DATA MINING USING R PROGRAMMING LANGUAGE

Mamdouh Alenezi

PRINCE MEGREN DATA MINING CENTER

مركز الأمير مقرن للاستكشاف المعرفي

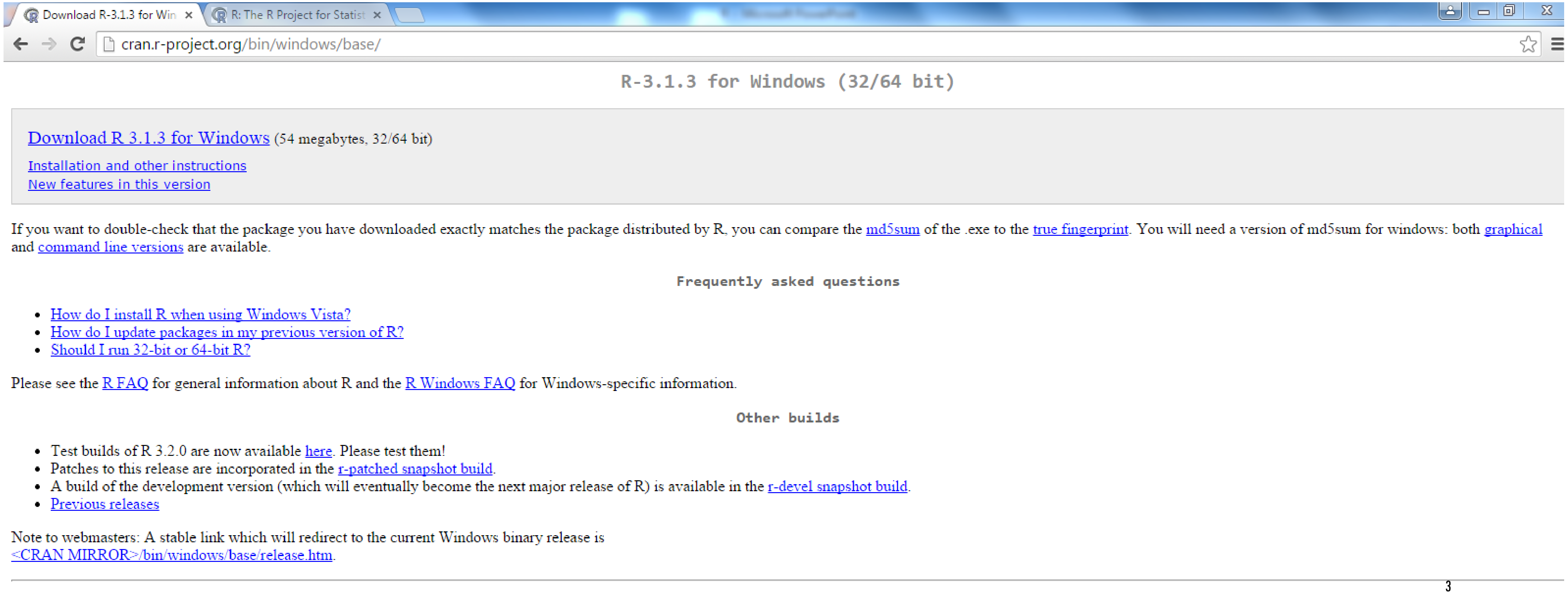
QUESTIONS

Have you heard of R?

Have you ever used R in your work?

Do you know data mining and its algorithms and techniques?

HTTP://WWW.R-PROJECT.ORG/ DOWNLOAD R


Download R-3.1.3 for Win
R: The R Project for Statist
cran.r-project.org/bin/windows/base/

R-3.1.3 for Windows (32/64 bit)

Download R 3.1.3 for Windows (54 megabytes, 32/64 bit)

Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded exactly matches the package distributed by R, you can compare the md5sum of the .exe to the true fingerprint. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- How do I install R when using Windows Vista?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

Other builds

- Test builds of R 3.2.0 are now available here. Please test them!
- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is
<CRAN MIRROR>/bin/windows/base/release.htm.

OUTLINE

WHAT IS R? AND WHY R?

WHAT IS R?

R is a free software environment for statistical computing and graphics.

R can be easily extended with around 6,000 packages available on CRAN3.

Many other packages provided on Bioconductor, R-Forge, GitHub, etc.

WHY DO DATA SCIENCE WITH R?

