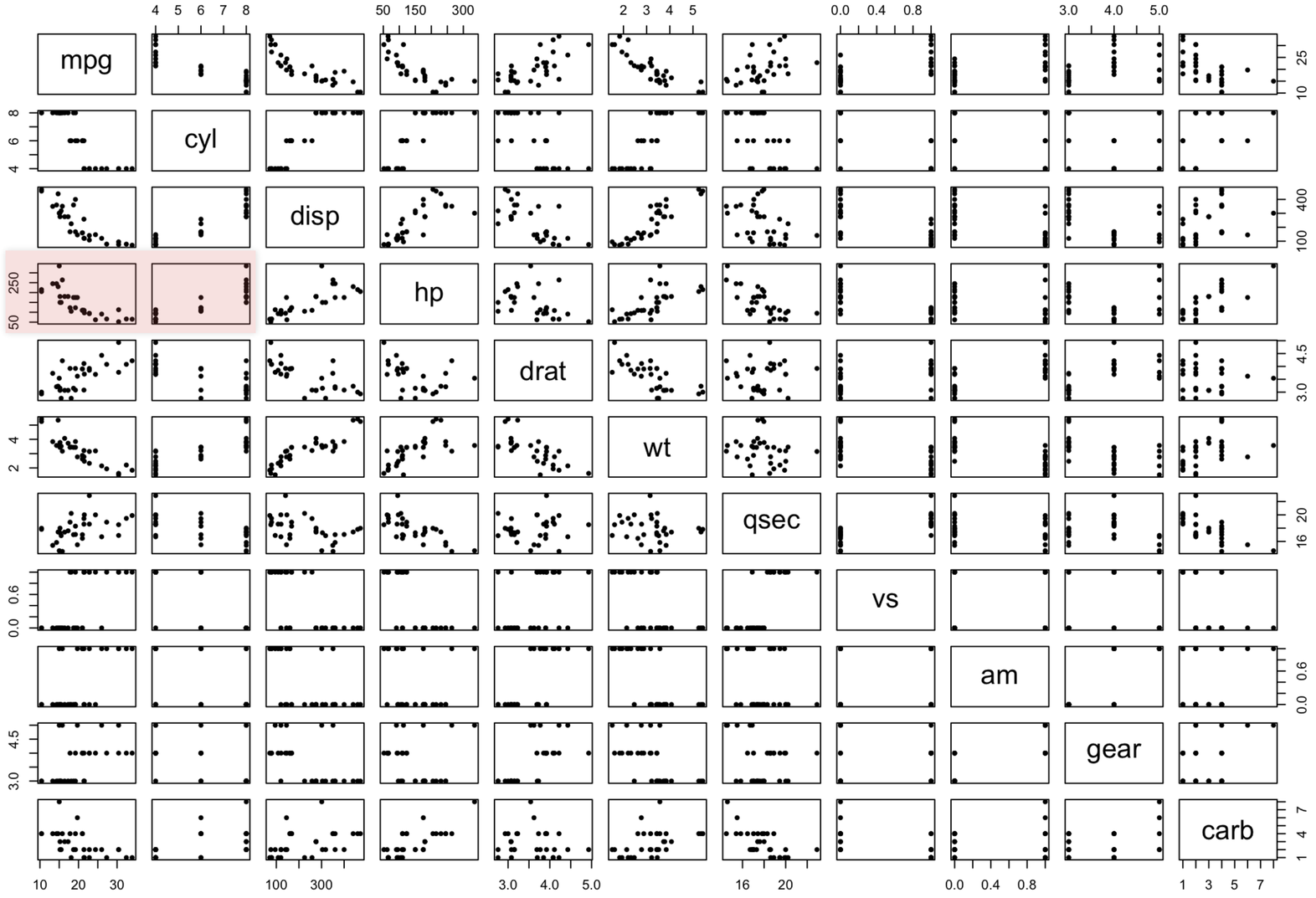
```
boxplot(mtcars) # make a boxplot of variables across car models
```





