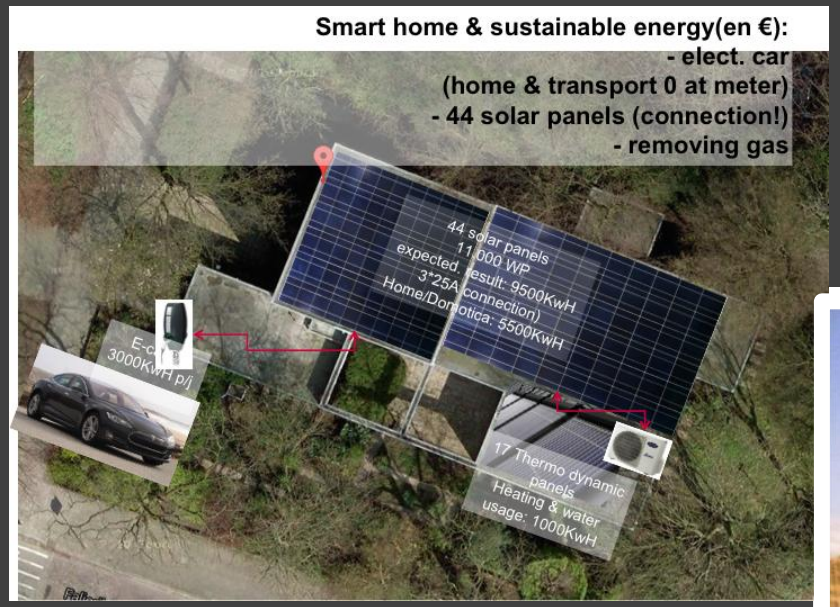
Dr.lec. Barry Derksen MSc MMC CISA CGEIT



antwerp management school
Powered by the University of Antwerp

VU

NCOI Onderzoeksinstituut

NTI

inc I'm in control

STEDIN.NET

KPMG

Speakers Academy
Bringing Knowledge to the World

ISACA
Vertrouwen in en waarde uit informatiesystemen

ICS VISA MasterCard
INTERNATIONAL CARD SERVICES

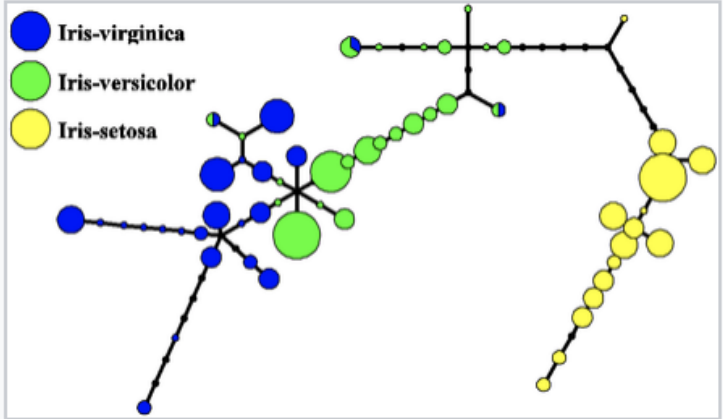
IRIS Case in R

- IRIS case is een echte doceer case voor R gebruik
- De dataset komt uit 1936 (The use of multiple measurements in taxonomic problems)
- 3 plantsoorten: Setosa, Virginica, Versicolor
- 4 meetpunten per plantsoort…in centimeters

IRIS case

- Species: Versicolor, Setosa, Virginica
- Variabelen: sepal.length , sepal.width , petal.length & petal.width
- Dataset: 50 voorbeelden van species
- Mogelijkheden: Linear discriminant model (species). Classificatie, clustering en algorithms.

Iris Versicolor **Iris Setosa** **Iris Virginica**



An example of the so-called "metro map" for the *Iris* data set.[4] Only a small fraction of *Iris-virginica* is mixed with *Iris-versicolor*. All other samples of the different *Iris* species belong to the different nodes.

Intro

- RStudio allows the user to run R in a more user-friendly environment. It is open-source (i.e. free) and available at http://www.rstudio.com/
- For R related tutorials and/or resources see the following links:
- http://dss.princeton.edu/training/ http://libguides.princeton.edu/dss

Over R

> 10.000 add-on packages

>100.000 LinkIn R group

File Edit Code View Plots Session Project Build Tools Help

Go to file/function

Project: (None)

Console H:/MyData/RFiles/

```
R version 3.0.0 (2013-04-03) -- "Masked Marvel"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> getwd()
[1] "H:/MyData/RFiles"
> 5*5
[1] 25
> A <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2)
> A
     [,1] [,2]
[1,]    1    5
[2,]    2    6
[3,]    3    7
[4,]    4    8
> B <- matrix(c(1,2,3,4,5,6,7,8), nrow=4, ncol=2, byrow=TRUE)
> B
     [,1] [,2]
[1,]    1    2
[2,]    3    4
[3,]    5    6
[4,]    7    8
>
```

The **console** is where you can type commands and see output

Workspace History

Import Dataset

Data	
A	4x2 double matrix
B	4x2 double matrix

The **workspace** tab shows all the active objects (see next slide). The **history** tab shows a list of commands used so far.

Files Plots Packages Help

New Folder Delete Rename More

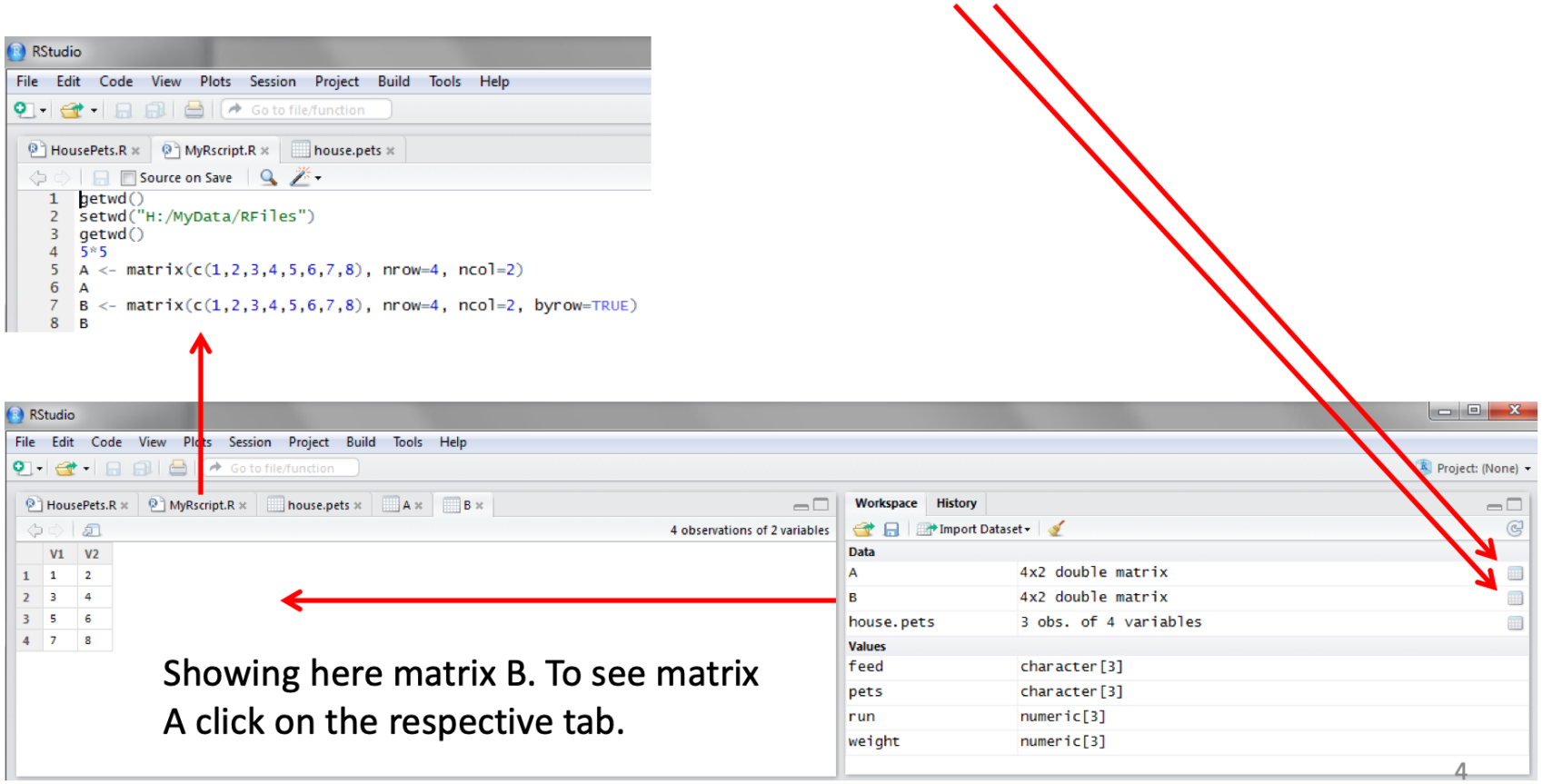
H: MyData RFiles

Name	Size	Modified
..		
.Rhistory	34 bytes	Aug 23, 2013, 1:26 PM

The **files** tab shows all the files and folders in your default workspace as if you were on a PC/Mac window. The **plots** tab will show all your graphs. The **packages** tab will list a series of packages or add-ons needed to run certain processes. For additional info see the **help** tab

Workspace tab (1)

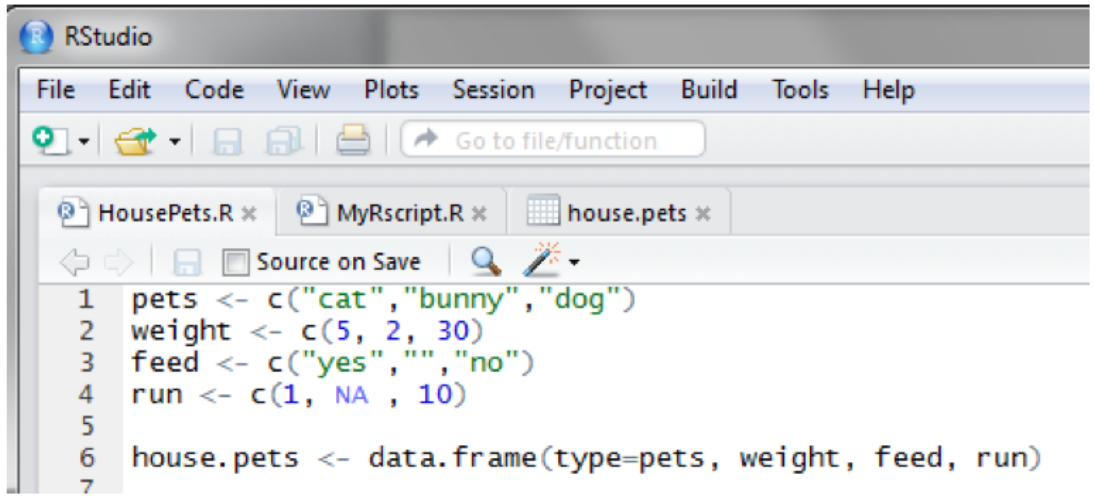
The workspace tab stores any object, value, function or anything you create during your R session. In the example below, if you click on the dotted squares you can see the data on a screen to the left.