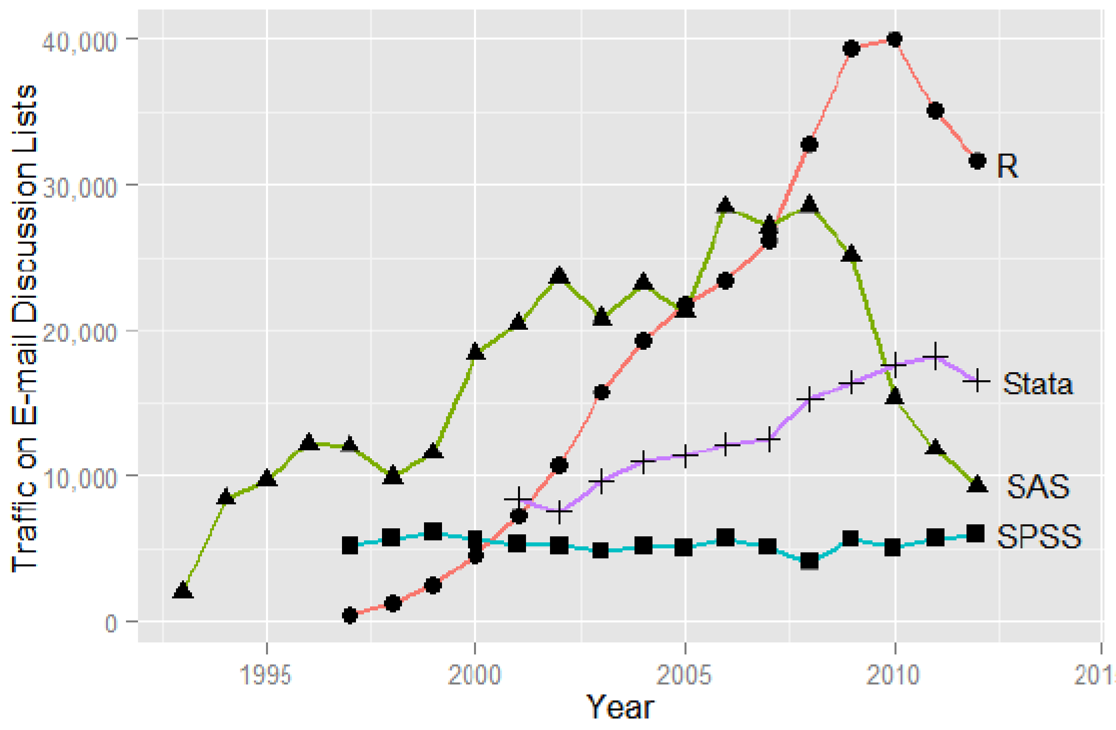
Most widely used Data Mining and Machine Learning Package

- Machine Learning
- Statistics
- Software Engineering and Programming with Data
- But not the nicest of languages for a Computer Scientist!

Free (Libre) Open Source Statistical Software

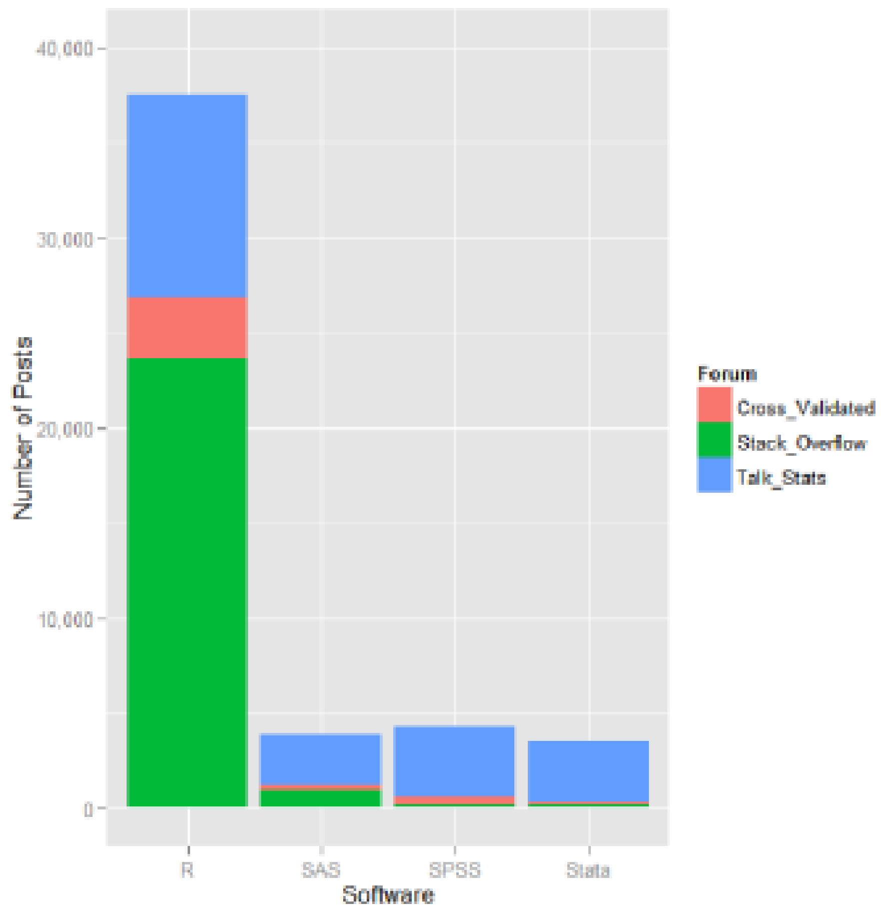
- All modern statistical approaches
- Many/most machine learning algorithms
- Opportunity to readily add new algorithms

HOW POPULAR IS R? DISCUSSION LIST TRAFFIC



Sum of monthly email traffic on each software’s main listserv discussion list.