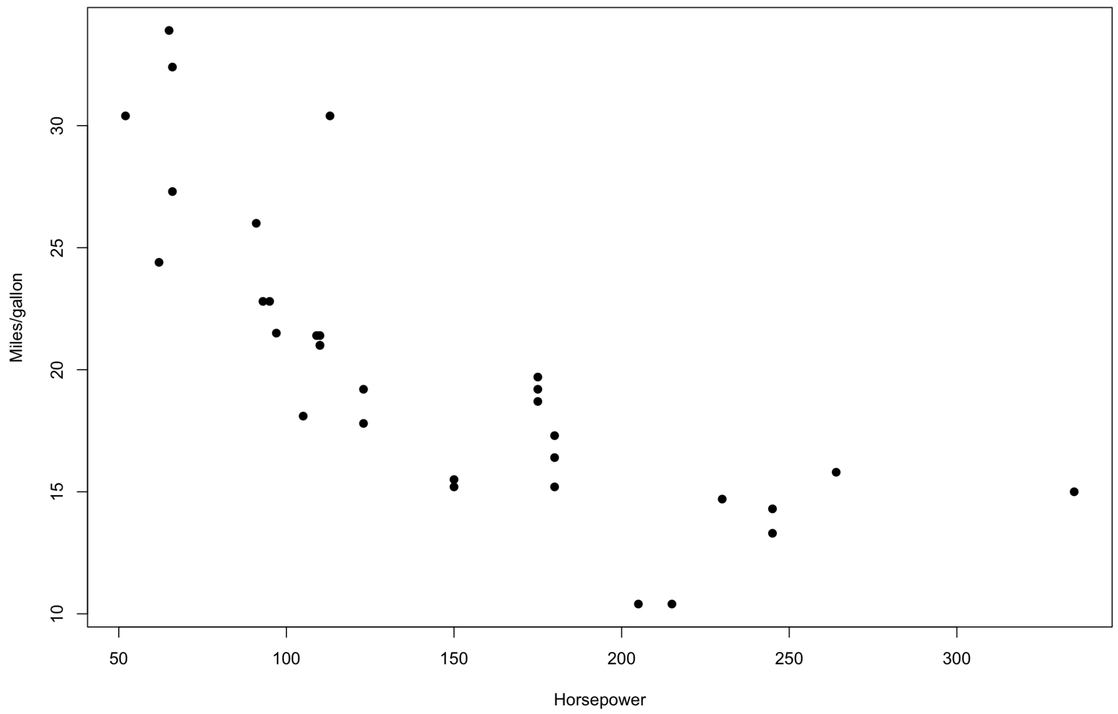

```
14 # [, 1]  mpg  Miles/(US) gallon
15 # [, 2]  cyl  Number of cylinders
16 # [, 3]  disp Displacement (cu.in.)
17 # [, 4]  hp Gross horsepower
18 # [, 5]  drat Rear axle ratio
19 # [, 6]  wt Weight (1000 lbs)
20 # [, 7]  qsec 1/4 mile time
