

```
dens <- density(data, n = npts)
dx <- dens$x
dy <- dens$y
if(add == TRUE)
  plot(0., 0,
       main = main,
       ylab = ylab)
if(orientation == "vertical")
  dx2 <- (dx - min(dx))/max(dx)
  x[1.]
  dy2 <- (dy - min(dy))/max(dy)
  y[1.]
seqbelow <- rep(y[1.], length(dx))
if(Fill == T)
  confshade(dx2, seqbelow, dy2)
```

Introduction to R

Ramiro Logares (ICM-CSIC, Barcelona)

History

- 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 4.2.3
- Developers: international R-core developing team



John Chambers



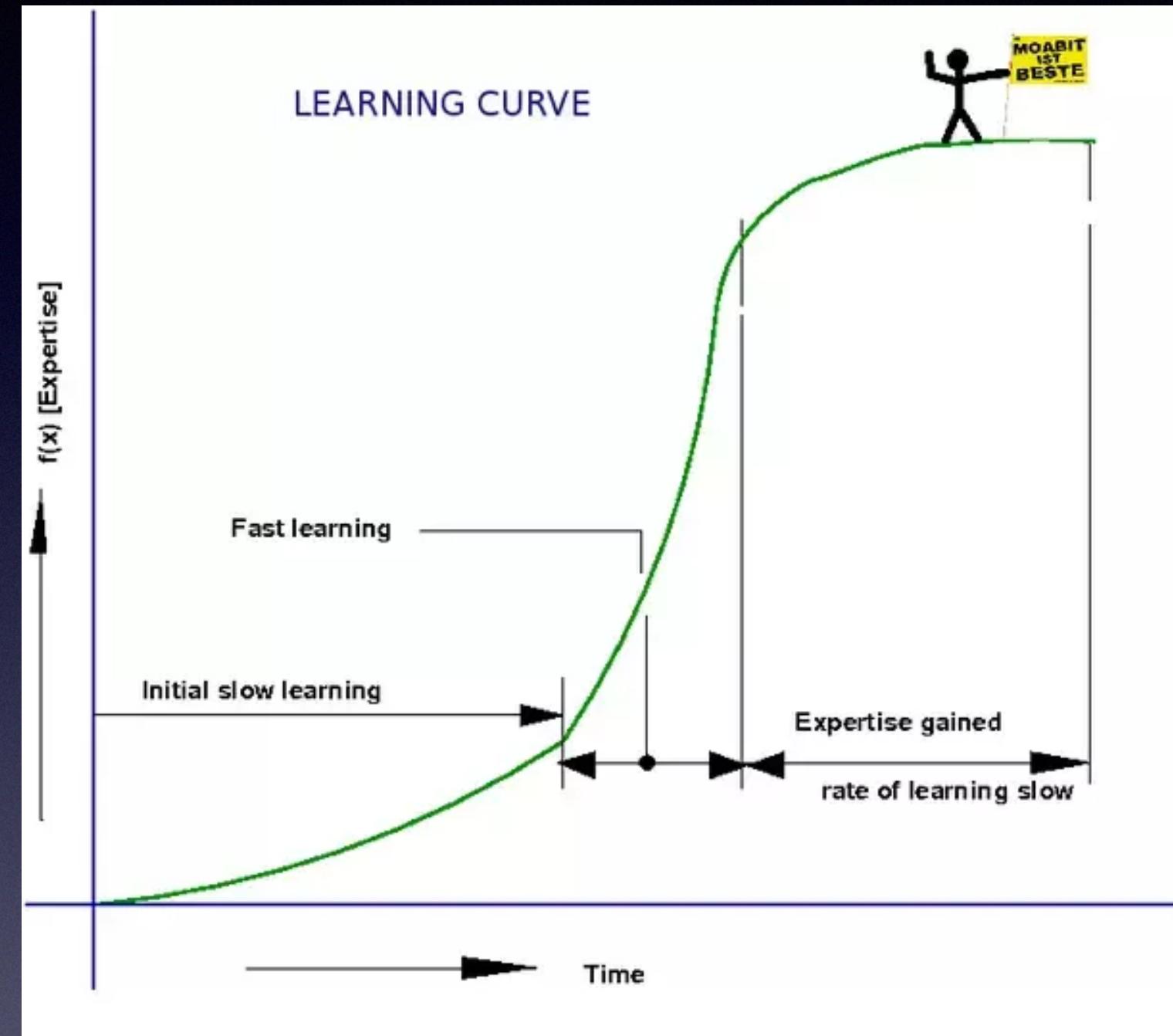
Robert Gentleman



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
- 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



[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](#)

R version 4.2.3 (2023-03-15) -- “Shortstop Beagle”

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 (2023-03-15, Shortstop Beagle) [R-4.2.3.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?

R is ‘GNU S’, a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelling, statistical tests, time series analysis, classification, clustering, etc. Please consult the [R project homepage](#) for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN [mirror](#) nearest to you to minimize network load.

Submitting to CRAN

This year's CRAN submission summer break will be from Jul 21, 2023 to Aug 7, 2023.

To “submit” a package to CRAN, check that your submission meets the [CRAN Repository Policy](#) and then use the [web form](#).

If this fails, send an email to CRAN-submissions@R-project.org following the policy. Please do not attach submissions to emails, because this will clutter up the mailboxes of half a dozen people.

Note that we generally do not accept submissions of precompiled binaries due to security reasons. All binary distribution listed above are compiled by selected maintainers, who are in charge for all

Installing R-Studio

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

RStudio Desktop

Used by millions of people weekly, the RStudio integrated development environment (IDE) is a set of tools built to help you be more productive with R and Python.

1: Install R

RStudio requires R 3.3.0+. Choose a version of R that matches your computer's operating system.

[DOWNLOAD AND INSTALL R](#)

2: Install RStudio

[DOWNLOAD RSTUDIO DESKTOP FOR MACOS 11+](#)

This version of RStudio is only supported on macOS 11 and higher. For earlier macOS environments, please [download a previous version](#).

Size: 374.55 MB | [SHA-256: ED87B818](#) | Version: 2023.03.0+386 |
Released: 2023-03-16

RStudio

Project: (None)

Untitled1* Untitled2* bbmo.sola.R bbmo.sola.tara.R global.soil.vs.global.ocean.R Comm.Ecology.R.BIO9905MERG1... * Run Source

1
2 ## BI09905MERG1_V21
3 ## Community ecology exercises
4 # Install packages
5
6 install.packages("vegan") # Community ecology functions
7
8
9
10
11 # Read dada2 output
12
13 otu.tab<-read.table("https://github.com/path/")
14
15
16 #Library Vegan
17
18
19
20

Environment History Connections

R Global Environment

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.bioc... winter.vertex.mat List of 6
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_00001" "bp_ASV_00001" "ep_ASV_00001" "ep_ASV_00002" "en_ASV...
bbmo.core.nw.deg.dist num [1:61] 1 1 0.775 0.672 0.538 ...