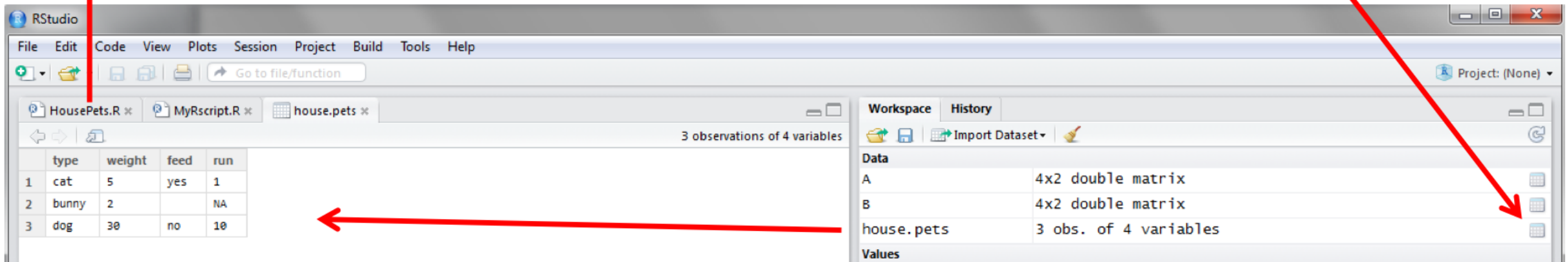
Showing here matrix B. To see matrix A click on the respective tab.

Workspace tab (2)

Here is another example on how the workspace looks like when more objects are added. Notice that the data frame `house.pets` is formed from different individual values or vectors.

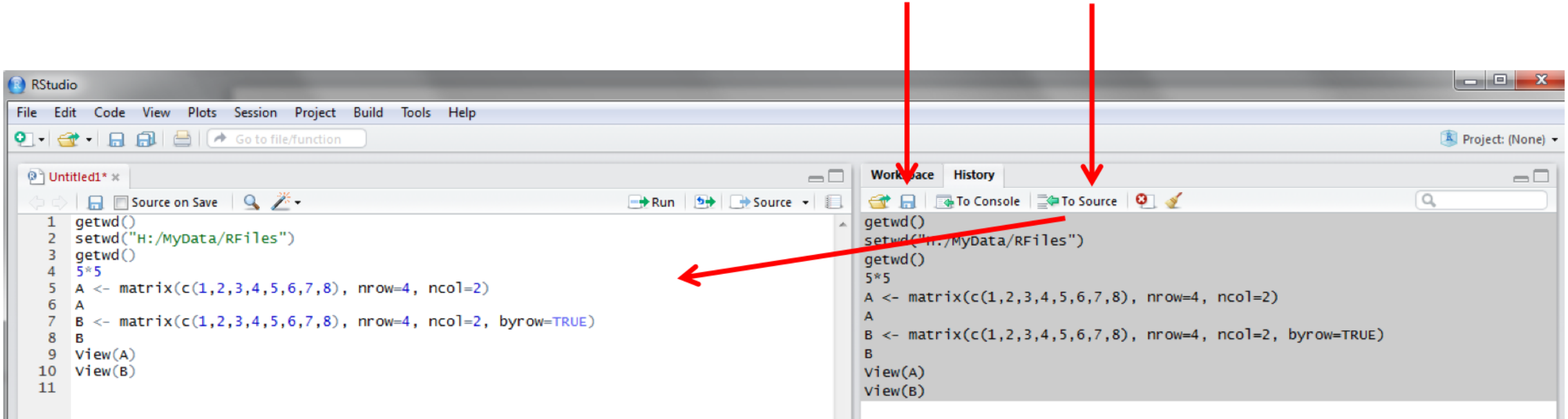


Click on the dotted square to look at the dataset in a spreadsheet form.

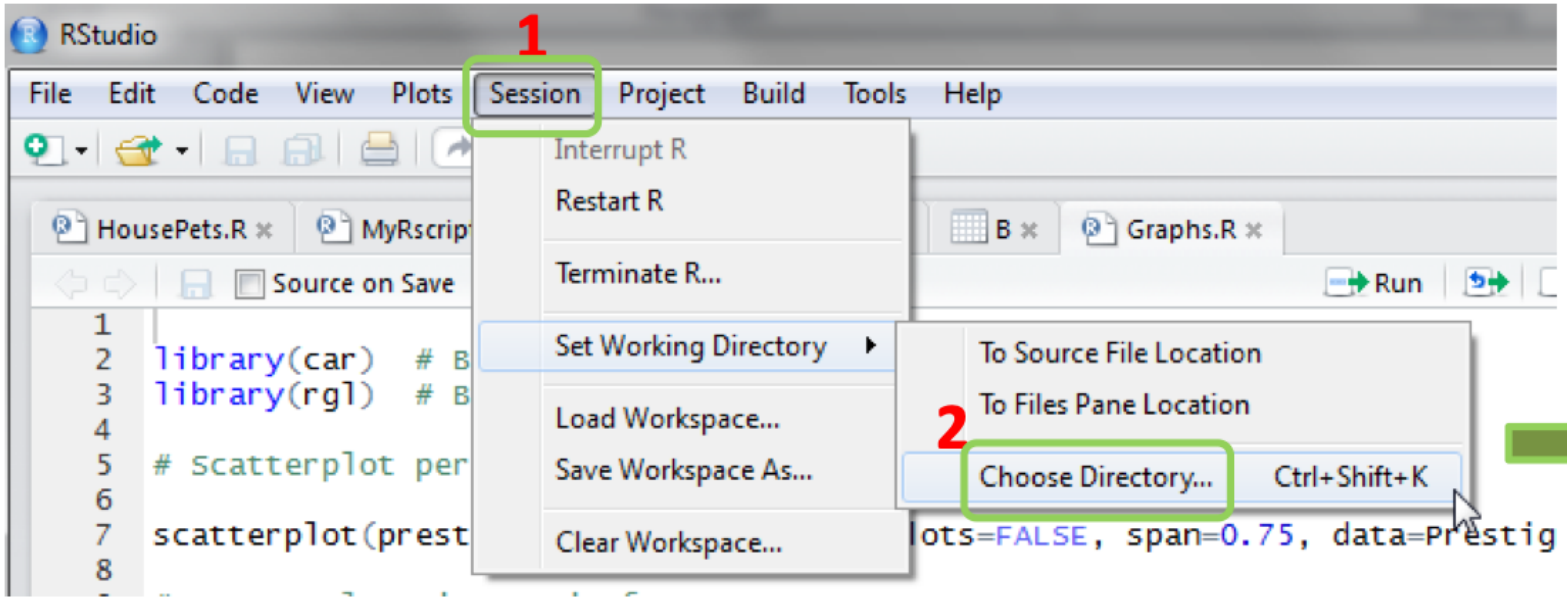


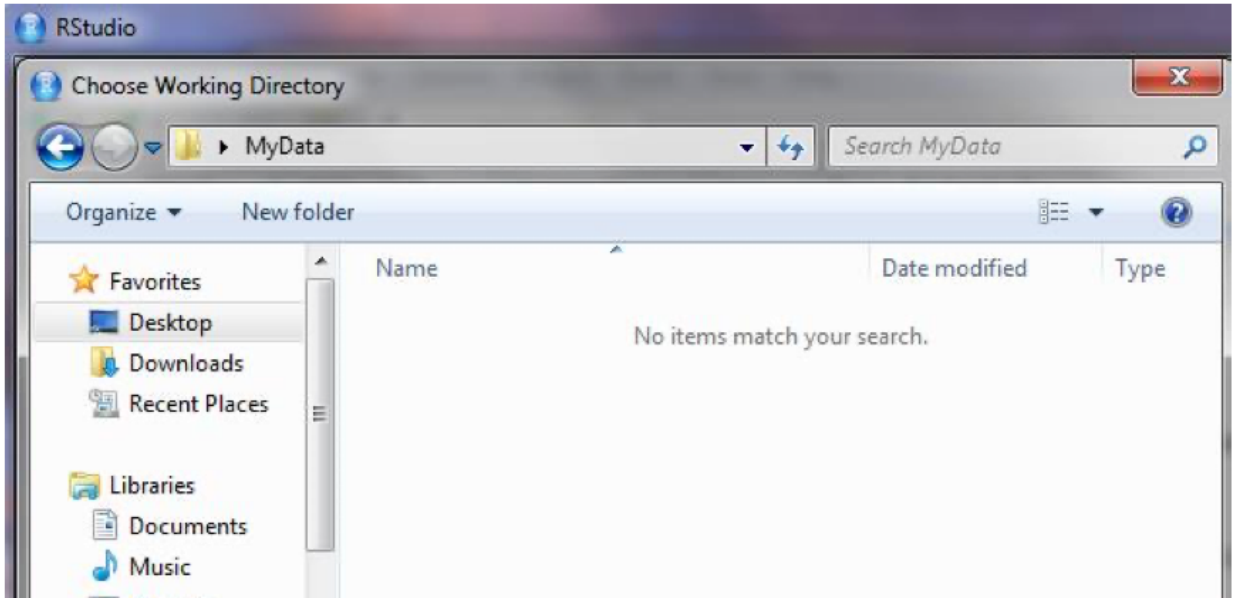
History tab

The history tab keeps a record of all previous commands. It helps when testing and running processes. Here you can either **save** the whole list or you can **select** the commands you want and send them to an R script to keep track of your work.
In this example, we select all and click on the "`To Source`" icon, a window on the left will open with the list of commands. Make sure to save the 'untitled1' file as an *.R script.