21 # [, 8]  vs Engine (0 = V-shaped, 1 = straight)
22 # [,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

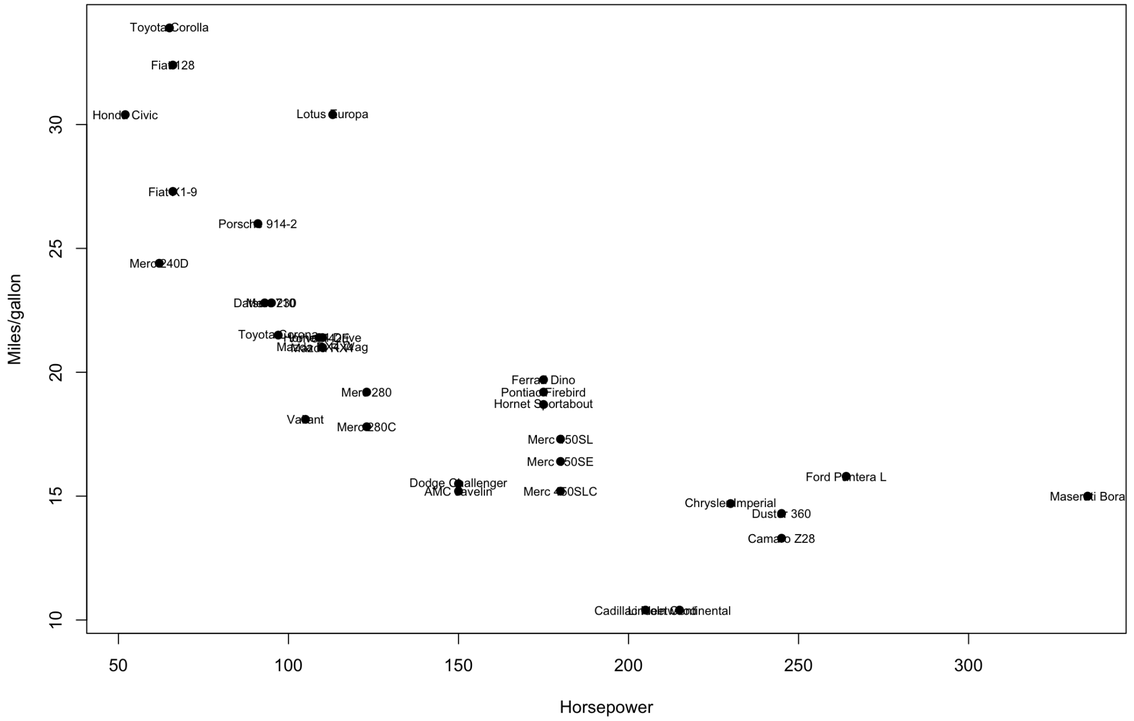
```
plot(mtcars, pch=19, cex=0.6) # make x-y plots for all variables
```



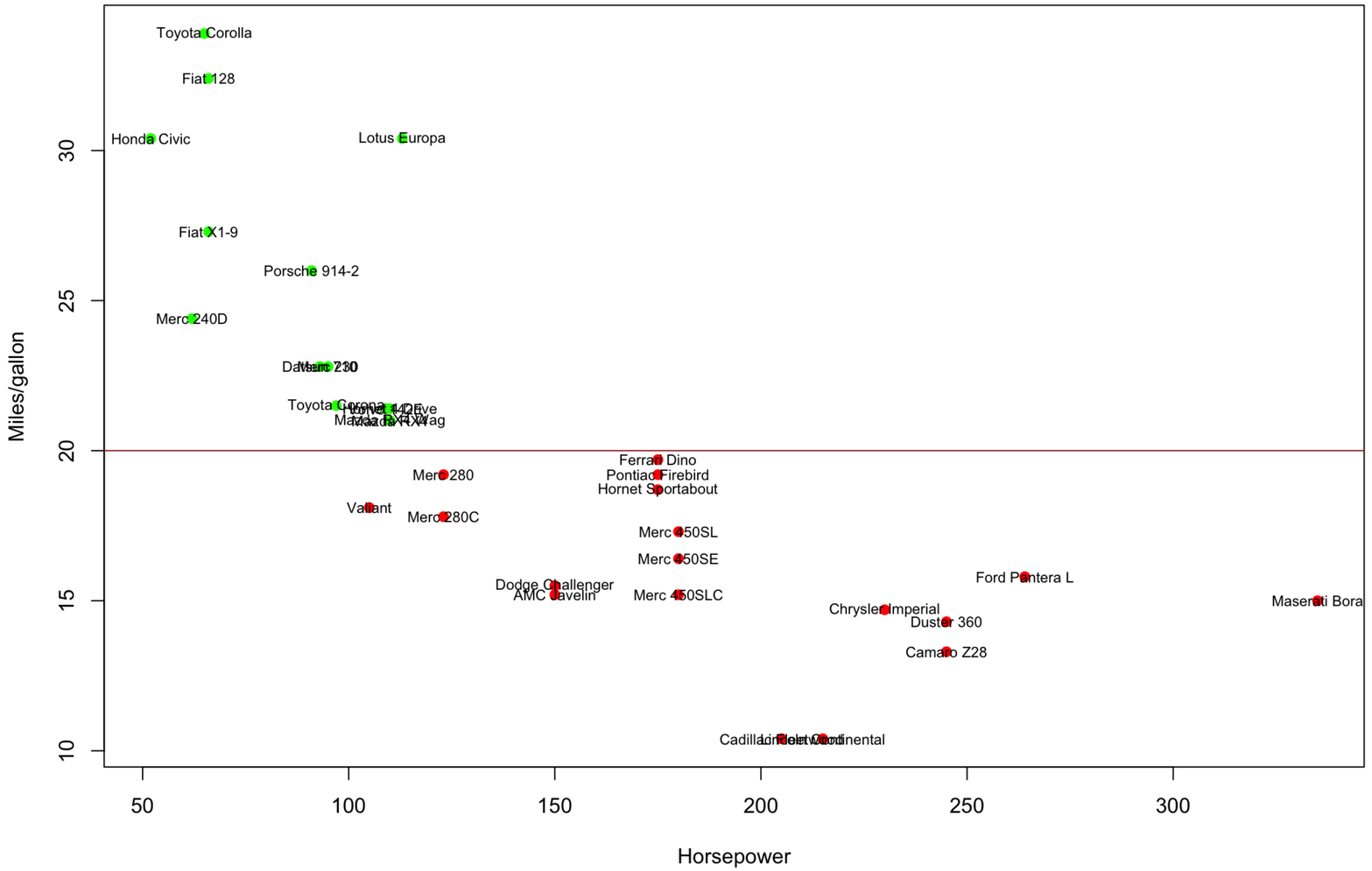