HOW POPULAR IS R? DISCUSSION TOPICS



WHY R?

R was ranked no. 1 in the KDnuggets 2014 poll on Top Languages for analytics, data mining, data science (actually R has been no. 1 in 2011, 2012 & 2013!).

http://www.kdnuggets.com/polls/2014/languages-analytics-data-mining-data-science.html

DATA MINING IN R

DATA MINING

A data driven analysis to uncover otherwise unknown but useful patterns in large datasets, to discover new knowledge and to develop predictive models, turning data and information into knowledge and (one day perhaps) wisdom, in a timely manner.

DATA MINING

Application of

- Machine Learning
- Statistics
- Software Engineering and Programming with Data
- Effective Communications and Intuition
- …..

To Datasets that vary by:

- Volume, Velocity, Variety, Value, Veracity

To discover new knowledge

To improve business outcomes

To deliver better tailored services

BASIC TOOLS: DATA MINING ALGORITHMS

Cluster Analysis (kmeans, wskm)

Association Analysis (arules)

Linear Discriminant Analysis (lda)

Logistic Regression (glm)

Decision Trees (rpart, wsrpart)

Random Forests (randomForest, wsrf)

Boosted Stumps (ada)

Neural Networks (nnet)

Support Vector Machines (kernlab)

That’s a lot of tools to learn in R!

Many with different interfaces and options.

PREDICTIVE MODELLING: CLASSIFICATION

Goal of classification is to build models (sentences) in a knowledge representation (language) from examples of past decisions.

The model is to be used on unseen cases to make decisions.

Often referred to as supervised learning.

Common approaches: decision trees; neural networks; logistic regression; support vector machines.

MAJOR CLUSTERING APPROACHES

Partitioning algorithms (kmeans, pam, clara, fanny)

Hierarchical algorithms: (hclust, agnes, diana)

Density-based algorithms

Grid-based algorithms

Model-based algorithms: (mclust for mixture of Gaussians)

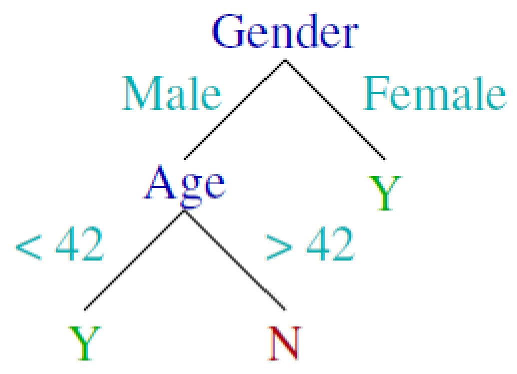
LANGUAGE: DECISION TREES

Knowledge representation: A flow-chart-like tree structure

Internal nodes denotes a test on a variable

Branch represents an outcome of the test

Leaf nodes represent class labels or class distribution



DATA SCIENTISTS ARE PROGRAMMERS OF DATA

Data Scientists Desire. . .

Scripting

Transparency

Repeatability

Sharing

SOCIAL NETWORK ANALYSIS WITH R

Packages: igraph, sna

Centrality measures: degree(), betweenness(), closeness(), transitivity()

Clusters: clusters(), no.clusters()

Cliques: cliques(), largest.cliques(), maximal.cliques(), clique.number()

Community detection: fastgreedy.community(), spinglass.community()

R AND BIG DATA

Hadoop

- Hadoop (or YARN) - a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models
- R Packages: RHadoop, RHIPE

Spark

- Spark - a fast and general engine for large-scale data processing, which can be 100 times faster than Hadoop
- SparkR - R frontend for Spark

H2O

- H2O - an open source in-memory prediction engine for big data science
- R Package: h2o

MongoDB

- MongoDB - an open-source document database
- R packages: rmongodb, RMongo

R AND HADOOP

Packages: RHadoop, Rhive

RHadoop is a collection of R packages:

- rmr2 - perform data analysis with R via MapReduce on a Hadoop cluster
- rhdfs - connect to Hadoop Distributed File System (HDFS)
- rhbase - connect to the NoSQL HBase database
- . . .

You can play with it on a single PC (in standalone or pseudo-distributed mode), and your code developed on that will be able to work on a cluster of PCs (in full-distributed mode)!