Editor

Unix terminal

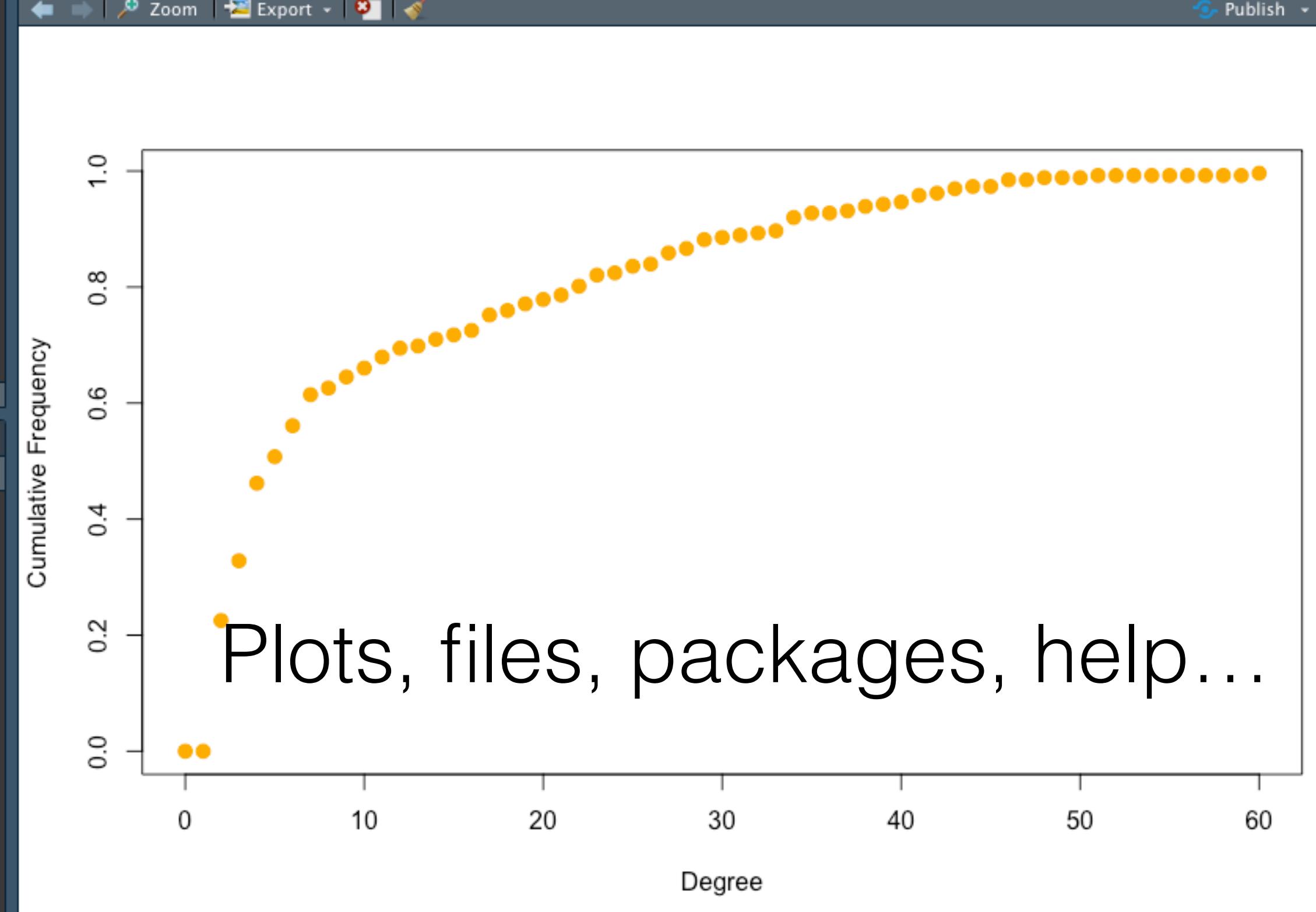
20:1 (Top Level) R Script

Console Terminal x Jobs x

~/

```
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")  
>
```

Console



Presentation format

- The following slides come directly 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)



```
1 ## BI09905MERG1_v23
2
3 # Intro to R
4
5 # <- use this symbol for lines that R should not interpret or as comments
   to yourself (highly recommended)
6
7 # Help
8 # Use "?" before a function. E.g. ?sum
9
10 # Execute from Editor: select the chunk of code to execute with the mouse
    and press then "control+enter"
11 # See the output in the console (+plot area)
```

```
13 #Basic operations
14

15 6+6 # sums two numbers
16 # Result [1] 12
17
18 mysum<-6+6 # sum two integers and assign it to a variable sum
19 mysum = 6+6 # same as above

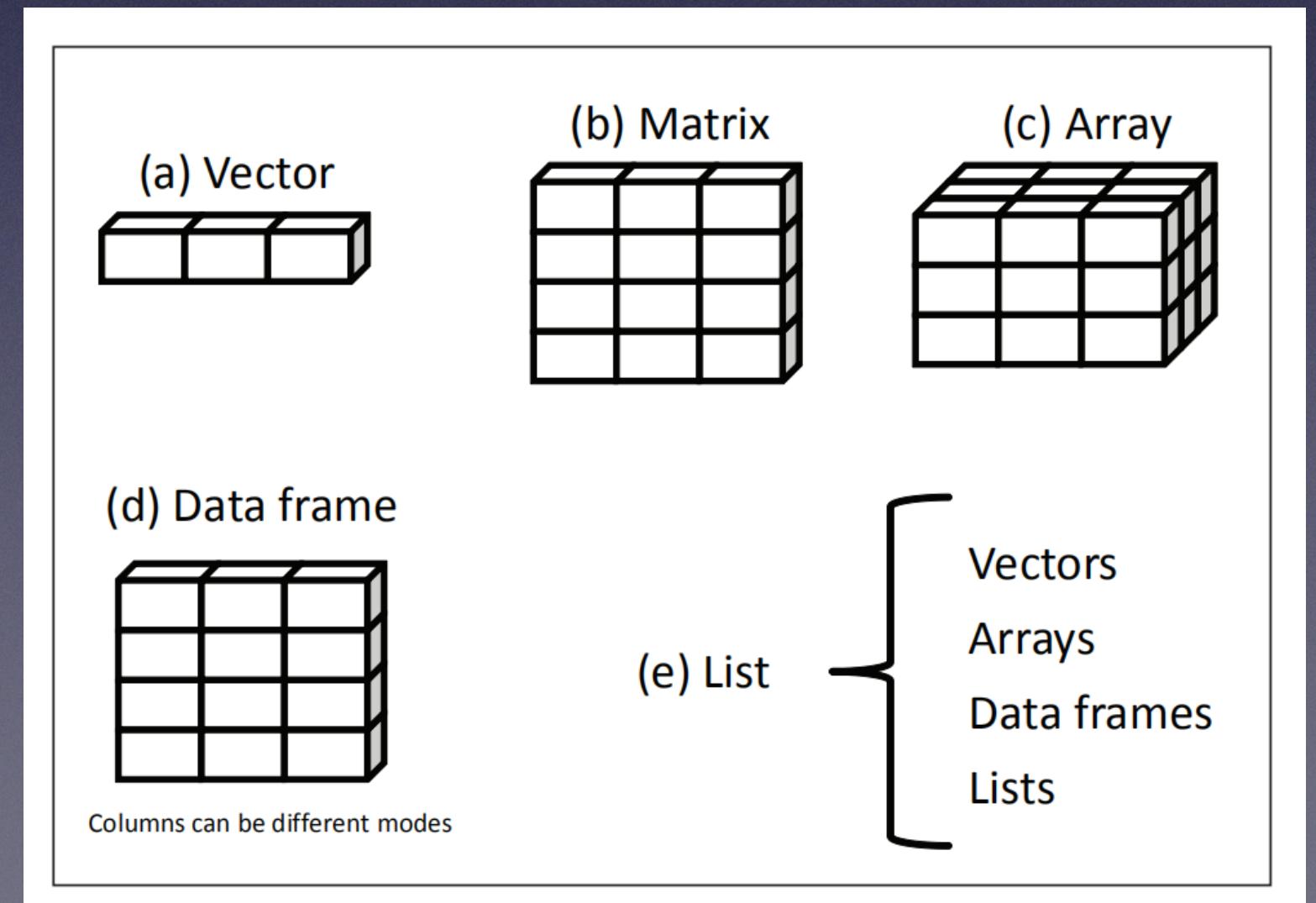
20 # check variable content by executing "mysum"
21 head(mysum) # useful to see the beginning of a variable if it is too long

22
23 #Check variables defined
24 ls() # this info is also shown in the "Environment" panel of RStudio

25
26 #Remove a variable
27 rm(mysum)
28 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



```
2 #Let's have a look to basic datatypes on which R objects are built
3
4 #Numeric: numbers with decimals
5 mynumber<-66.6

6 print(mynumber)
7 # [1] 66.6

8 class(mynumber) # use it to know what is the data type
9 # [1] "numeric"
10
11 #Integer: numbers with no decimals
12 mynumber.int<-as.integer(mynumber)
13 # [1] 66

14 class(mynumber.int)
15 # [1] "integer"
```

```
17 #Character: can be a letter or a combination of letters enclosed  
by quotes
```

```
18 mychar<-"bioinfo course"  
19 print(mychar)  
20 # [1] "bioinfo course"
```

```
21 class(mychar)  
22 #[1] "character"  
23  
24 #Logical: a variable that can be TRUE or FALSE (boolean)
```

```
25 im.true<-TRUE  
26 print(im.true)  
27 #[1] TRUE
```

```
28 class(im.true)  
29 # [1] "logical"
```

1 #vectors

Vectors

2 # Objects that are used to store values or other information of the same data type

3 # They are created with the function "c()" that will generate a 1D array

4 species<-c(123,434,655,877,986) # we create a numeric vector

5 class(species)

6 #[1] "numeric"

7 length(species) # number of elements in the vector

8 #[1] 5

9 species[5] # accessing the fifth element in the vector

10 #[1] 986

11 species[1:3]

12 #[1] 123 434 655

13 sum(species) # sum the values in the vector

14 #[1] 3075

Vectors

```
16 species.names<-c("dog","lion","human","pig","cow") # we create a character vector
17 class(species.names)
18 # [1] "character"
19

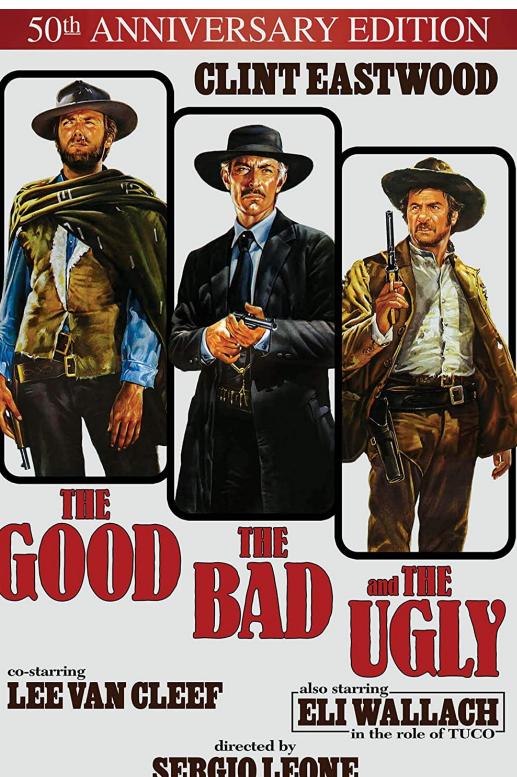
20 seq.num<-c(1:100) # we create a sequence of numbers
21
22 # [1] 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
23 #[33] 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
24 #[65] 65 66 67 68 69 70 71 72 73 74 75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

27 seq.num.by2<-seq(1,100, 2) # same sequence as above but taken by 2
28
29 # [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
30 # [44] 87 89 91 93 95 97 99
31

32 seq.num.by2[5:10] # we access the 5th to the 10th element
33 # [1] 9 11 13 15 17 19
```

Factors

```
1 #Factor: used to refer to a qualitative relationship.  
  