```
14 # [, 1] mpg   Miles/(US) gallon
15 # [, 2] cyl   Number of cylinders
16 # [, 3] disp  Displacement (cu.in.)
17 # [, 4] hp Gross horsepower
18 # [, 5] drat  Rear axle ratio
19 # [, 6] wt Weight (1000 lbs)
20 # [, 7] qsec  1/4 mile time
21 # [, 8] vs Engine (0 = V-shaped, 1 = straight)
22 # [, 9]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

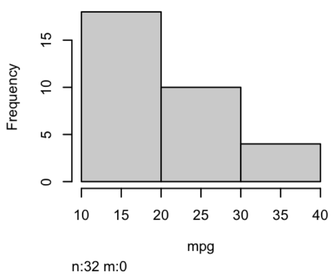


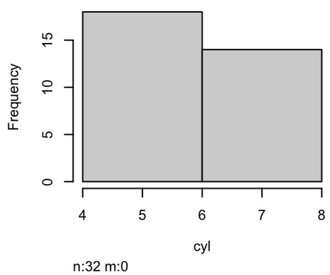


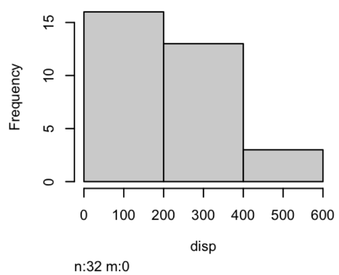
```
plot(mtcars$hp, mtcars$mpg, xlab="Horsepower", ylab="Miles/gallon", pch=19, col=ifelse(mtcars$mpg<20,"red", "green"))
 # We color dots according to a condition (20<x<20 mpg)

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"
```

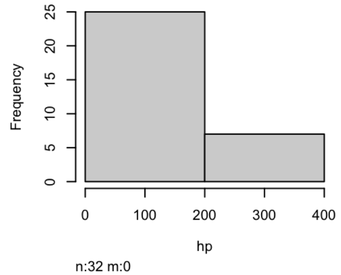


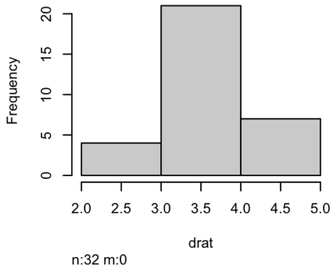
```
hist(mtcars) # we plot an histogram for the different variables
```