Changing the working directory



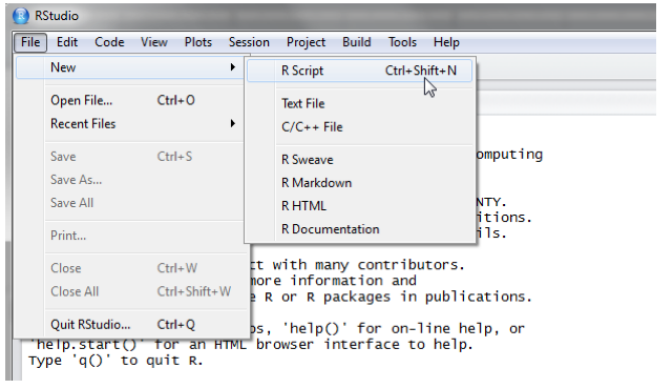


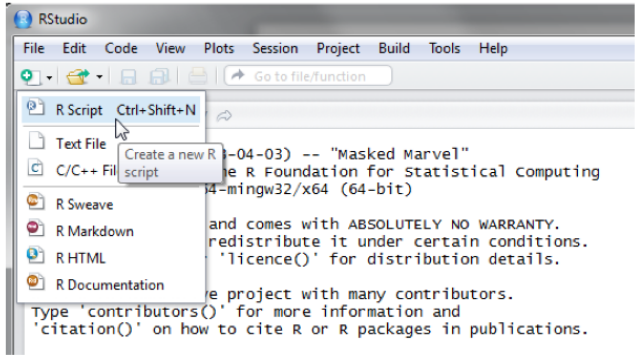
If you have different projects you can change the working directory for that session, see above. Or you can type:

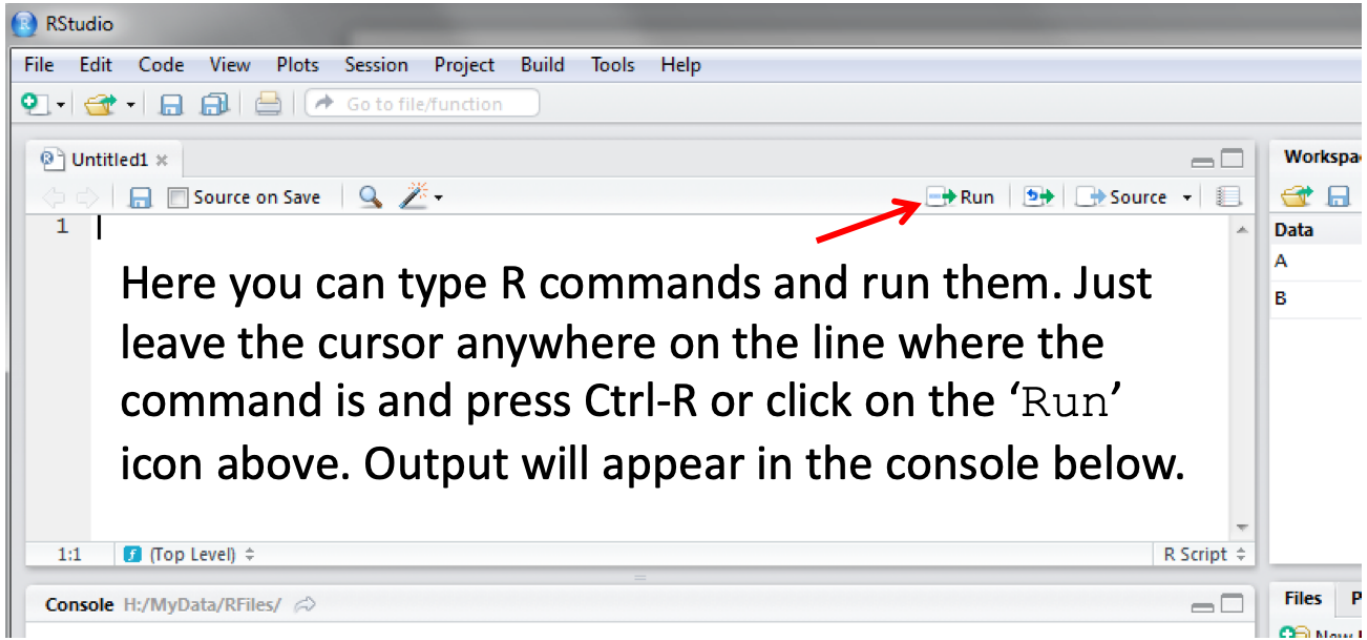
```
# Shows the working directory (wd)

getwd()

# Changes the wd
```

R script (2)

To create a new R script you can either go to `File -> New -> R Script`, or click on the icon with the "+" sign and select "R Script", or simply press Ctrl+Shift+N. Make sure to save the script.







R install

- Download R en voor de installatie uit voor het gewenste OS :
- https://lib.ugent.be/CRAN/
- Download RStudio for Desktop en voor de installatie uit:
- https://www.rstudio.com/products/rstudio/#Desktop

Scripts uitvoeren (dit zijn functionaliteiten)

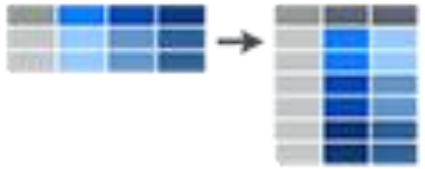
```
install.packages("caret", repos = 'https://lib.ugent.be/CRAN/')
install.packages("tidyr", repos = 'https://lib.ugent.be/CRAN/')
install.packages("ggthemes", repos = 'https://lib.ugent.be/CRAN/')
install.packages("MASS", repos = 'https://lib.ugent.be/CRAN/')
install.packages("e1071", repos = 'https://lib.ugent.be/CRAN/')
install.packages("randomForest", repos = 'https://lib.ugent.be/CRAN/')
install.packages("gbm", repos = 'https://lib.ugent.be/CRAN/')
install.packages("lda", repos = 'https://lib.ugent.be/CRAN/')
```

Voorbeeld: Caret Package

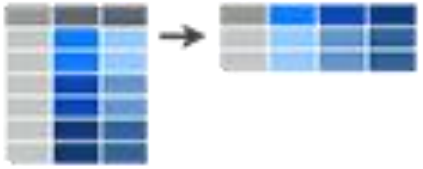
- The caret package (short for _C_lassification _A_nd _RE_gression _T_raining) is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for:
- data splitting
- pre-processing
- feature selection
- model tuning using resampling
- variable importance estimation
- as well as other functionality

Organize Your Data for Easier Analyses in R

gather()



spread()



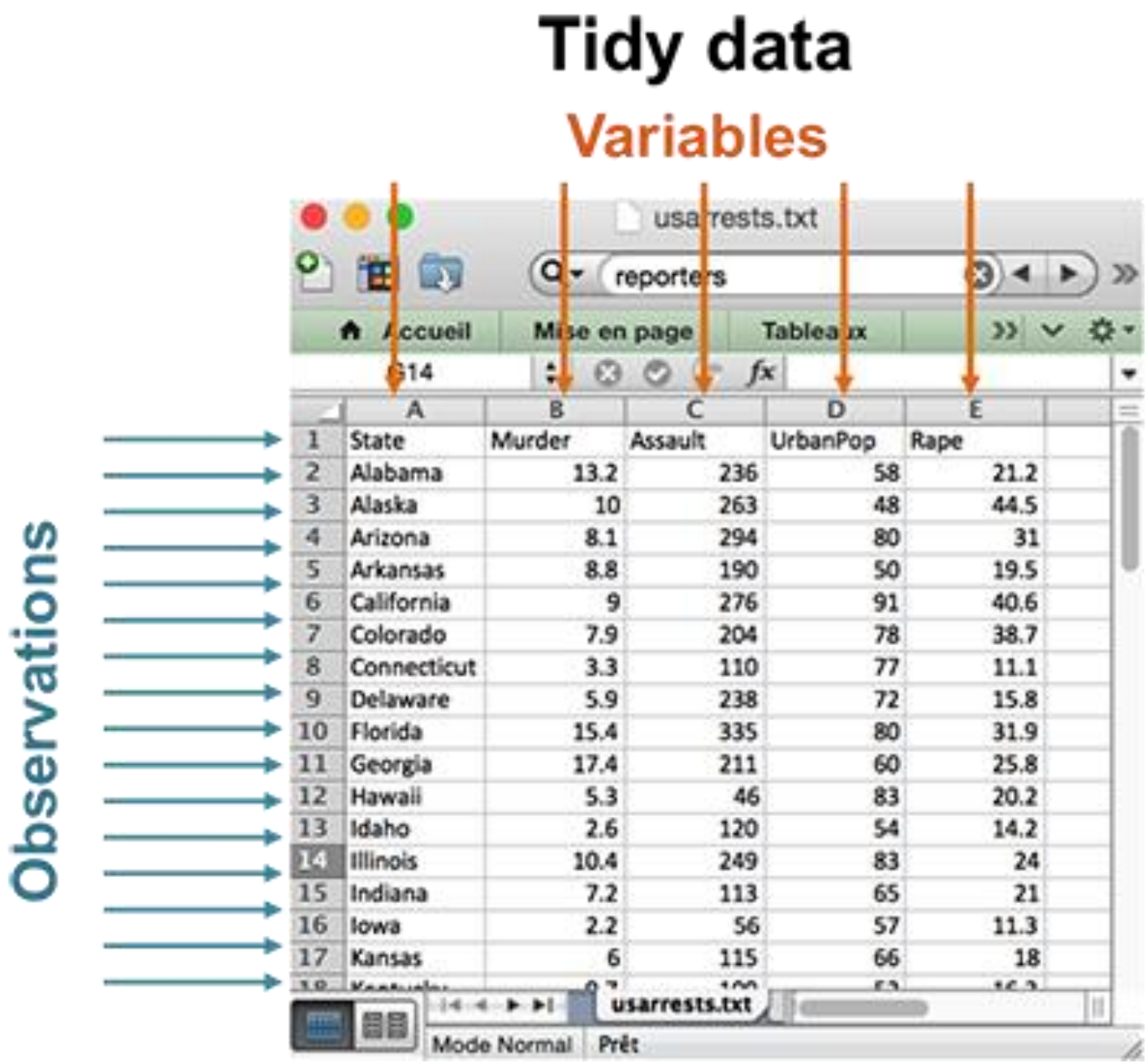
separate()

unite()