2 # to generate a factor, we'll use a vector defined with the function c()  
3 myfactor<-factor(c("good", "bad", "ugly", "good","good","bad", "ugly"))  
  
4 print(myfactor)  
5 #[1] good bad ugly good good bad ugly  
6 #Levels: bad good ugly <- NB: levels of the factor  
  
7 class(myfactor)  
8 #[1] "factor"  
  
9 levels(myfactor) # this can be used to check the levels of a factor  
10 #[1] "bad" "good" "ugly"  
  
11 nlevels(myfactor)  
12 #[1] 3  
  
13 class(levels(myfactor))  
14 #[1] "character"
```



List

```
1 #List
2 #It can contain elements of various data types
  e.g.vectors, functions, matrices, another list)

3 # Example of vectors with three different data types in one list
4 list1<-c(1:5) # integer vector
5 #[1] 1 2 3 4 5
6 list2<-factor(1:5) # factor vector
7 #[1] 1 2 3 4 5
8 # Levels: 1 2 3 4 5
9 list3<-letters[1:5]
10 #[1] "a" "b" "c" "d" "e"

11 grouped.lists<-list(list1,list2,list3)
12 #[[1]]
13 #[1] 1 2 3 4 5
14
15 #[[2]]
16 #[1] 1 2 3 4 5
17 #Levels: 1 2 3 4 5
18
19 #[[3]]
20 #[1] "a" "b" "c" "d" "e"
```

```
22 #Accessing elements of a list

23 grouped.lists[[1]] # accessing the first vector
24 # [1] 1 2 3 4 5

25 grouped.lists[[3]][5] # accessing the 5th element from the third vector
26 # [1] "e"
27
28 #Ungroup the list
29 ungrouped.list<-unlist(grouped.lists)
30 # [1] "1" "2" "3" "4" "5" "1" "2" "3" "4" "5" "a" "b" "c" "d" "e"

31 class(ungrouped.list)
32 # [1] "character" # NB: the list becomes a character datatype
33 length(ungrouped.list)

34 # [1] 15
```

Matrix

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

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

Dataframes

```
1 #Dataframes  
2 # More general than a matrix and can contain different data types  
3 # Variables or features are in columns, while observations are in rows  
4 # =>NB: this is one of the most common objects in metabarcoding analyses<=  
5 # Generated with the data.frame() function  
6  
7 my.data.frame<-data.frame(  
8   Name=c("Game of Thrones", "MrRobot", "WestWorld"),  
9   Budget=c(344,59,122),  
10  Seasons=c(8,4,3),  
11  Audience=c(300,14,80),  
12  Actors=c(221,56, 90)  
13 )
```

data frame

	numeric	character	logical
1		"R"	TRUE
2		"S"	FALSE
3		"T"	TRUE

Dataframes

```
14 print(my.data.frame)
15 #          Name Budget Seasons Audience Actors
16 #1 Game of Thrones    344      8       300     221
17 #2 MrRobot           59       4       14      56
18 #3 WestWorld         122      3       80      90
19

20 row.names(my.data.frame)<-my.data.frame[,1] # Assign to the row names the
names in the first column

21 my.data.frame<-my.data.frame[,-1] # Remove the first column

22 print(my.data.frame) # By clicking this object in the "Environment" panel
on the right, you'll see a window with the dataframe
23
24 #          Budget Seasons Audience Actors
25 # Game of Thrones    344      8       300     221
26 # MrRobot           59       4       14      56
27 # WestWorld         122      3       80      90
```

Dataframes

```
1 class(my.data.frame)
2 # [1] "data.frame"
3 ncol(my.data.frame) # Number of columns
4 # [1] 4
5 nrow(my.data.frame) # Number of rows
6 # [1] 3

7 colnames(my.data.frame) # Column names
8 # [1] "Budget"    "Seasons"   "Audience"  "Actors"

9 rownames(my.data.frame) # Name of rows
10 # "Game of Thrones"    "MrRobot"    "WestWorld"

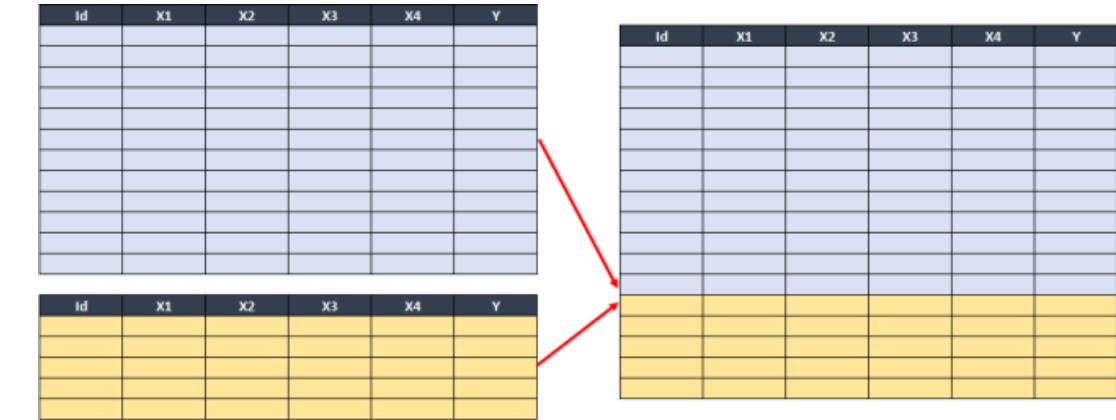
11 colSums(my.data.frame) # Sum values in columns
12 # Budget  Seasons Audience  Actors
13 # 525      15       394      367

14 rowSums(my.data.frame) # We sum the values, even if they make no sense in
the example
15 # Game of Thrones          MrRobot          WestWorld
16 # 873                      133              295
```

Dataframes

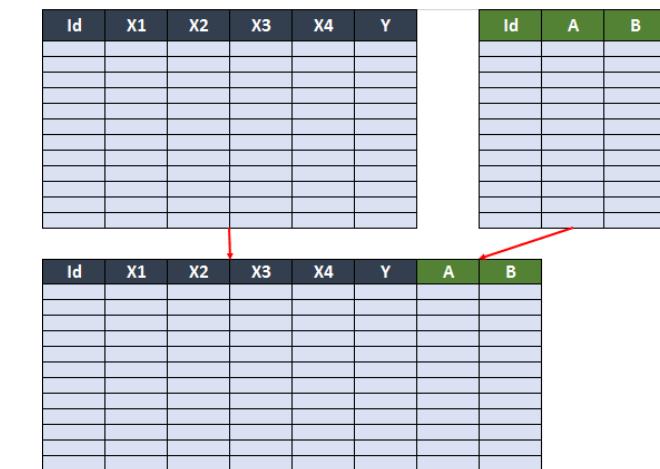
```
1 rbind(my.data.frame,my.data.frame) # appends dataframes one below the other  
(column names identical)
```

		Budget	Seasons	Audience	Actors
2	#				
3	# Game of Thrones	344	8	300	221
4	# MrRobot	59	4	14	56
5	# WestWorld	122	3	80	90
6	# Game of Thrones1	344	8	300	221
7	# MrRobot1	59	4	14	56
8	# WestWorld1	122	3	80	90
9					



```
10 cbind(my.data.frame,my.data.frame) # appends dataframes one next to the other  
      (row names identical)
```

Rank	Name	Budget	Seasons	Audience	Actors	Budget	Seasons	Audience	Actors
11	#								
12	# Game of Thrones	344	8	300	221	344	8	300	221
13	# MrRobot	59	4	14	56	59	4	14	56
14	# WestWorld	122	3	80	90	122	3	80	90



Dataframes

```
16 head(my.data.frame, 2) # Useful to have a look to the beginning of the  
  dataframe (especially useful in big tables)  
17 # Here, asking to print only 2 rows  
18 #  
          Budget Seasons Audience Actors  
19 # Game of Thrones      344        8       300     221  
20 # MrRobot              59         4       14      56  
21  
  
22 my.data.frame[1:2,2:4] # Useful to look at specific sections of the  
  dataframe  
  
23 #  
          Seasons Audience Actors  
24 # Game of Thrones      8       300     221  
25 # MrRobot              4       14      56
```

Dataframes

```
1 #Let's generate a dataframe with different data types
2

3 my.data.frame.2<-data.frame(
4   Name=c("Game of Thrones","MrRobot","WestWorld", "Chernobyl"),
5   Rating=c("Excellent","Very Good","Excellent", "Very Good"),
6   Audience.Restriction=c(TRUE, FALSE, TRUE, FALSE)
7 )

8 print(my.data.frame.2)
#          Name    Rating Audience.Restriction
10 # 1 Game of Thrones Excellent                  TRUE
11 # 2 MrRobot      Very Good                 FALSE
12 # 3 WestWorld    Excellent                  TRUE
13 # 4 Chernobyl   Very Good                 FALSE
```

```
14 #Rename row names  
15 row.names(my.data.frame.2)<-my.data.frame.2[,1]  
16 my.data.frame.2<-my.data.frame.2[,-1] # Remove redundant column 1  
17  
  
18 #  
19 # Game of Thrones Excellent TRUE  
20 # MrRobot Very Good FALSE  
21 # WestWorld Excellent TRUE  
22 # Chernobyl Very Good FALSE  
23  
  
24 str(my.data.frame.2) # Let's look at the data types within this dataframe  
25  
26 # 'data.frame': 4 obs. of 2 variables:  
27 # $ Rating : chr "Excellent" "Very Good" "Excellent" "Very Good"  
28 # $ Audience.Restriction: logi TRUE FALSE TRUE FALSE  
29  
30 # Variables, in this case, are characters and logical (TRUE/FALSE)  
31
```

Dataframes

```
1 #Merge two dataframes based on a pattern  
2 # We will use the series names to merge these dataframes as this is what they have in common  
3  
4 data.frame.large<-merge(my.data.frame, my.data.frame.2, by="row.names")  
# "by" indicates the column used for merging  
  
6 #           Row.names Budget Seasons Audience Actors      Rating Audience.Restriction  
7 # 1 Game of Thrones    344       8     300    221 Excellent          TRUE  
8 # 2 MrRobot            59        4      14     56 Very Good        FALSE  
9 # 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

```
1 #Useful commands to work with tables or dataframes  
  
2 getwd()          # get working directory  
3 # [1] "/Users/admin"  
4 setwd("path/to/my/directory") # set working directory  
5  
6 my.table<-read.table(file="table.tsv", sep="\t", header=T) # read table;  
several other options are available  
  
7 dim(my.table)      # Table dimensions  
8 nrow(my.table)     # Number of rows  
9 ncol(my.table)     # Number of columns  
10 colnames(my.table) # Name of columns  
11 rownames(my.table) # Name of rows  
12 colSums(my.table)  # Sum of numeric values in columns  
13 rowSums(my.table)  # Sum of numeric values in rows  
14 head(my.table)     # See table header  
15 t(my.table)        # Transpose table
```