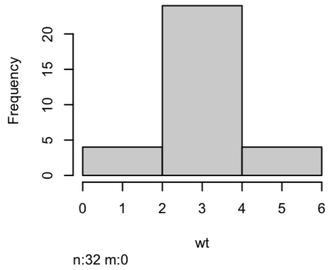


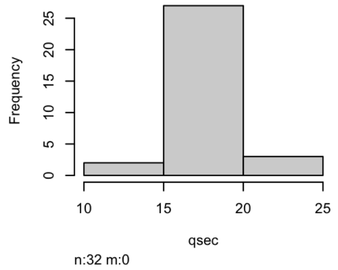


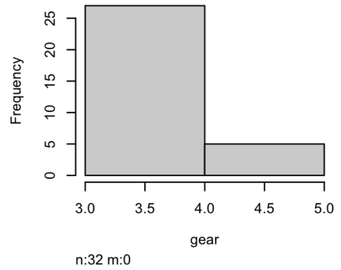


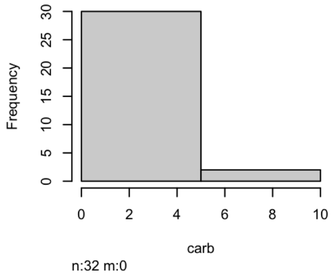






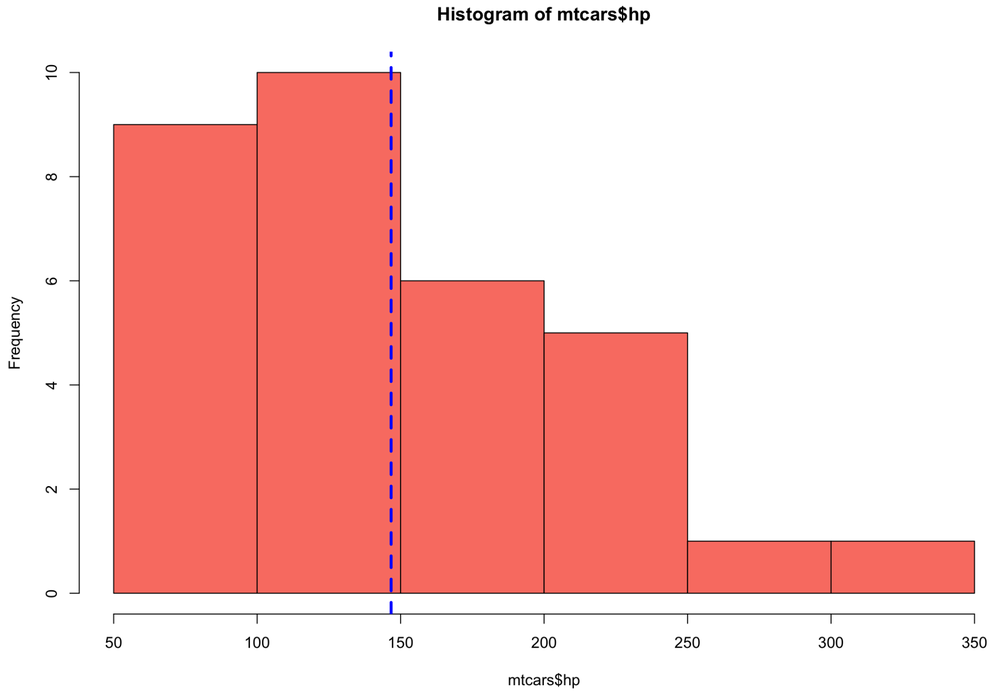


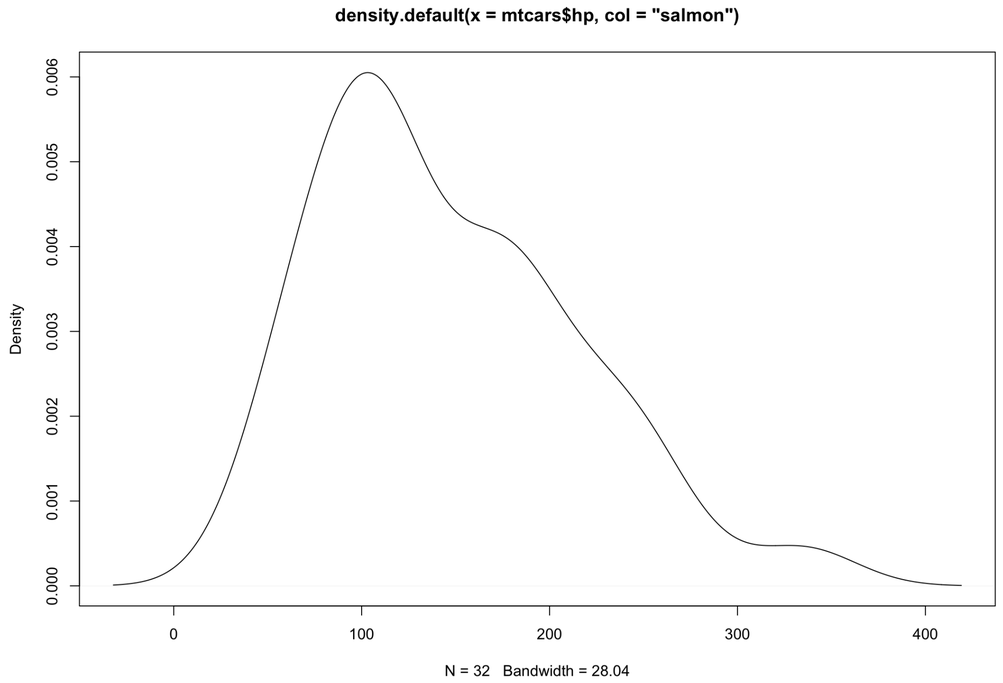




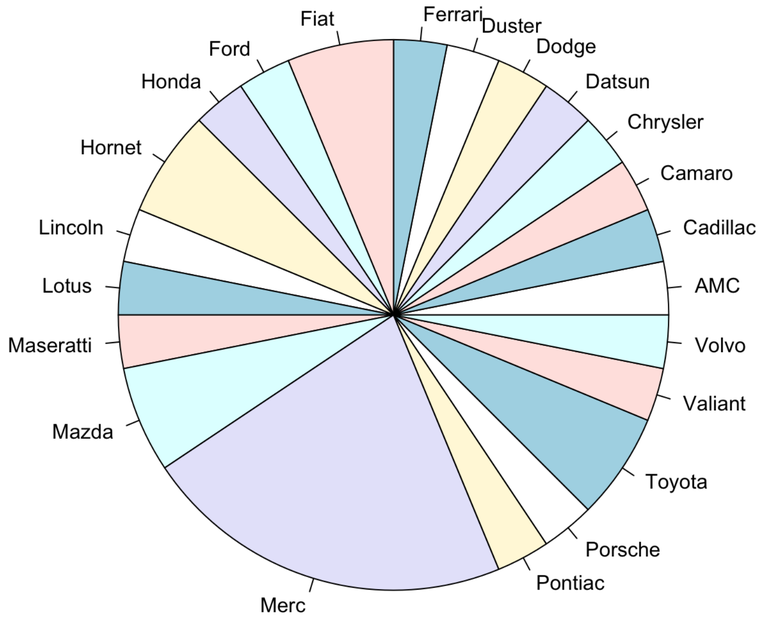

```
14 # [, 1]  mpg  Miles/(US) gallon
15 # [, 2]  cyl  Number of cylinders
16 # [, 3]  disp Displacement (cu.in.)
17 # [, 4]  hp Gross horsepower
18 # [, 5]  drat Rear axle ratio
19 # [, 6]  wt Weight (1000 lbs)
20 # [, 7]  qsec 1/4 mile time
21 # [, 8]  vs Engine (0 = V-shaped, 1 = straight)
22 # [,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





```
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

```
#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
```

Installing and loading packages

```
# 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")
```

-Reproduce all steps shown in this presentation

-The code is available in:

https://github.com/krabberod/BIO9905MERG1_V25/blob/main/Lectures/intro.to.R/Intro.to.R.BIO9905MERG1_V25.R