DATA IMPORT/EXPORT IN R

DATA IMPORT AND EXPORT

Read data from and write data to

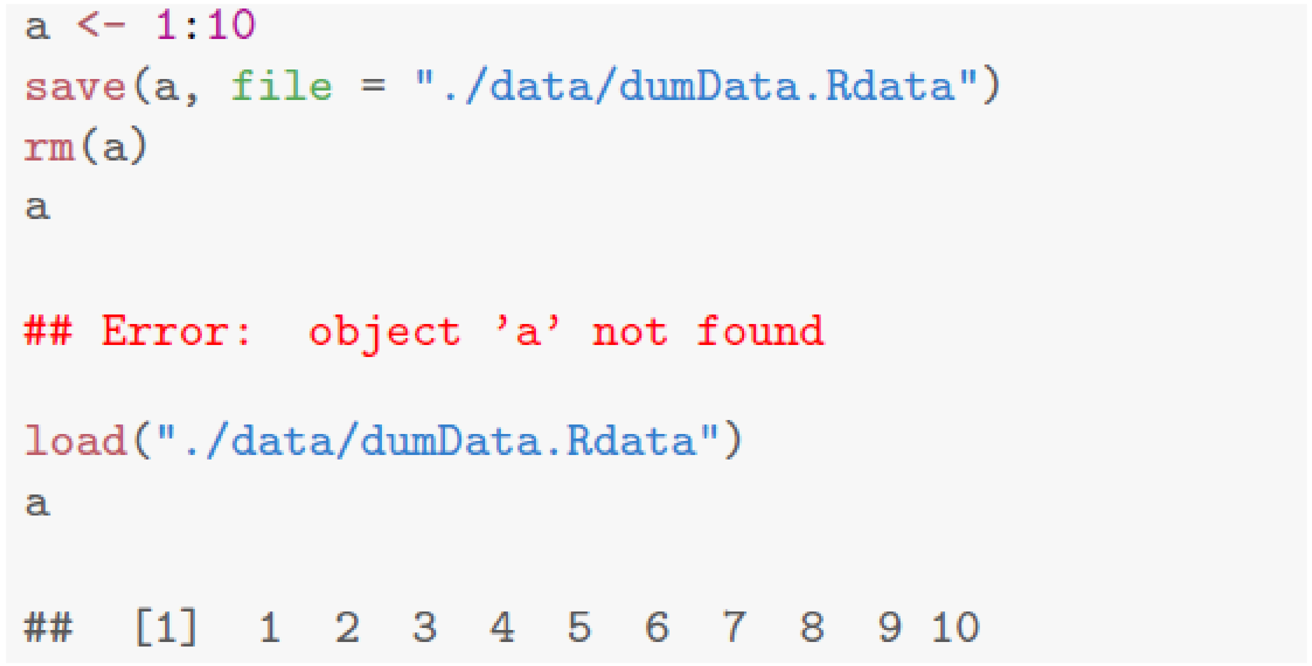
- R native formats (incl. Rdata and RDS)
- CSV files
- EXCEL files
- ODBC databases
- SAS databases

SAVE AND LOAD R OBJECTS

save(): save R objects into a .Rdata file

load(): read R objects from a .Rdata file

rm(): remove objects from R



```
a <- 1:10
save(a, file = "./data/dumData.Rdata")
rm(a)
a

## Error:  object ’a’ not found

load("./data/dumData.Rdata")
a

##  [1]  1  2  3  4  5  6  7  8  9 10
```

SAVE AND LOAD R OBJECTS - MORE FUNCTIONS

save.image():

- save current workspace to a file
- It saves everything!

readRDS():

- read a single R object from a .rds file

saveRDS():

- save a single R object to a file

Advantage of readRDS() and saveRDS():

- You can restore the data under a different object name.