17 #Table subsetting

18 # Format: my.table[row, column]

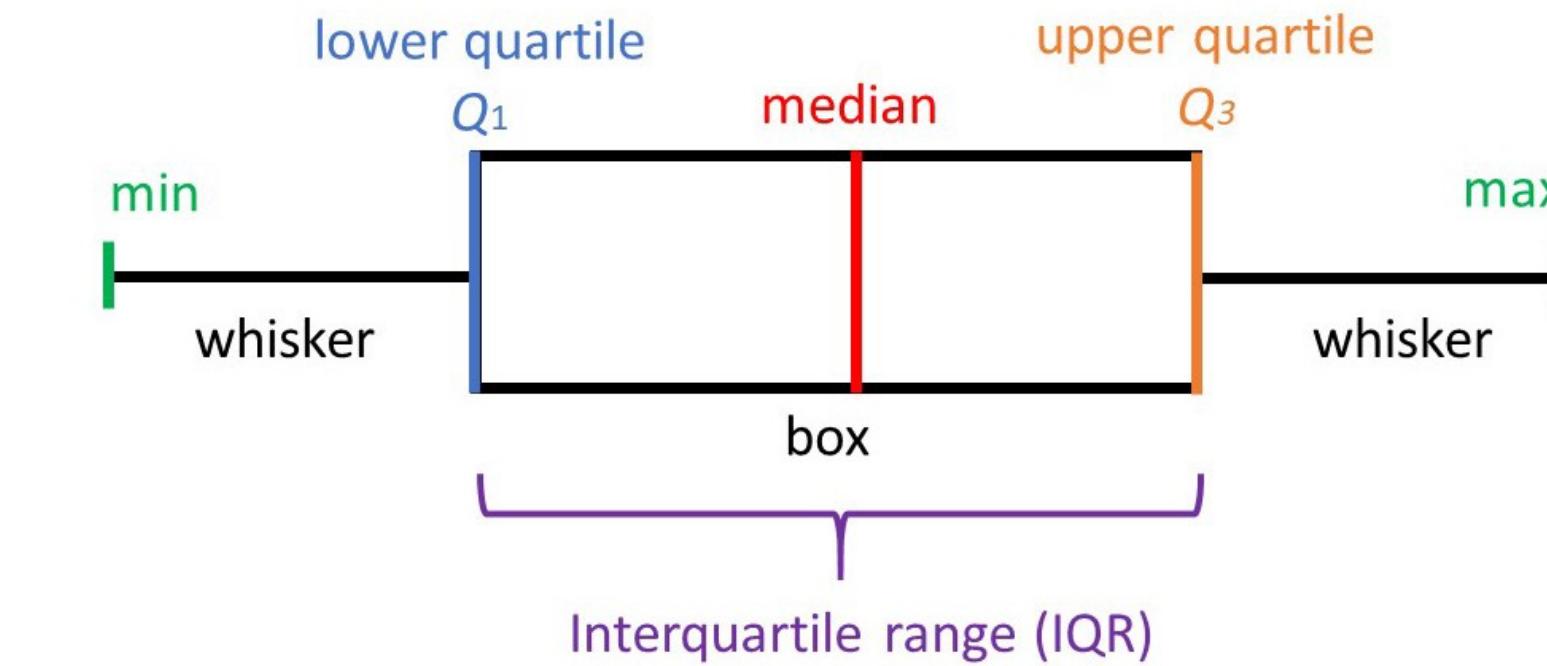
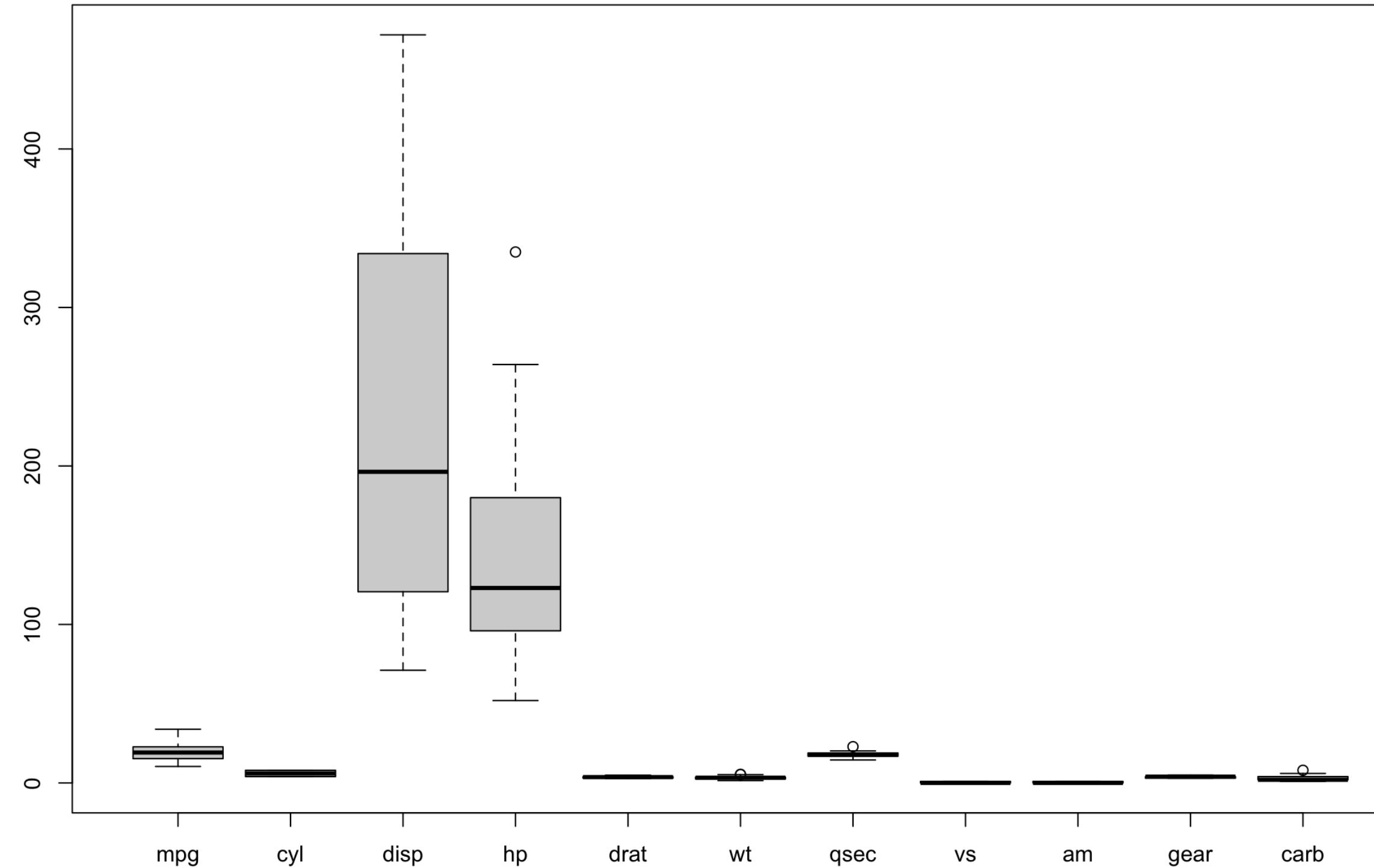
```
19 my.table[1,2]                      # Get value from row 1, column 2
20 my.table[1,]                        # Get values from row 1 across all columns
21 my.table$column.name<-NULL         # Remove column
22 my.table[-5,-2]                    # Remove row 5 and column 2
23 my.table[-(5:10),]  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

Plotting

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

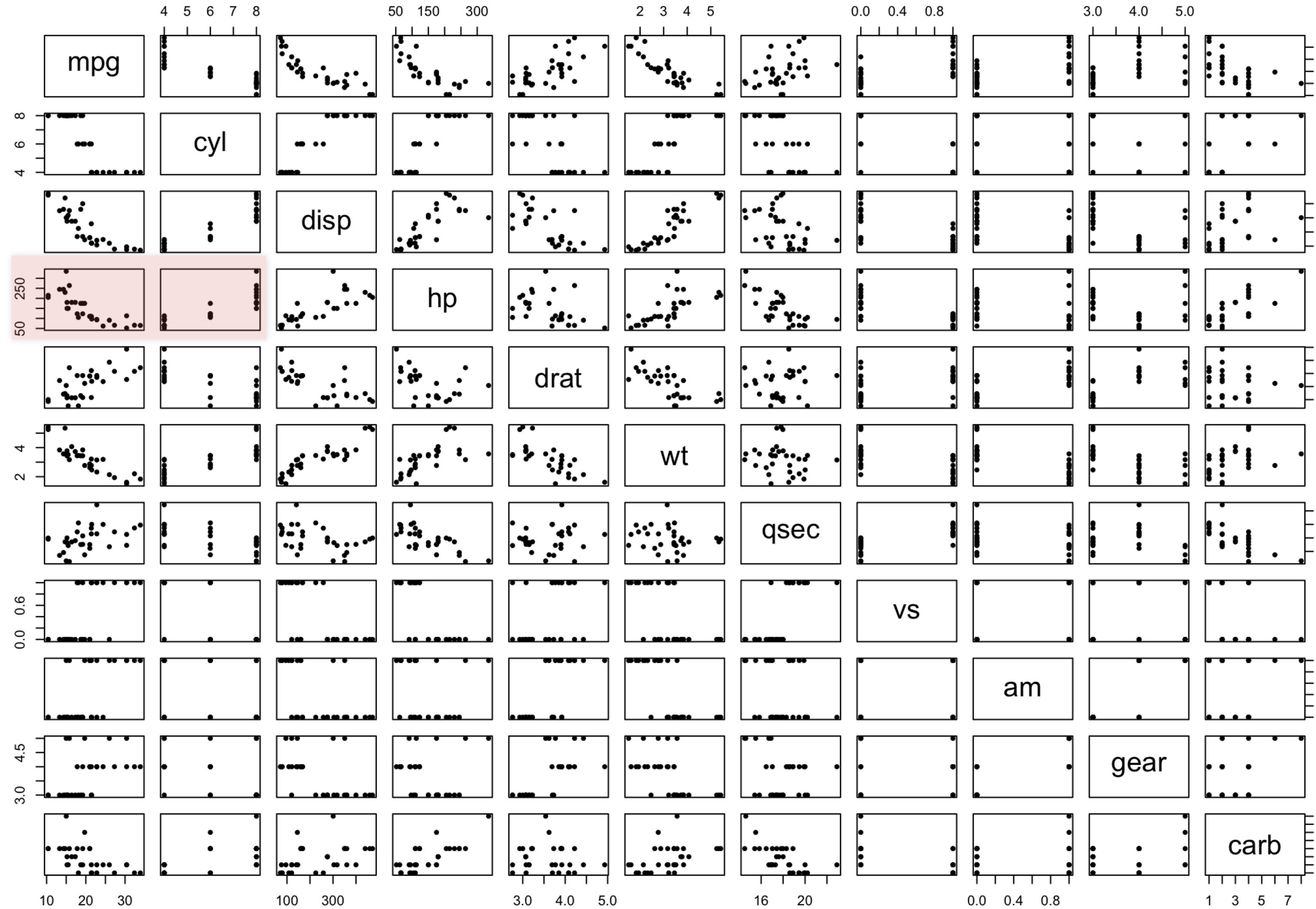
6 #Data structure
7 #
8 #          mpg cyl  disp  hp drat    wt  qsec vs am gear carb
9 # Mazda RX4        21.0   6 160.0 110 3.90 2.620 16.46  0  1     4     4
10 # Mazda RX4 Wag      21.0   6 160.0 110 3.90 2.875 17.02  0  1     4     4
11 # Datsun 710       22.8   4 108.0  93 3.85 2.320 18.61  1  1     4     1
12 # Hornet 4 Drive     21.4   6 258.0 110 3.08 3.215 19.44  1  0     3     1
13
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
```

```
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
```