- **gather()**: collapse multiple columns into key-pair values
- **spread()**: reverse of gather. Separate one column into multiple
- **separate()**: separate one column into multiple
- **unite()**: unite multiple columns into one

tidyr R package

Voorbeeld: Tidyr package



	A	B	C	D	E
1	State	Murder	Assault	UrbanPop	Rape
2	Alabama	13.2	236	58	21.2
3	Alaska	10	263	48	44.5
4	Arizona	8.1	294	80	31
5	Arkansas	8.8	190	50	19.5
6	California	9	276	91	40.6
7	Colorado	7.9	204	78	38.7
8	Connecticut	3.3	110	77	11.1
9	Delaware	5.9	238	72	15.8
10	Florida	15.4	335	80	31.9
11	Georgia	17.4	211	60	25.8
12	Hawaii	5.3	46	83	20.2
13	Idaho	2.6	120	54	14.2
14	Illinois	10.4	249	83	24
15	Indiana	7.2	113	65	21
16	Iowa	2.2	56	57	11.3
17	Kansas	6	115	66	18

Test datasets

```
library(caret)
library(tidyr)
library(ggthemes)
library(MASS)
library(e1071)
library(randomForest)
library(gbm)
library(lda)
```

Laten we naar de data kijken (5 rijen)

```
# Verkrijgen eerste 5 rijen van elke subset
```

```
subset(iris, Species == "setosa")[1:5,]
CHECK result
```

```
subset(iris, Species == "versicolor")[1:5,]
CHECK result
```

```
subset(iris, Species == "virginica")[1:5,]
```

```
check result
```

Analyseer de 3 keer 5 rijen

- Waar zie je al onderscheid?
- Schrijf je eerste bevindingen op om de 3 soorten te onderscheiden

Exploratief data analyse

Snel is te zien dat petal.length van sort Setosa korter is dan petal.length van andere soorten. Is dit waar?

 # Get column "Species" for all lines where Petal.Length < 2

subset(iris, Petal.Length < 2)[,"Species"]

Je hebt nu een eerste selectie dat een deel van de data verklaart

lets meer van de data leren

SUMMARY(IRIS)

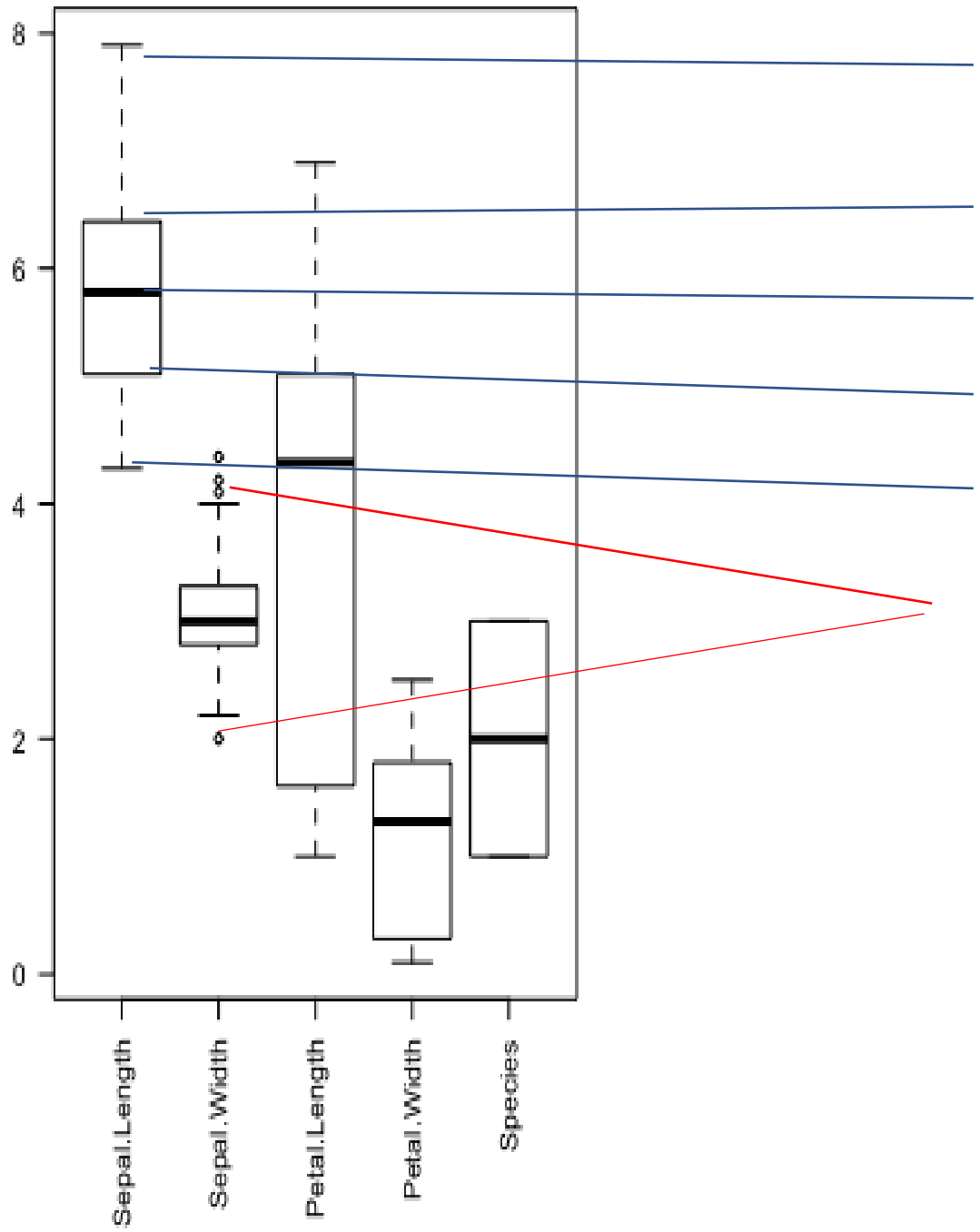
WAT LEES JE HIER?

summary(iris)

```
Sepal.Length    Sepal.Width
Min.   :4.300   Min.   :2.000
1st Qu.:5.100   1st Qu.:2.800
Median :5.800   Median :3.000
Mean   :5.843   Mean   :3.057
3rd Qu.:6.400   3rd Qu.:3.300
Max.   :7.900   Max.   :4.400
Petal.Length    Petal.Width
Min.   :1.000   Min.   :0.100
1st Qu.:1.600   1st Qu.:0.300
Median :4.350   Median :1.300
Mean   :3.758   Mean   :1.199
3rd Qu.:5.100   3rd Qu.:1.800
Max.   :6.900   Max.   :2.500
Species
setosa    :50
versicolor:50
virginica :50
```

Visualeren?
→ boxplot

```
par(mar=c(7,5,1,1))
# more space to labels
boxplot(iris,las=2)
```



Maximum value (excl. outliers

Upper Quartile: 25% of values are higher than this

Median: 50% of values are higher / 50% lower

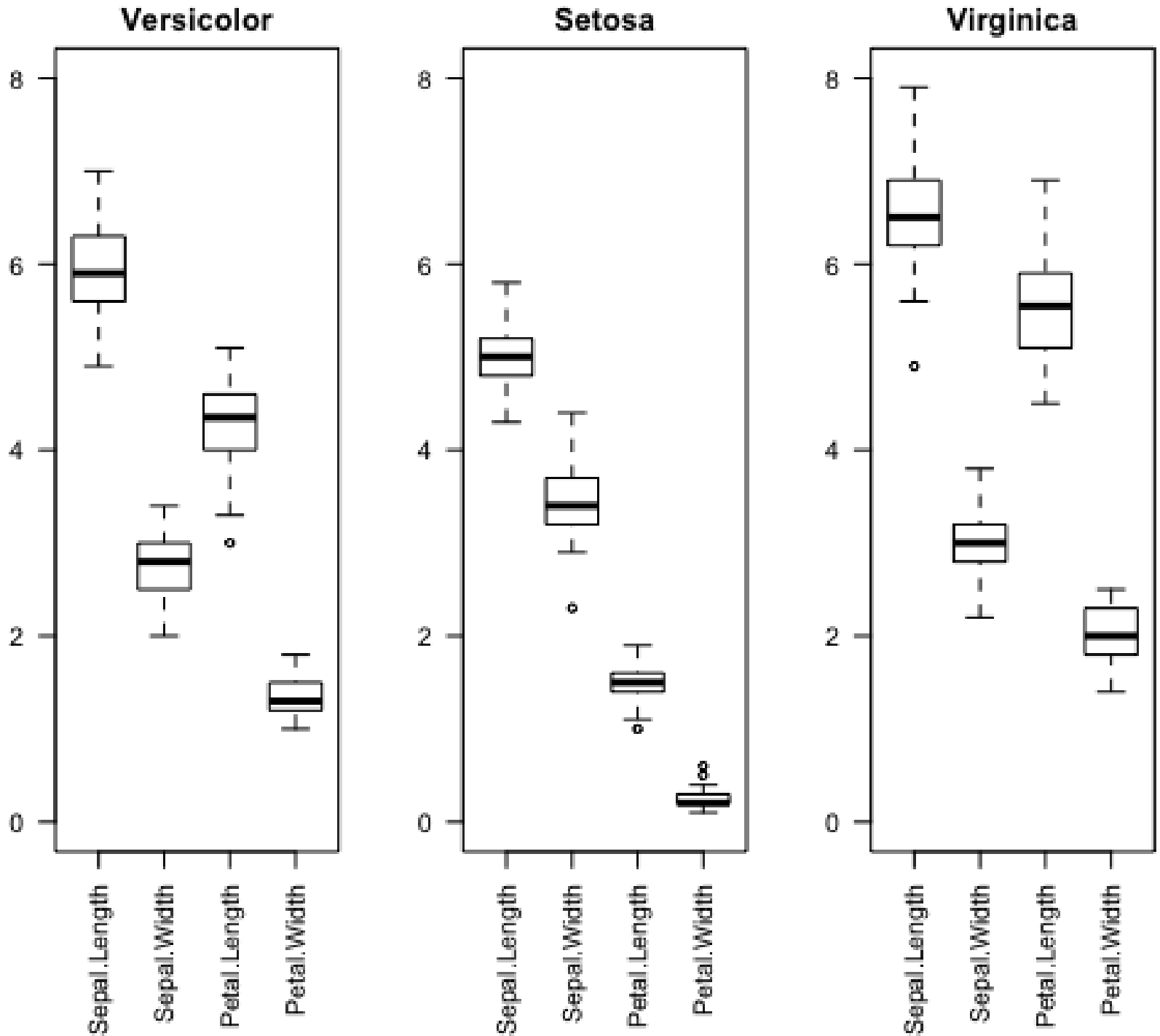
Lower Quartile: 25% of values are Lower than this