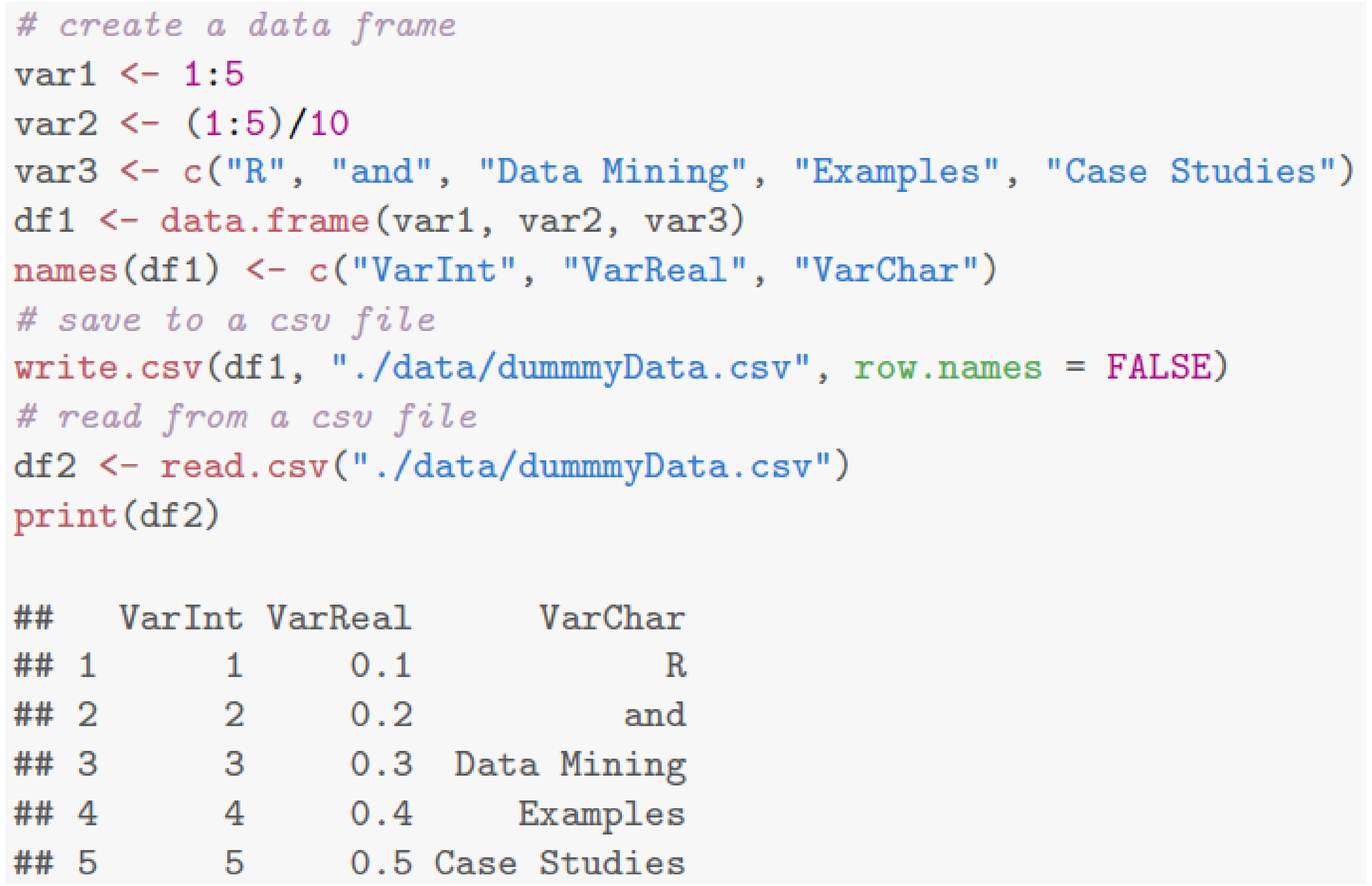
Advantage of load() and save():

- You can save multiple R objects to one file.

IMPORT FROM AND EXPORT TO .CSV FILES

write.csv(): write an R object to a .CSV file

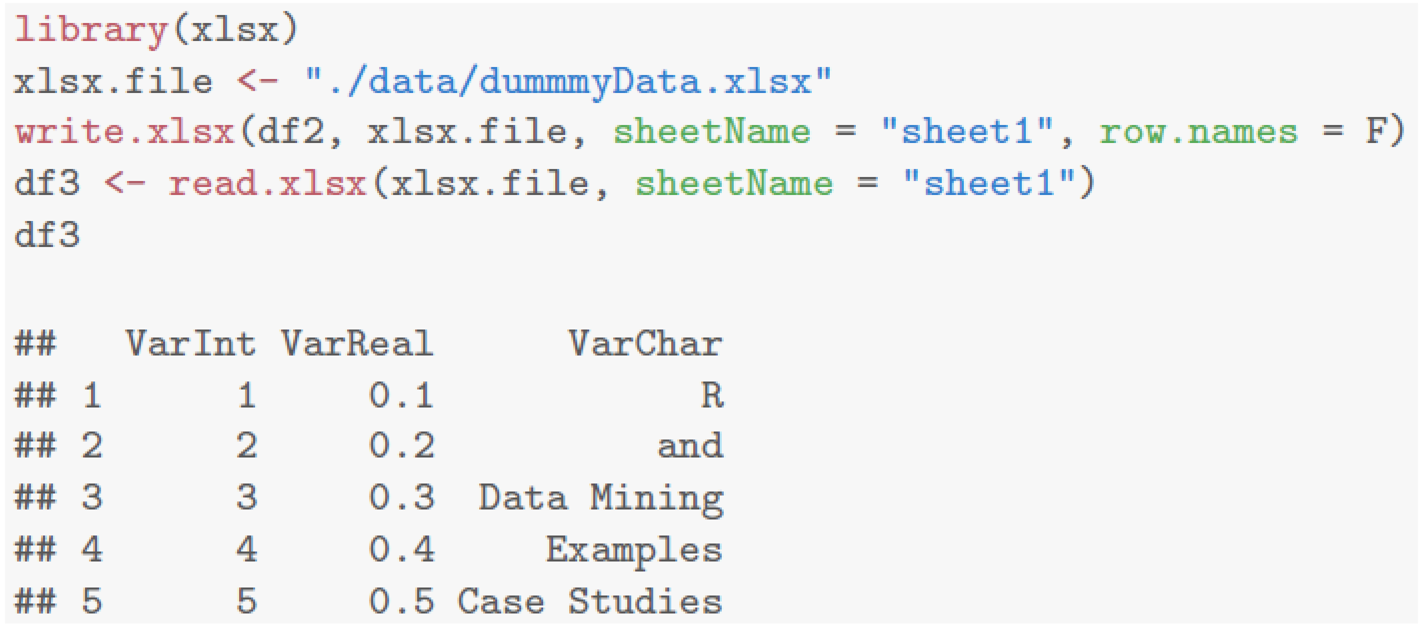
read.csv(): read an R object from a .CSV file



```
# create a data frame
var1 <- 1:5
var2 <- (1:5)/10
var3 <- c("R", "and", "Data Mining", "Examples", "Case Studies")
df1 <- data.frame(var1, var2, var3)
names(df1) <- c("VarInt", "VarReal", "VarChar")
# save to a csv file
write.csv(df1, "./data/dummmyData.csv", row.names = FALSE)
# read from a csv file
df2 <- read.csv("./data/dummmyData.csv")
print(df2)

##   VarInt VarReal      VarChar
## 1      1     0.1            R
## 2      2     0.2          and
## 3      3     0.3  Data Mining
## 4      4     0.4     Examples
## 5      5     0.5 Case Studies
```