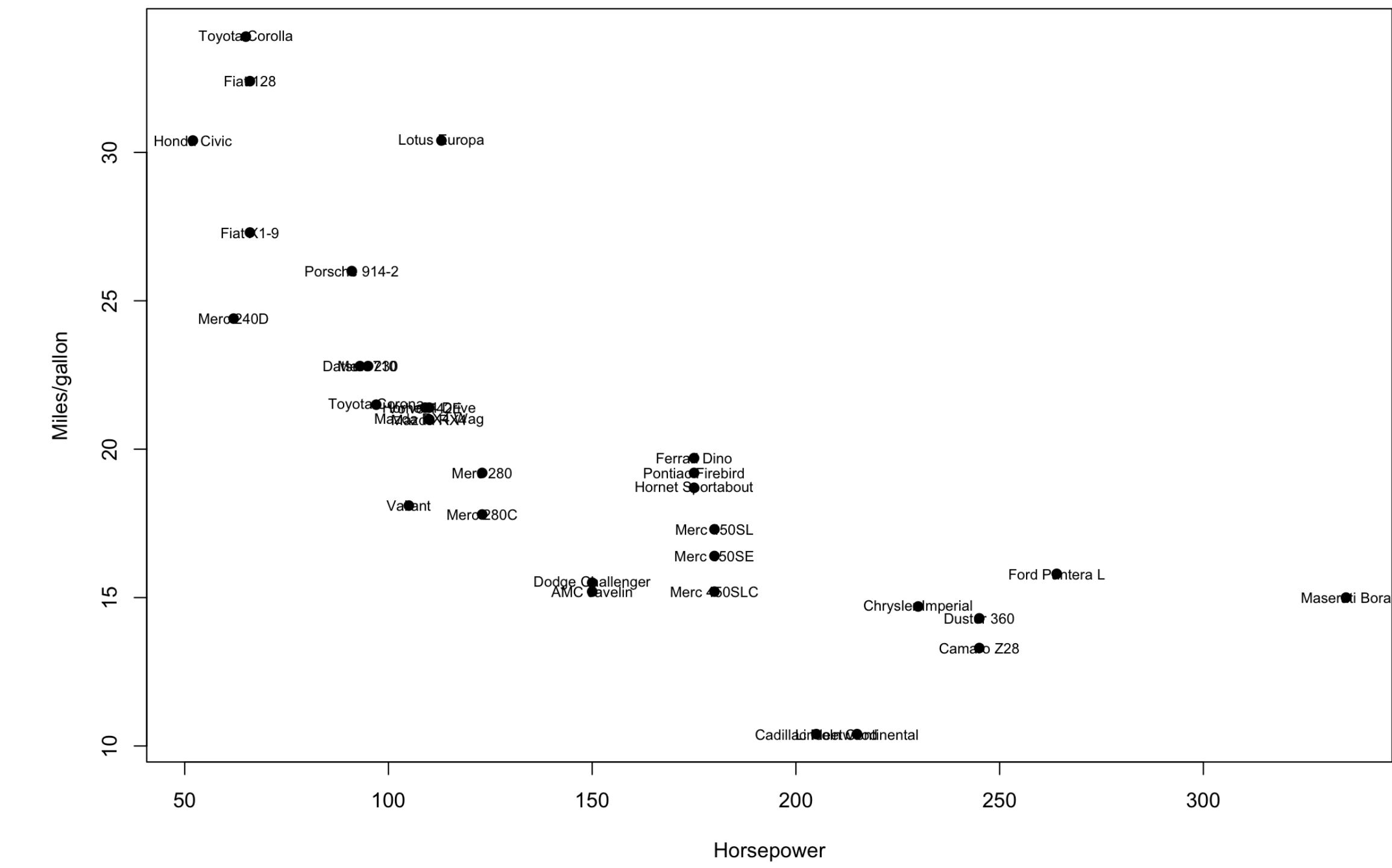
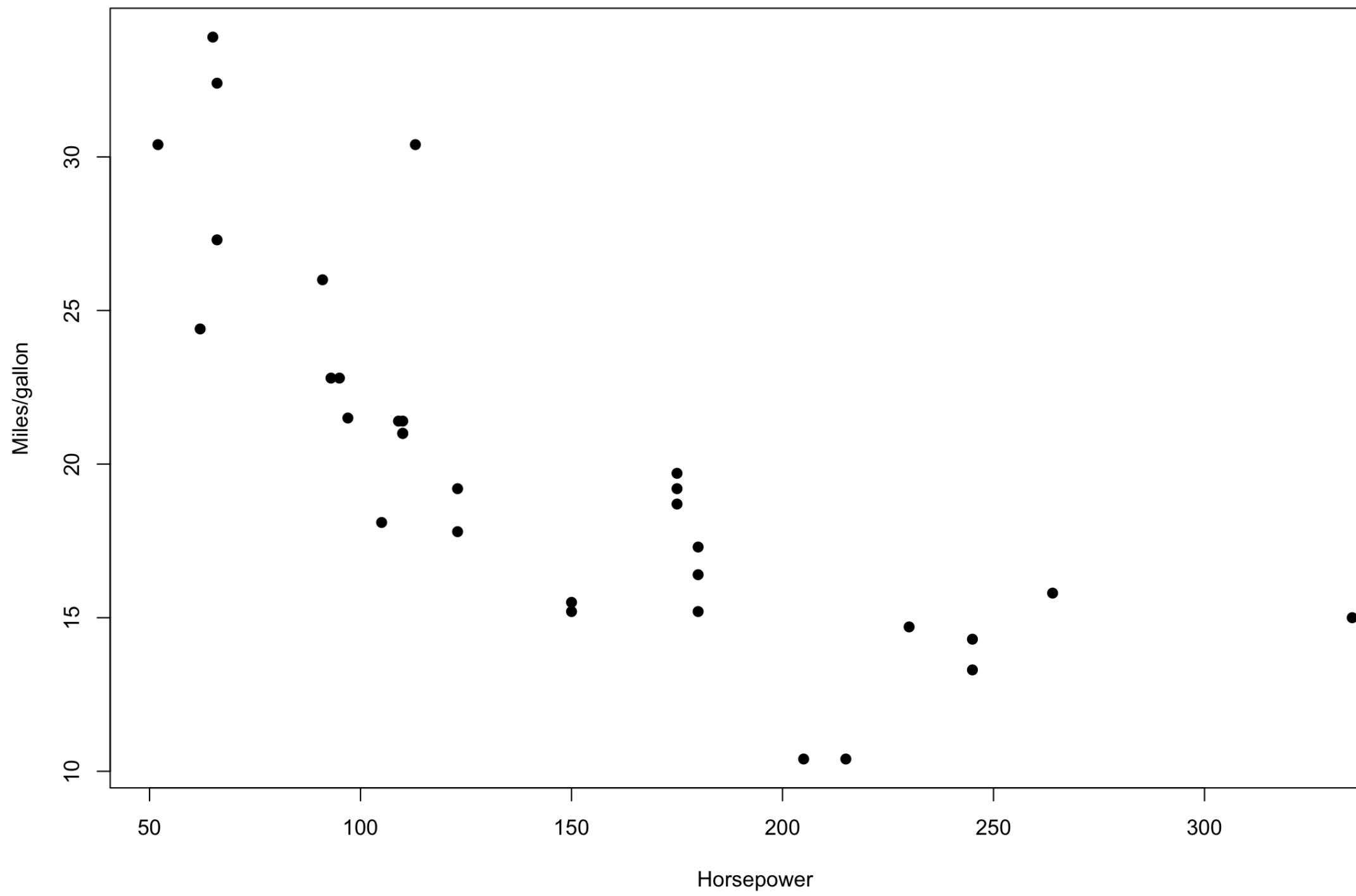
```
plot(mtcars, pch=19, cex=0.6) # make x-y plots for all variables
```



```
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
```