Lower Quartile: 25% of values are lower than this

Outliers: values above or below 1,5 times the interquartile range

Een scherper beeld van elke soort verkrijgen

```
irisVer <- subset(iris, Species == "versicolor")
irisSet <- subset(iris, Species == "setosa")
irisVir <- subset(iris, Species == "virginica")
par(mfrow=c(1,3),mar=c(6,3,2,1))
boxplot(irisVer[,1:4], main="Versicolor",ylim = c(0,8),las=2)
boxplot(irisSet[,1:4], main="Setosa",ylim = c(0,8),las=2)
boxplot(irisVir[,1:4], main="Virginica",ylim = c(0,8),las=2)
```


Versicolor
Setosa
Virginica
0
2
4
6
8
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width

Histogram (te calculeren per attribute)

```
hist(iris$Petal.Length)
```

```
#histogram voor 1 attribute maar per soort
```

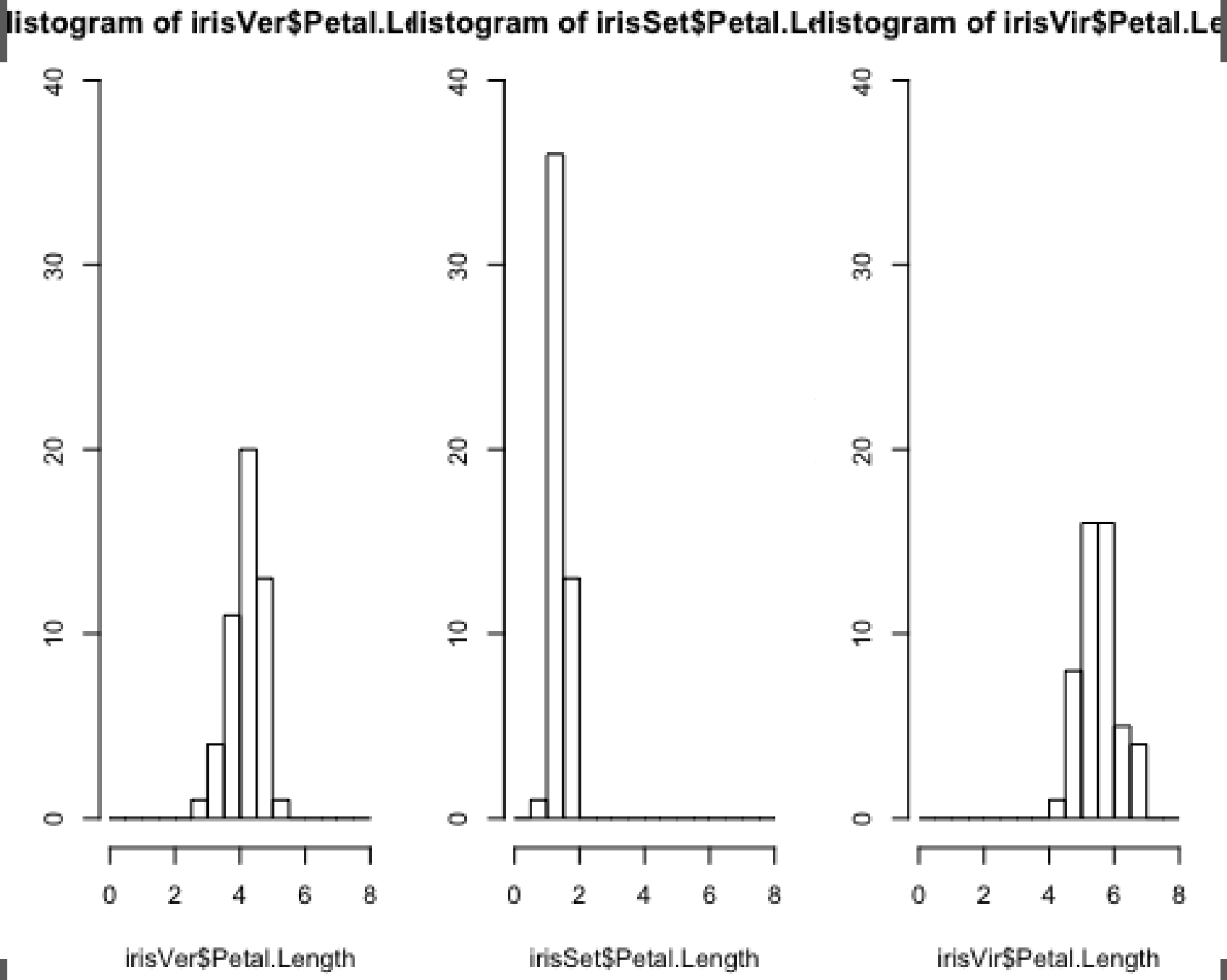
```
par(mfrow=c(1,3))
```

```
hist(irisVer$Petal.Length,breaks=seq(0,8,l=17),xlim=c(0,8),ylim=c(0,40))
```

```
hist(irisSet$Petal.Length,breaks=seq(0,8,l=17),xlim=c(0,8),ylim=c(0,40))
```

```
hist(irisVir$Petal.Length,breaks=seq(0,8,l=17),xlim=c(0,8),ylim=c(0,40))
```

Je ziet de distributie van de waarde van petal.length verschillend zijn per soort


Histogram of irisVer$Petal.L
Histogram of irisSet$Petal.L
Histogram of irisVir$Petal.Le
0
10
20
30
40
0
2
4
6
8
irisVer$Petal.Length
irisSet$Petal.Length
irisVir$Petal.Length

Violin plots tonen statistiek en data distributie

- **library**(vioplot)

Als het goed zie je dit:

- ## Loading required package: sm
- ## Package 'sm', version 2.2-5.4: type help(sm) for summary information

Code:

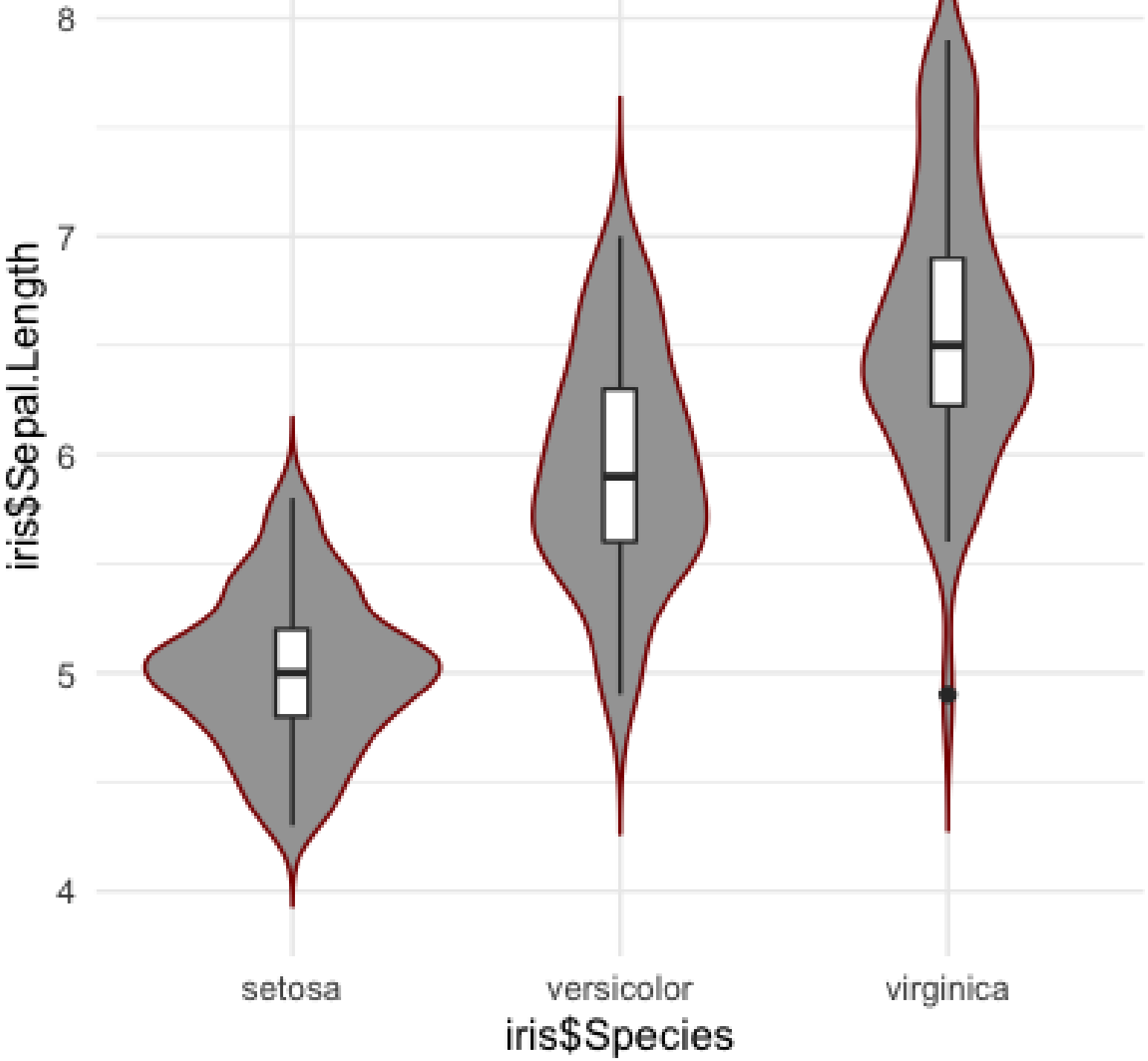
vioplot(iris$Sepal.Length,iris$Sepal.Width,iris$Petal.Length,iris$Petal.Width,
names=**c**("Sep.Len","Sep.Wid","Pet.Len","Pet.Wid"),
col=”green")

MAAR…..HET KAN ZOMAAR MIS GAAN EN NU?