IMPORT FROM AND EXPORT TO EXCEL FILES

Package xlsx: read, write, format Excel 2007 and Excel 97/2000/XP/2003 files



```r
library(xlsx)
xlsx.file <- "./data/dummmyData.xlsx"
write.xlsx(df2, xlsx.file, sheetName = "sheet1", row.names = F)
df3 <- read.xlsx(xlsx.file, sheetName = "sheet1")
df3

##   VarInt VarReal      VarChar
## 1      1     0.1            R
## 2      2     0.2          and
## 3      3     0.3  Data Mining
## 4      4     0.4     Examples
## 5      5     0.5 Case Studies
```

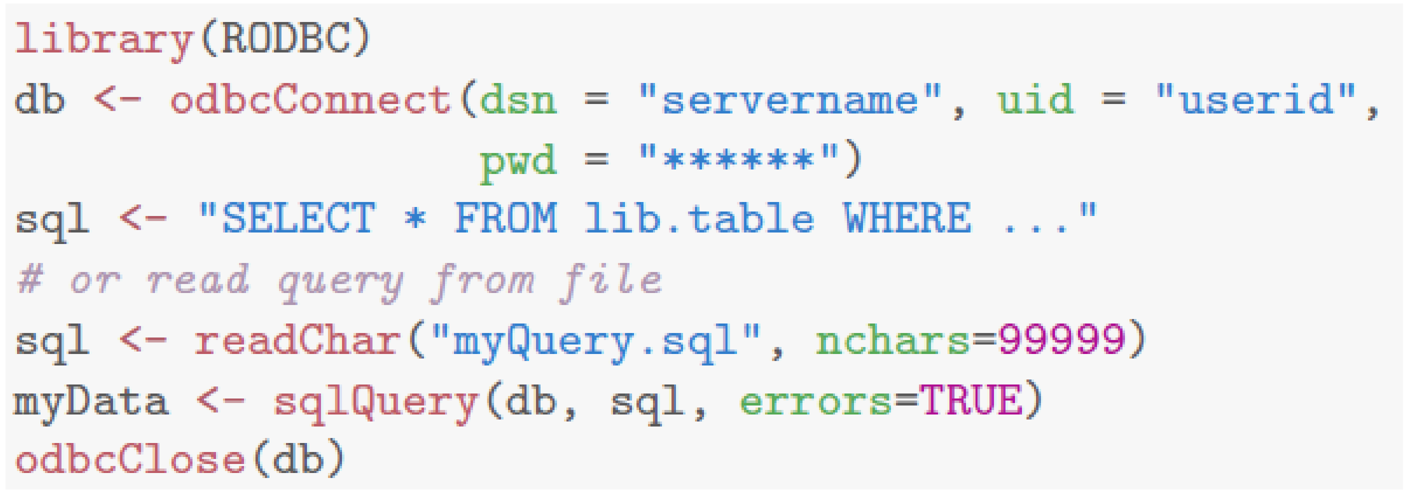
READ FROM DATABASES

Package RODBC: provides connection to ODBC databases.

Function odbcConnect(): sets up a connection to database

sqlQuery(): sends an SQL query to the database

odbcClose() closes the connection.