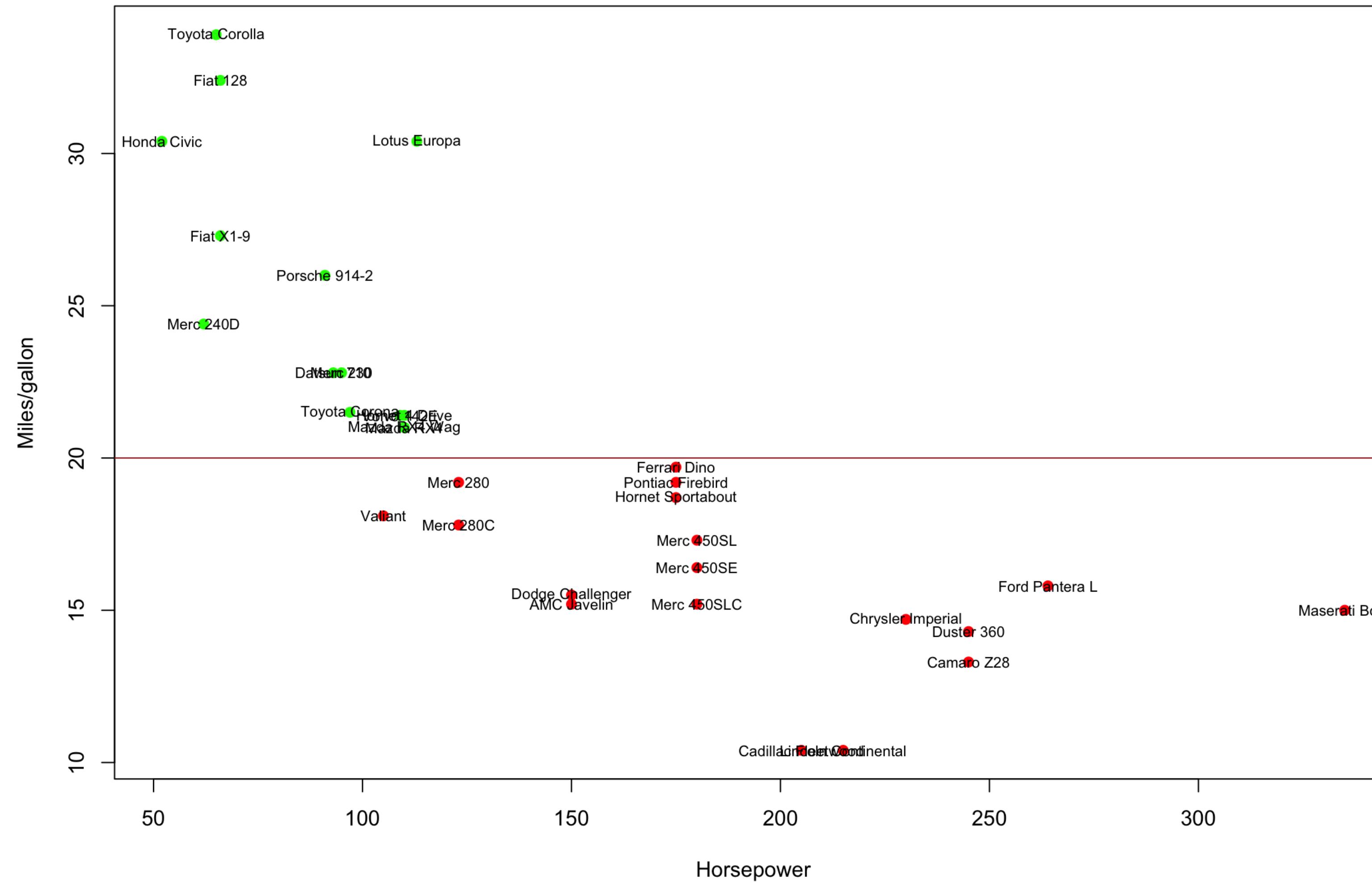
```

1 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)

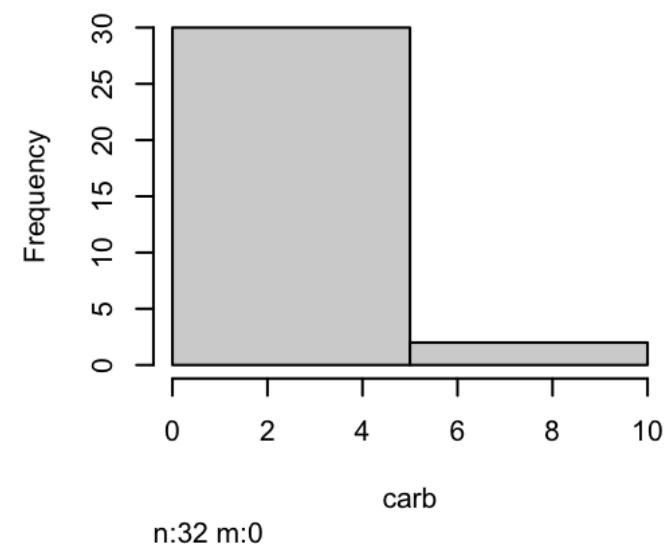
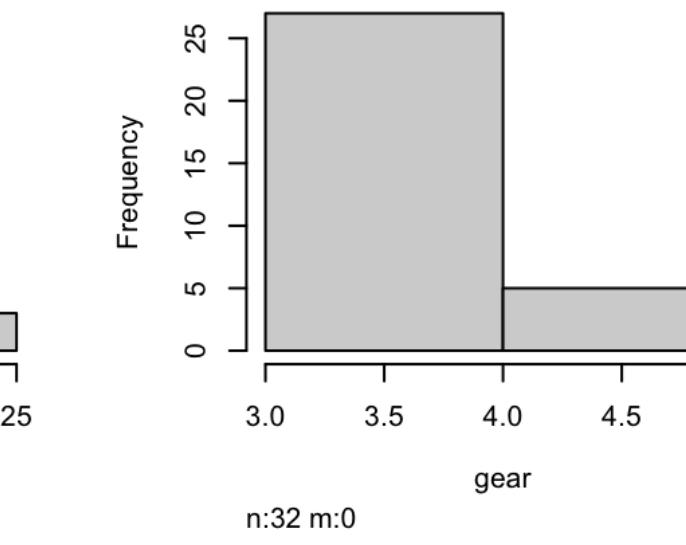
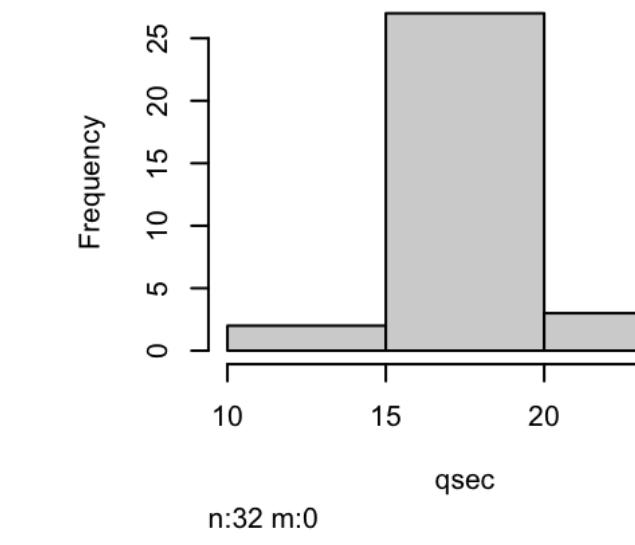
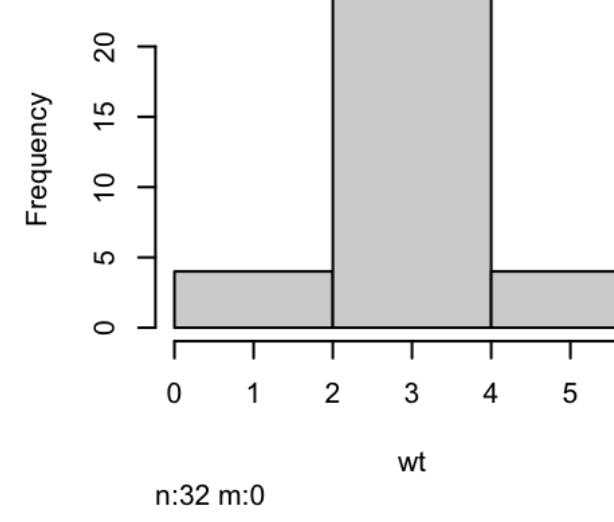
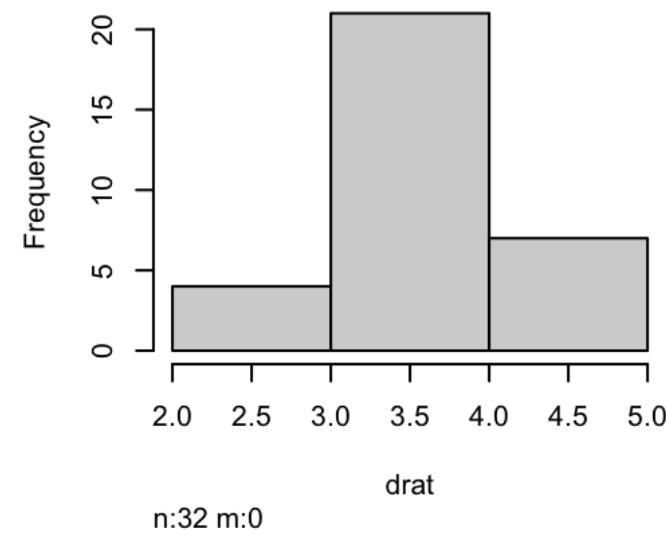
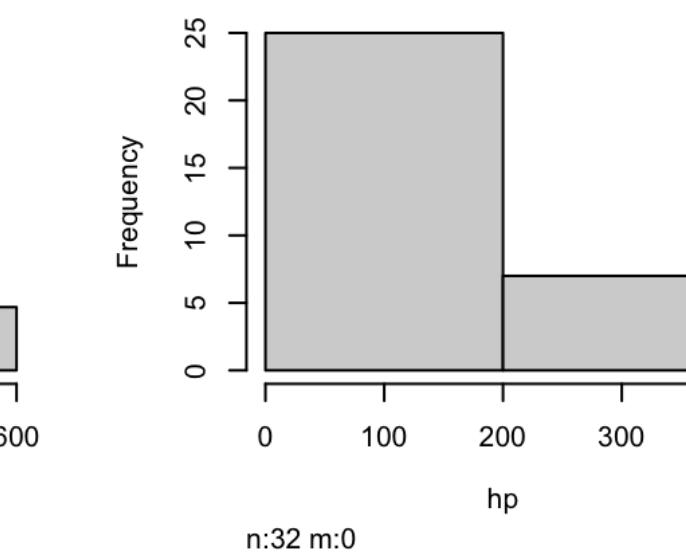
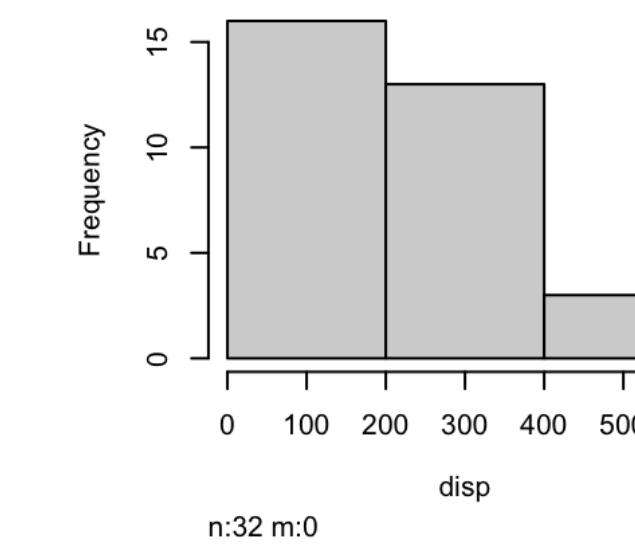
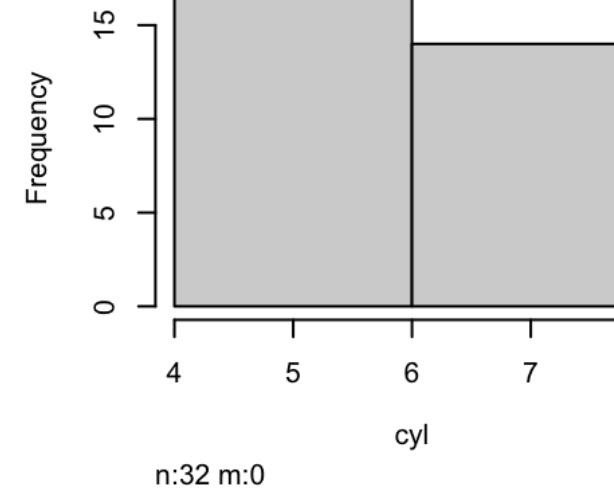
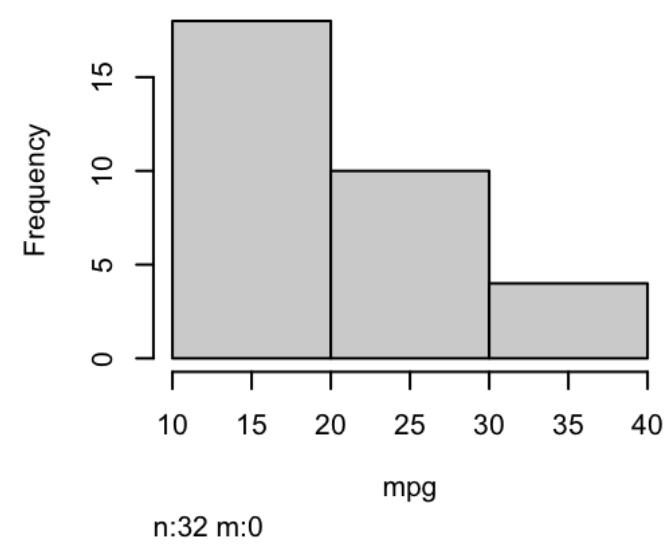
3 text(mtcars$hp, mtcars$mpg, row.names(mtcars), cex=0.7) # we add the car model

4 abline(h=20, col="brown") # we add an horizontal line at "20"