ZELF PROBEREN UIT TE VOGELEN, HINT:

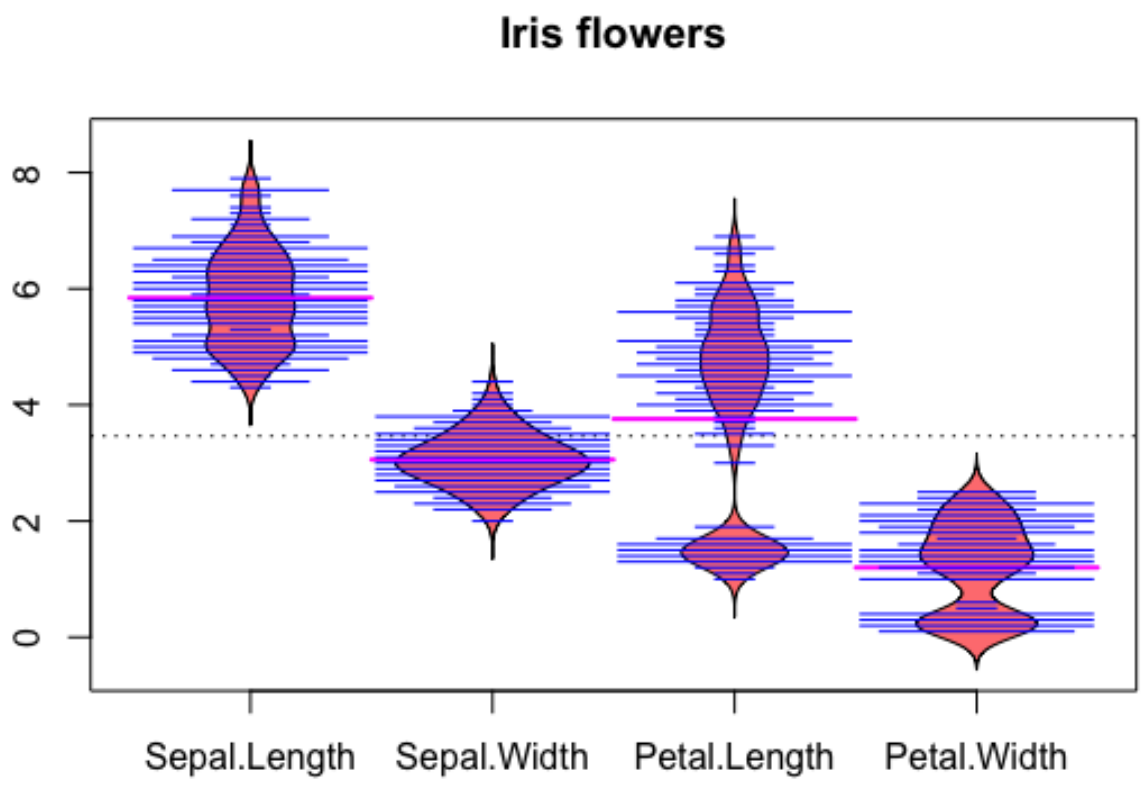
- http://www.sthda.com/english/wiki/ggplot2-violin-plot-quick-start-guide-r-software-and-data-visualization

- Hoe ver kom jij in 30 minuten?



Of:

```
> library(beanplot)
> xiris <- iris
> xiris$Species <- NULL
> beanplot(xiris, main = "Iris flowers",col=c('#ff8080','#0000FF','#0000FF','#FF00FF'), border = "#000000")
```



Correlaties tussen variabelen

```
corr <- cor(iris[,1:4])
```

```
round(corr,3)
```

Hoe lezen?

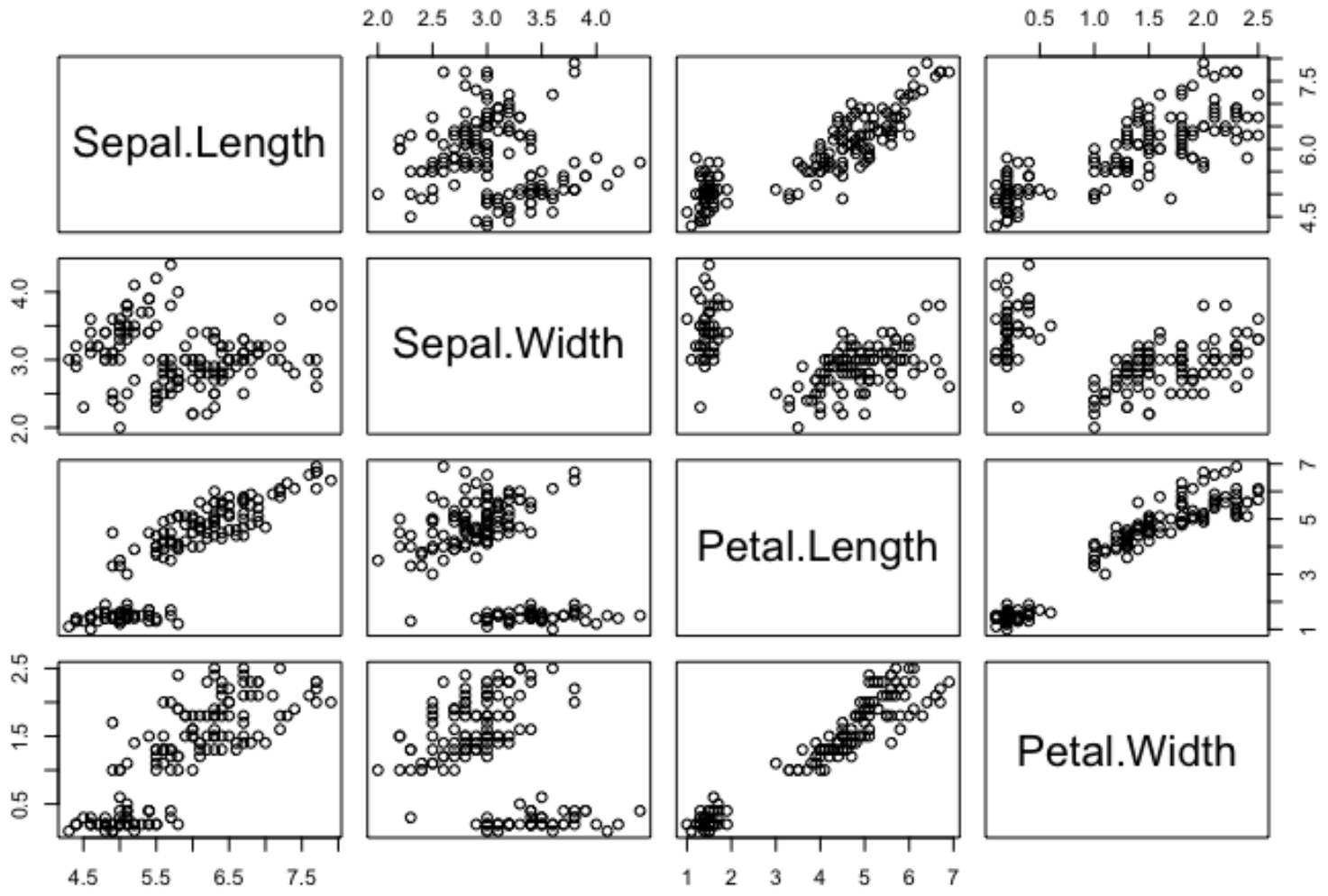
Correlatie tussen variabelen

Variabelen correleren volledig

- Sepal.Length Sepal.Width
- Sepal.Length 1.000 -0.118
- Sepal.Width -0.118 1.000
- Petal.Length 0.872 -0.428
- Petal.Width 0.818 -0.366
- Petal.Length Petal.Width
- Sepal.Length 0.872 0.818
- Sepal.Width -0.428 -0.366
- Petal.Length 1.000 0.963
- Petal.Width 0.963 1.000

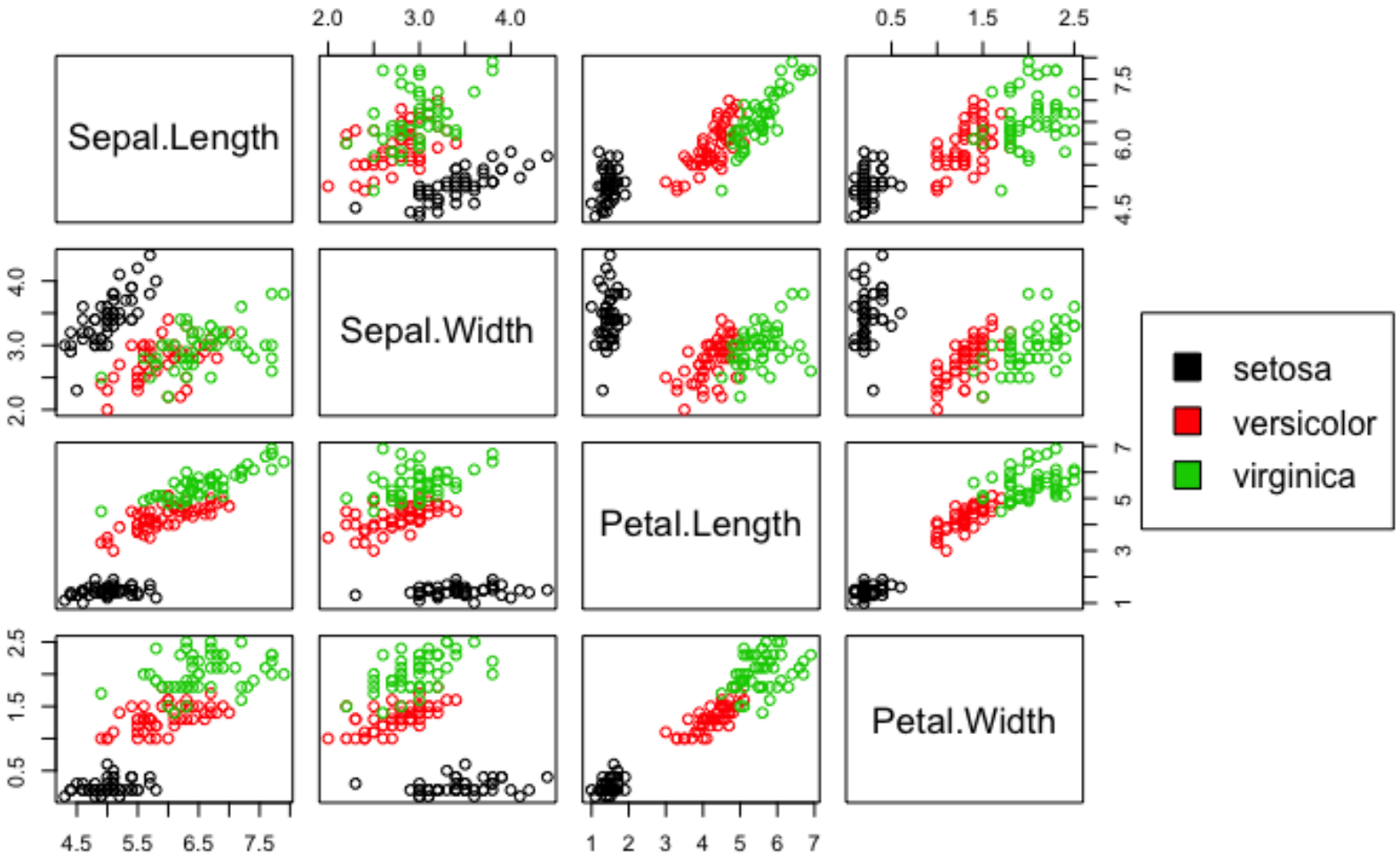
Scatterplot matrix

- **pairs**(iris[,1:4])
- Visuele bevestiging van vorige opdracht!