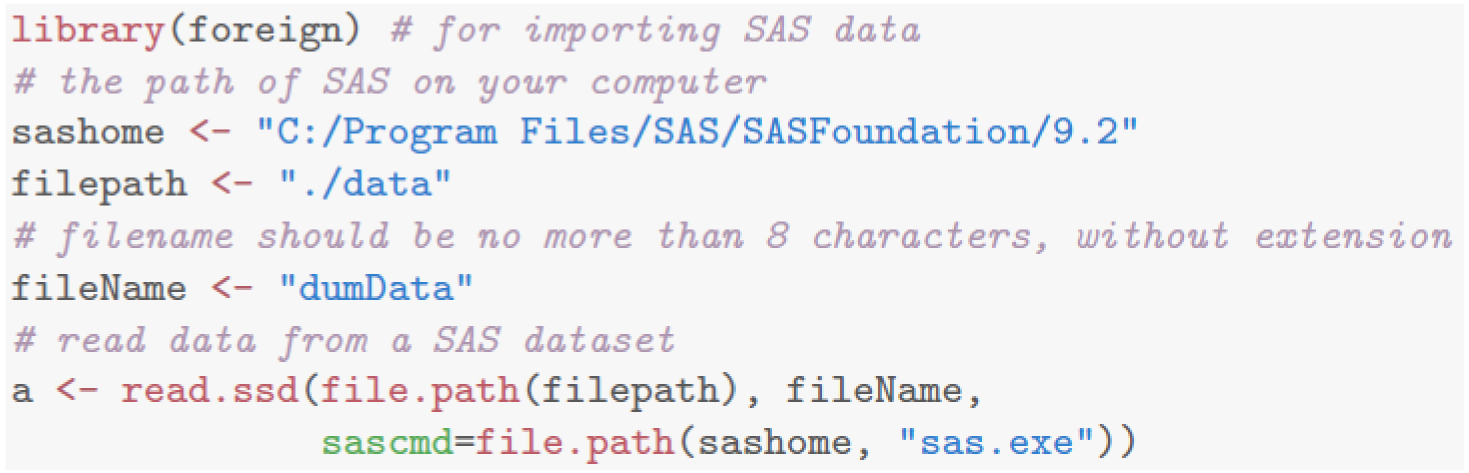


```
library(RODBC)
db <- odbcConnect(dsn = "servername", uid = "userid",
                  pwd = "******")
sql <- "SELECT * FROM lib.table WHERE ..."
# or read query from file
sql <- readChar("myQuery.sql", nchars=99999)
myData <- sqlQuery(db, sql, errors=TRUE)
odbcClose(db)
```

Functions sqlFetch(), sqlSave() and sqlUpdate(): read, write or update a table in an ODBC database

IMPORT DATA FROM SAS

Package foreign provides function read.ssd() for importing SAS datasets (.sas7bdat files) into R.



```r
library(foreign) # for importing SAS data
# the path of SAS on your computer
sashome <- "C:/Program Files/SAS/SASFoundation/9.2"
filepath <- "./data"
# filename should be no more than 8 characters, without extension
fileName <- "dumData"
# read data from a SAS dataset
a <- read.ssd(file.path(filepath), fileName,
              sascmd=file.path(sashome, "sas.exe"))
```

DATA EXPLORATION AND VISUALIZATION

DATA EXPLORATION AND VISUALIZATION WITH R

Data Exploration and Visualization

- Summary and stats
- Various charts like pie charts and histograms
- Exploration of multiple variables
- Level plot, contour plot and 3D plot
- Saving charts into files of various formats

SIZE AND STRUCTURE OF DATA

```
dim(iris)

## [1] 150   5

names(iris)

## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Wid...
## [5] "Species"

str(iris)

## 'data.frame': 150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0...
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",....
```

ATTRIBUTES OF DATA

```
attributes(iris)

## $names
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Wid...
## [5] "Species"
##
## $row.names
##   [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  ...
##  [16]  16  17  18  19  20  21  22  23  24  25  26  27  28  ...
##  [31]  31  32  33  34  35  36  37  38  39  40  41  42  43  ...
##  [46]  46  47  48  49  50  51  52  53  54  55  56  57  58  ...
##  [61]  61  62  63  64  65  66  67  68  69  70  71  72  73  ...
##  [76]  76  77  78  79  80  81  82  83  84  85  86  87  88  ...
##  [91]  91  92  93  94  95  96  97  98  99 100 101 102 103 1...
## [106] 106 107 108 109 110 111 112 113 114 115 116 117 118 1...
## [121] 121 122 123 124 125 126 127 128 129 130 131 132 133 1...
## [136] 136 137 138 139 140 141 142 143 144 145 146 147 148 1...
##
## $class
## [1] "data.frame"
```

FIRST ROWS OF DATA

```
iris[1:3, ]

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa

head(iris, 3)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa

tail(iris, 3)

##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Spe...
## 148          6.5         3.0          5.2         2.0 virgi...
## 149          6.2         3.4          5.4         2.3 virgi...
## 150          5.9         3.0          5.1         1.8 virgi...
```

A SINGLE COLUMN

The first 10 values of Sepal.Length

```
iris[1:10, "Sepal.Length"]

##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9

iris$Sepal.Length[1:10]

##  [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
```

SUMMARY OF DATA

Function summary()