```



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

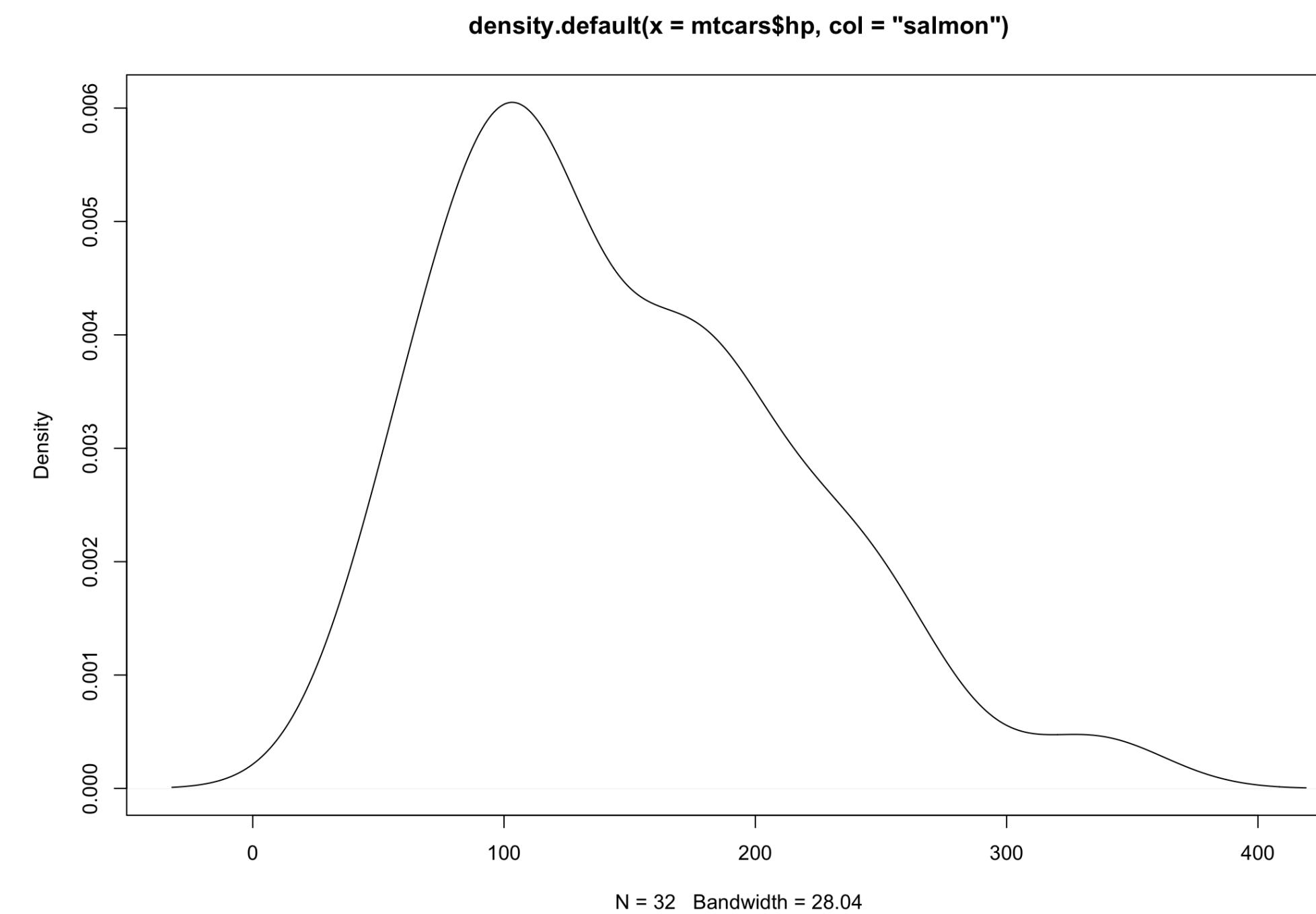
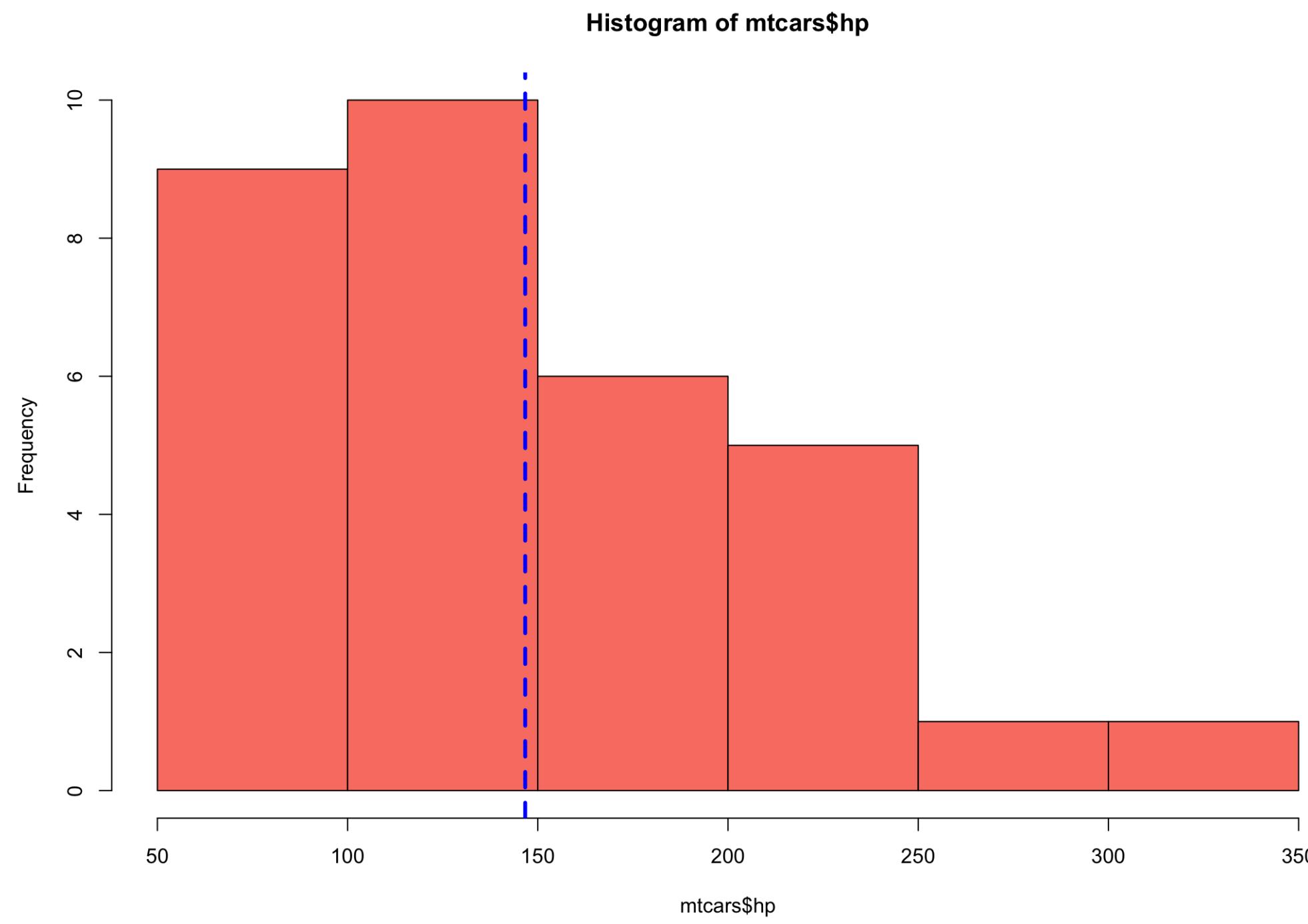


```
14 # [, 1] mpg Miles/(US) gallon  
15 # [, 2] cyl Number of cylinders  
16 # [, 3] disp Displacement (cu.in.)  
17 # [, 4] hp Gross horsepower  
18 # [, 5] drat Rear axle ratio  
19 # [, 6] wt Weight (1000 lbs)  
20 # [, 7] qsec 1/4 mile time  
21 # [, 8] vs Engine (0 = V-shaped, 1 = straight)  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