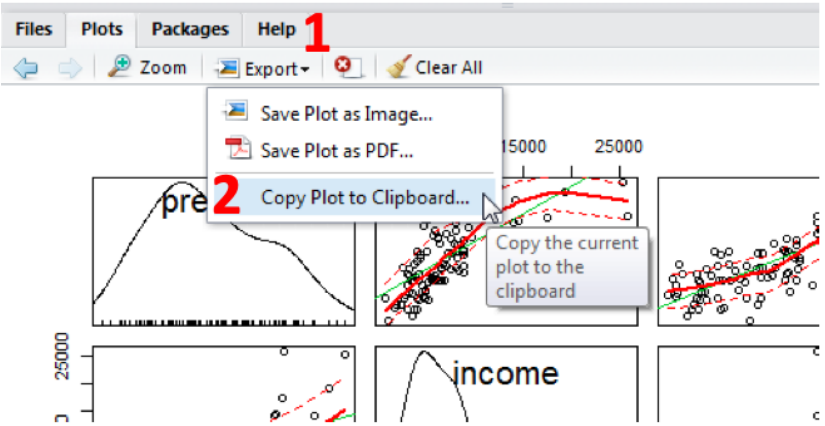


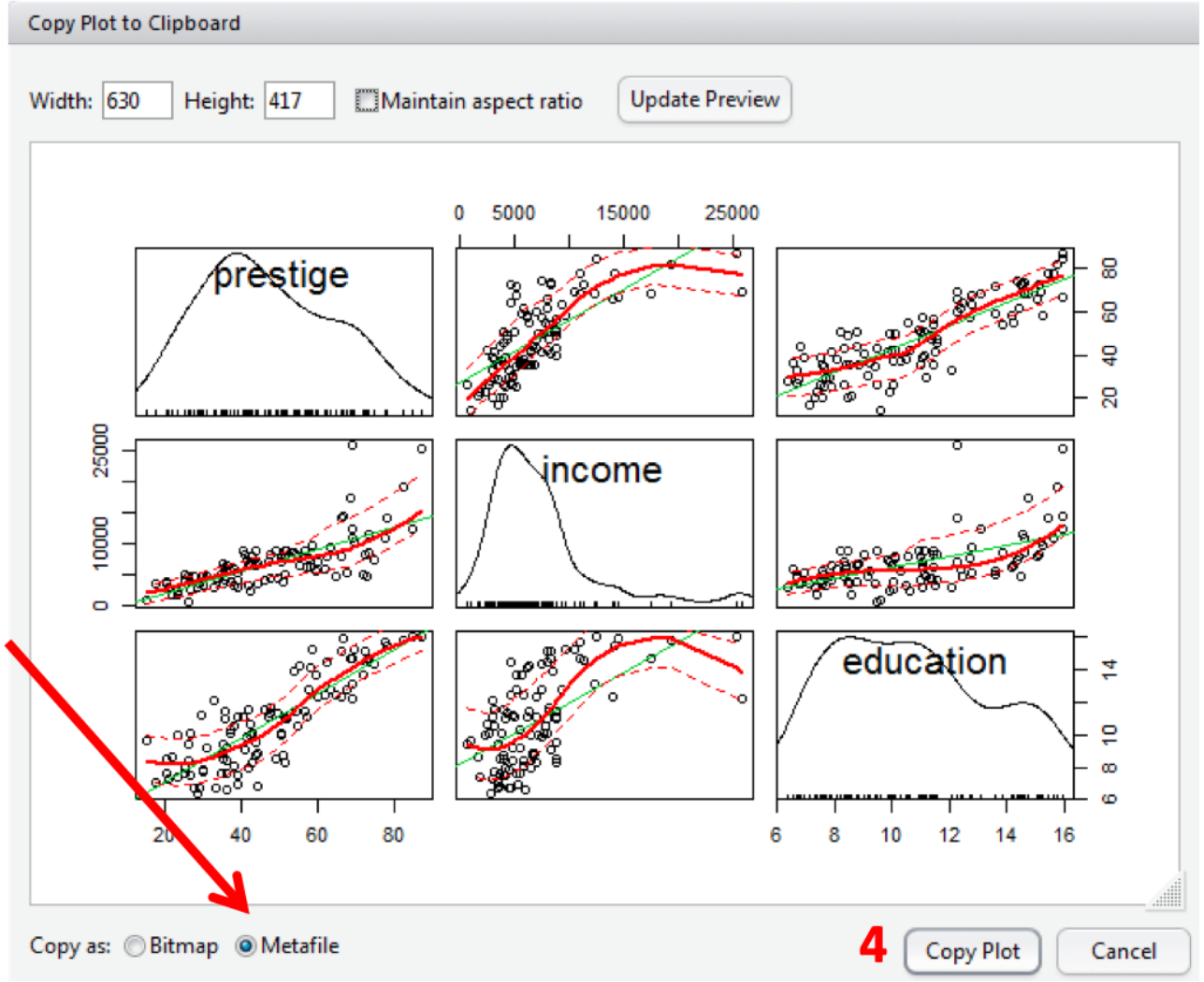
Dit kunnen we ook doen per soort

```
pairs(iris[,1:4],col=iris[,5],oma=c(4,4,6,12))
par(xpd=TRUE)
legend(0.85,0.6,
as.vector(unique(iris$Species)),fill=c(1,2,3))
```


Sepal.Length
Sepal.Width
Petal.Length
Petal.Width
setosa
versicolor
virginica
2.0
3.0
4.0
0.5
1.5
2.5
4.5
6.0
7.5
1
3
5
7
4.5
5.5
6.5
7.5

To extract the graph, click on "`Export`" where you can save the file as an image (PNG, JPG, etc.) or as PDF, these options are useful when you only want to share the graph or use it in a LaTeX document. Probably, the easiest way to export a graph is by copying it to the clipboard and then paste it directly into your Word document.





3 Make sure to select '`Metafile`'

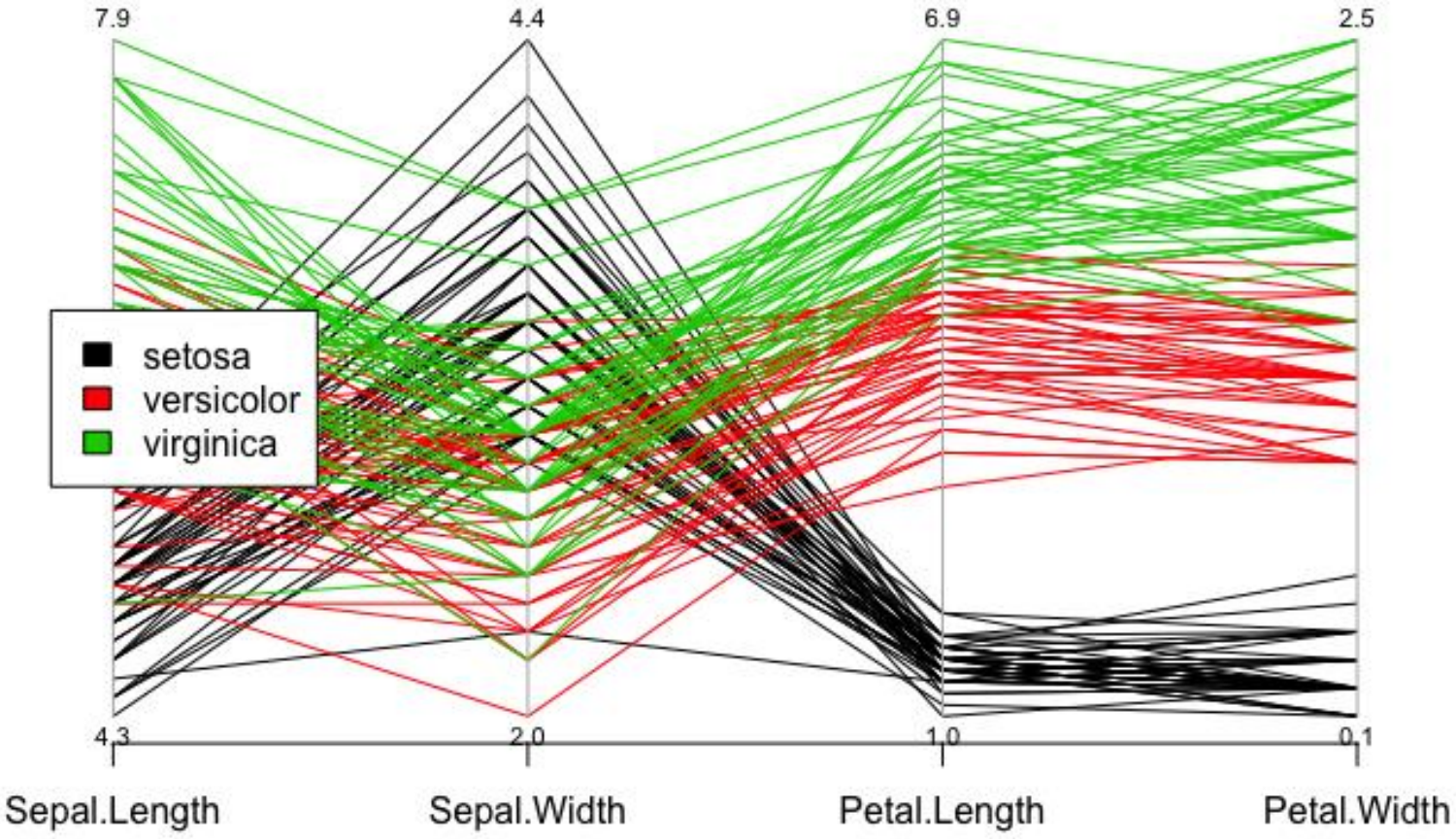
Een andere manier is parallell coordinate plot

```
library(MASS)
```

```
parcoord(iris[,1:4],
col=iris[,5],var.label=TRUE,oma=c(4,4,6,12))
```

```
par(xpd=TRUE)
```

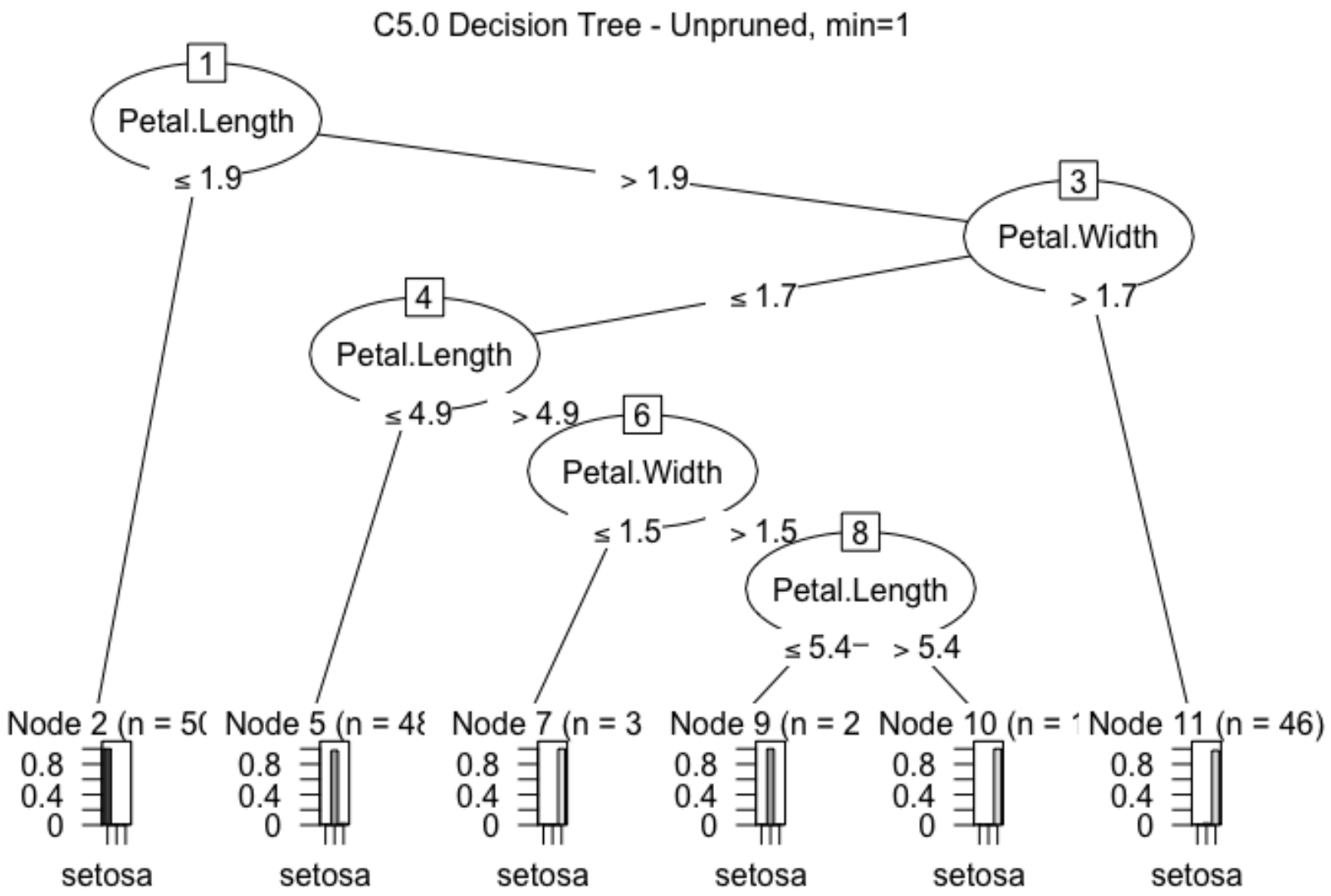
```
legend(0.85,0.6,
as.vector(unique(iris$Species)),fill=c(1,2,3))
```


7.9
4.4
6.9
2.5
setosa
versicolor
virginica
4.3
2.0
1.0
0.1
Sepal.Length
Sepal.Width
Petal.Length
Petal.Width

Laten we een besluitboom maken (classificatie)

- We weten dat er classes voor 150 instances van ‘Irises’. Interessant is of er een predictive model is voor de soorten gebaseerd op petal en sepal width en length. Hiervoor maken ween besluitboom

```
library(C50)
input <- iris[,1:4]
output <- iris[,5]
model1 <- C5.0(input, output, control = C5.0Control(noGlobalPruning = TRUE,minCases=1))
plot(model1, main="C5.0 Decision Tree - Unpruned, min=1")
```


C5.0 Decision Tree - Unpruned, min=1
1
Petal.Length
≤ 1.9
> 1.9
3
Petal.Width
≤ 1.7
> 1.7
4
Petal.Length
≤ 4.9
> 4.9
6
Petal.Width
≤ 1.5
> 1.5
8
Petal.Length
≤ 5.4
> 5.4
Node 2 (n = 50
Node 5 (n = 48
Node 7 (n = 3
Node 9 (n = 2
Node 10 (n = 1
Node 11 (n = 46)
0.8
0.4
0
setosa

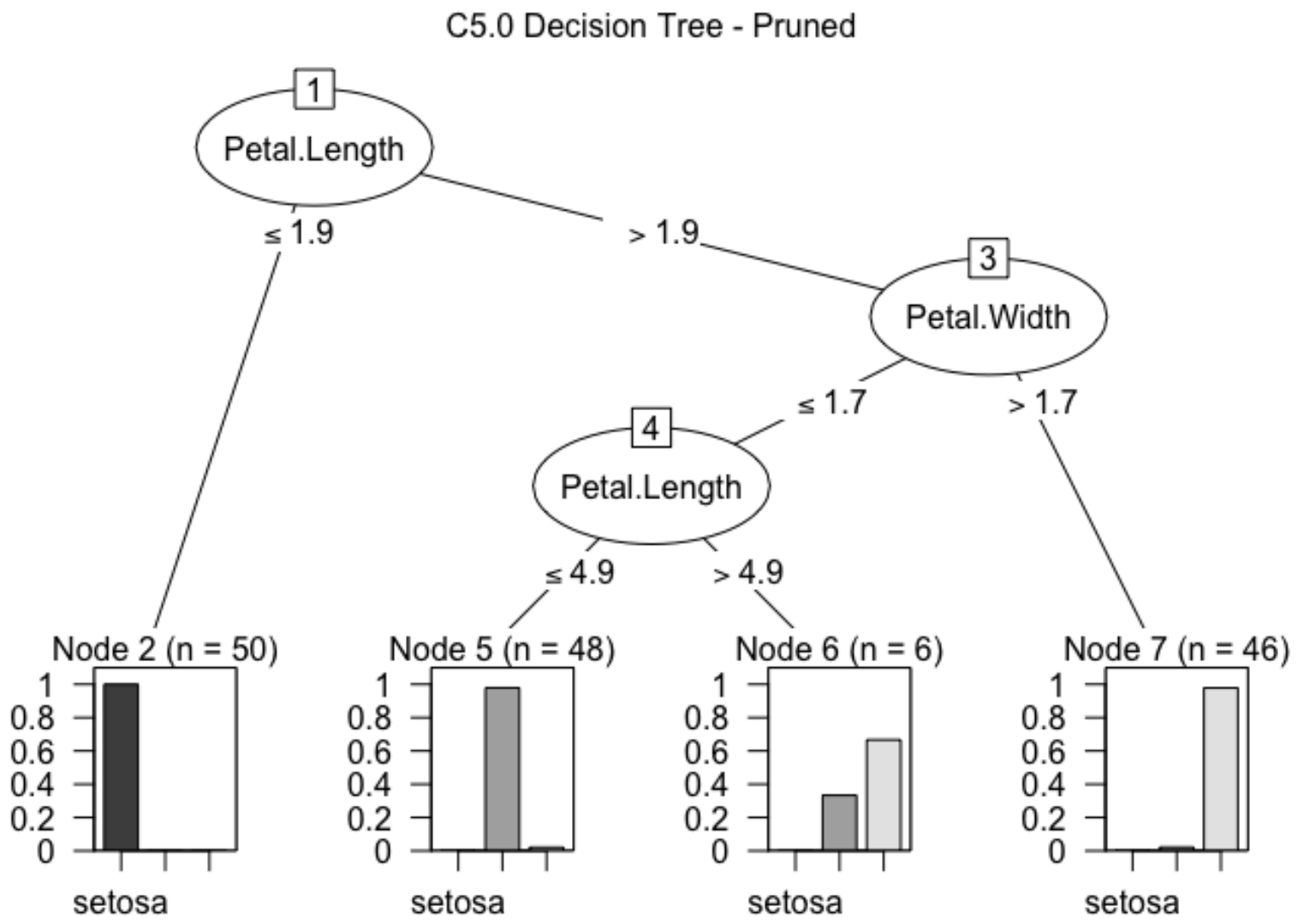
Eenvoudiger model maken

model2 <- **C5.0**(input, output, control = **C5.0Control**(noGlobalPruning = FALSE))

plot(model2, main="C5.0 Decision Tree - Pruned")

NA UITVOERING bovestaande:

summary(model2)


C5.0 Decision Tree - Pruned
1
Petal.Length
≤ 1.9
> 1.9
3
Petal.Width
≤ 1.7
> 1.7
4
Petal.Length
≤ 4.9
> 4.9
Node 2 (n = 50)
Node 5 (n = 48)
Node 6 (n = 6)
Node 7 (n = 46)
1
0.8
0.6
0.4
0.2
0
setosa

inzoomen

C5imp(model2,metric='usage’)

Predection gebaseerd op numerieke variabelen:

newcases <- iris[**c**(1:3,51:53,101:103),]

newcases

Voorspelling maken (bijv. Voor zonder species)

predicted <- **predict**(model2, newcases, type="class")

Predicted

Verrijken van je model:

predicted <- **predict**(model2, iris, type="class")

predicted

Vergelijk (als in database en voorspeld)

```
iris$predictedC501 <- predicted
```

```
iris[iris$Species != iris$predictedC501,]
```