```

1 brands<-c("Mazda", "Mazda", "Datsun", "Hornet", "Hornet", "Valiant", "Duster", "Merc", "Merc", "Merc",
  "Merc", "Merc", "Merc", "Cadillac", "Lincoln", "Chrysler", "Fiat", "Honda",
  "Toyota", "Toyota", "Dodge", "AMC", "Camaro", "Pontiac", "Fiat", "Porsche", "Lotus", "Ford", "Ferrari",
  "Maseratti", "Volvo")

4 mtcars$brand<-brands # we add an extra column with brands

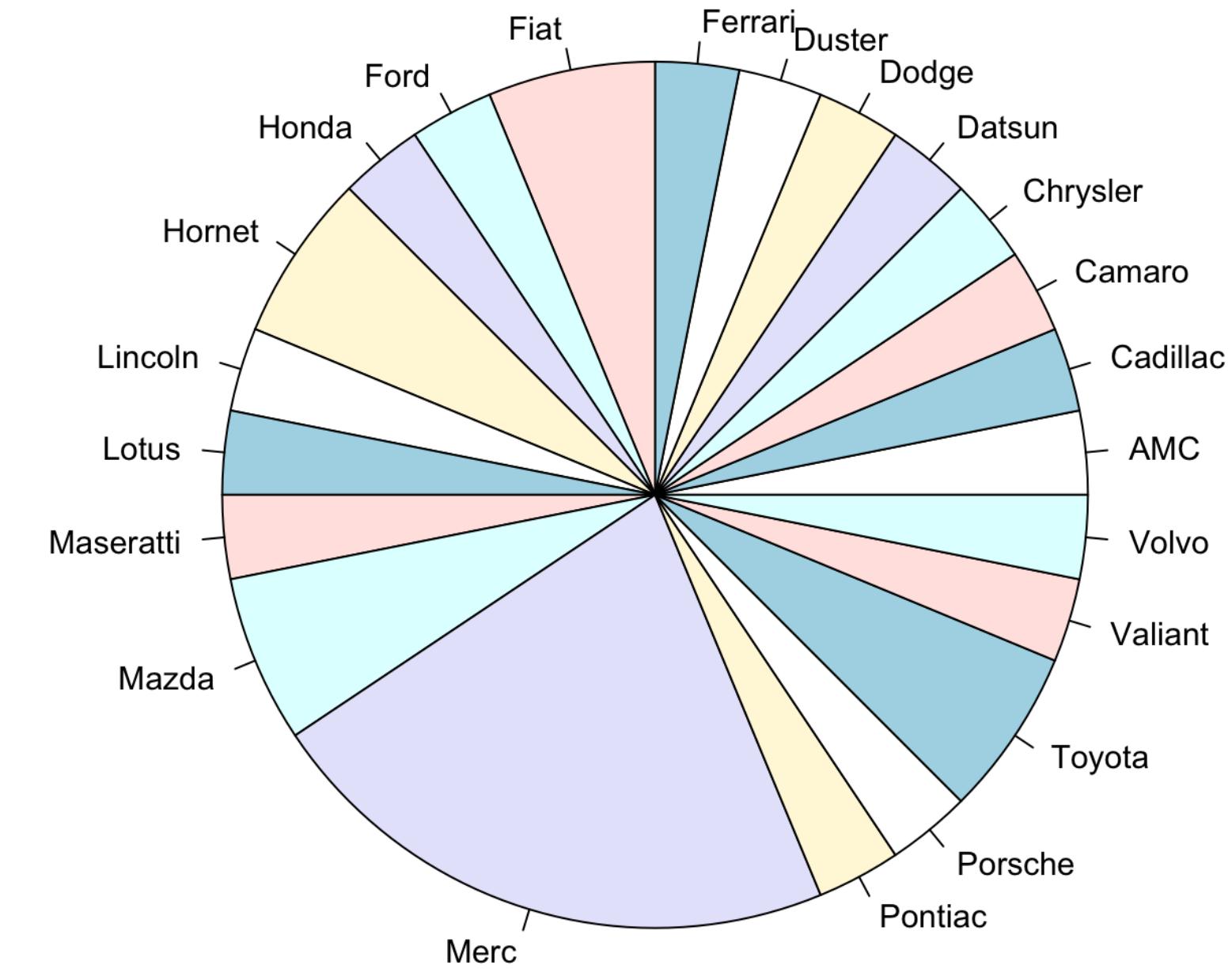
5 mtcars[1:5,] # let's double check
6

7 #
    mpg cyl disp hp drat    wt  qsec vs am gear carb brand
8 # Mazda RX4      21.0   6 160 110 3.90 2.620 16.46  0  1    4    4  Mazda
9 # Mazda RX4 Wag  21.0   6 160 110 3.90 2.875 17.02  0  1    4    4  Mazda
10 # Datsun 710    22.8   4 108  93 3.85 2.320 18.61  1  1    4    1 Datsun
11 # Hornet 4 Drive 21.4   6 258 110 3.08 3.215 19.44  1  0    3    1 Hornet
12 # Hornet Sportabout 18.7   8 360 175 3.15 3.440 17.02  0  0    3    2 Hornet

```

13

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



Installing and loading packages

```
1 #Installing packages
2
3 # R has a large repository of packages for different applications
4
5 install.packages("spaa") # Installs the ecological package spaa
6
7 library("vegan") # load Vegan
8 #Loading required package: permute
9 # Loading required package: lattice
10 # This is vegan 2.5-7
```

Installing and loading packages

```
13 # Other relevant packages
14
15 install.packages("readr")      # To read and write files
16 install.packages("readxl")     # To read excel files
17 install.packages("dplyr")       # To manipulate dataframes
18 install.packages("tibble")      # To work with data frames
19 install.packages("tidyverse")    # To work with data frames
20 install.packages("stringr")     # To manipulate strings
21 install.packages("ggplot2")      # To do plots
22 install.packages("kableExtra")   # necessary for nice table formatting with knitr
23 install.packages("tidyverse")
24
25 if (!requireNamespace("BiocManager", quietly = TRUE))
26   install.packages("BiocManager")
27 #BiocManager::install(version = "3.10")
28 BiocManager::install(c("dada2", "phyloseq", "Biostrings"))
29
30 install.packages("devtools")
31 devtools::install_github("pr2database/pr2database") # Installs directly from github resources that are not in R repos
32 devtools::install_github("GuillemSalazar/EcolUtils") # Installs other tools for ecological analyses
33
34 #Load libraries
35 ##### Load libraries #####
36
37 library("dada2")
38 library("phyloseq")
39 library("Biostrings")
40 library("ggplot2")
41 library("dplyr")
42 library("tidyverse")
43 library("tibble")
44 library("readxl")
45 library("readr")
46 library("stringr")
47 library("kableExtra")
48 library("tidyverse")
49 #library("pr2database")
50
```



- Reproduce all steps shown in this presentation
- The code is available in:
https://github.com/krabberod/BIO9905MERG1_V21/blob/main/intro.to.r/Intro.to.R.BIO9905MERG1_V21.R