- numeric variables: minimum, maximum, mean, median, and the first (25%) and third (75%) quartiles
- categorical variables (factors): frequency of every level

```
summary(iris)

##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.30   Min.   :2.00   Min.   :1.00   Min.   :0.1
##  1st Qu.:5.10   1st Qu.:2.80   1st Qu.:1.60   1st Qu.:0.3
##  Median :5.80   Median :3.00   Median :4.35   Median :1.3
##  Mean   :5.84   Mean   :3.06   Mean   :3.76   Mean   :1.2
##  3rd Qu.:6.40   3rd Qu.:3.30   3rd Qu.:5.10   3rd Qu.:1.8
##  Max.   :7.90   Max.   :4.40   Max.   :6.90   Max.   :2.5
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
```

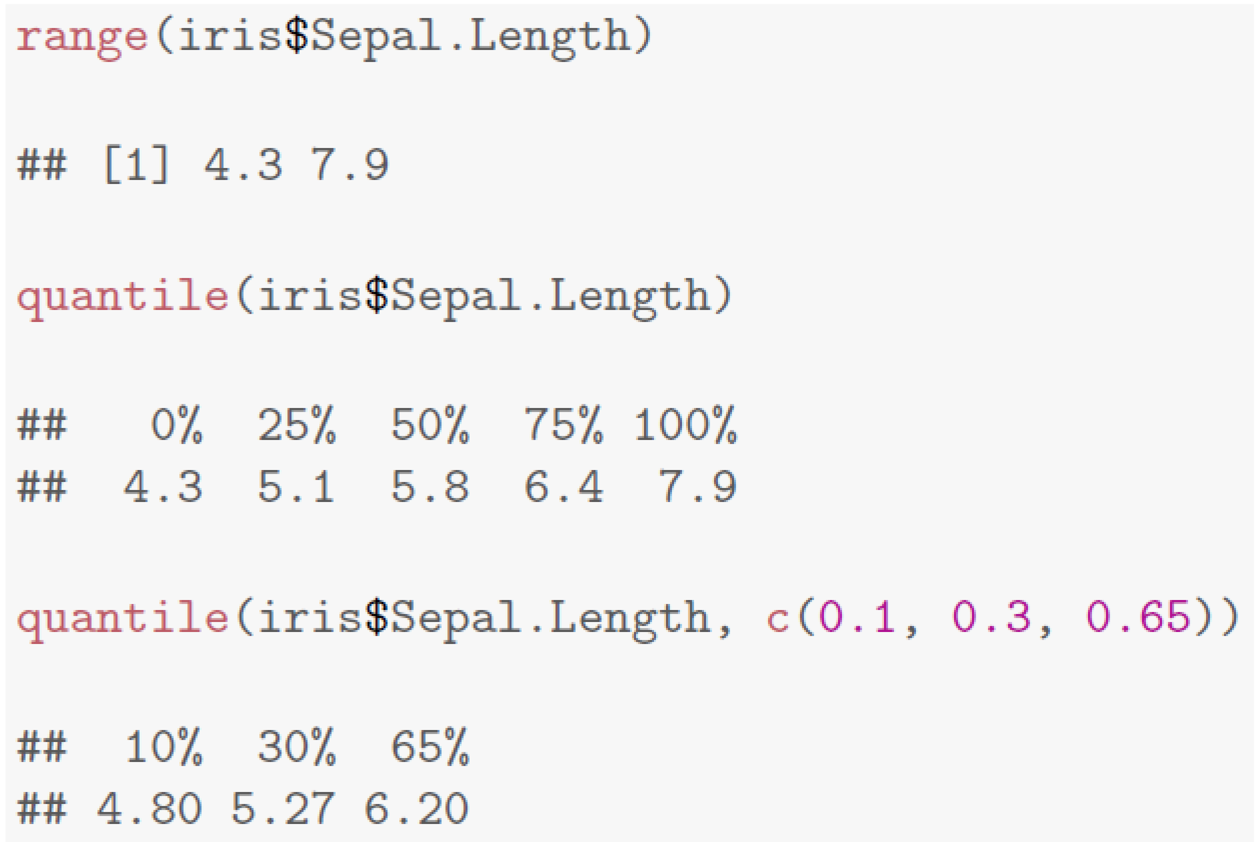
```r
library(Hmisc)
describe(iris[, c(1, 5)])  # check columns 1 & 5
```

```
## iris[, c(1, 5)]
##
##  2  Variables      150  Observations
## ---------------------------------------------------------...
## Sepal.Length
##        n missing  unique    Info    Mean     .05     .10    ...
##      150       0      35       1   5.843   4.600   4.800   5...
##      .50     .75     .90     .95
##    5.800   6.400   6.900   7.255
##
## lowest : 4.3 4.4 4.5 4.6 4.7, highest: 7.3 7.4 7.6 7.7 7.9
## ---------------------------------------------------------...
## Species
##        n missing  unique
##      150       0       3
##
## setosa (50, 33%), versicolor (50, 33%)
## virginica (50, 33%)
## ---------------------------------------------------------...
```

MEAN, MEDIAN, RANGE AND QUARTILES

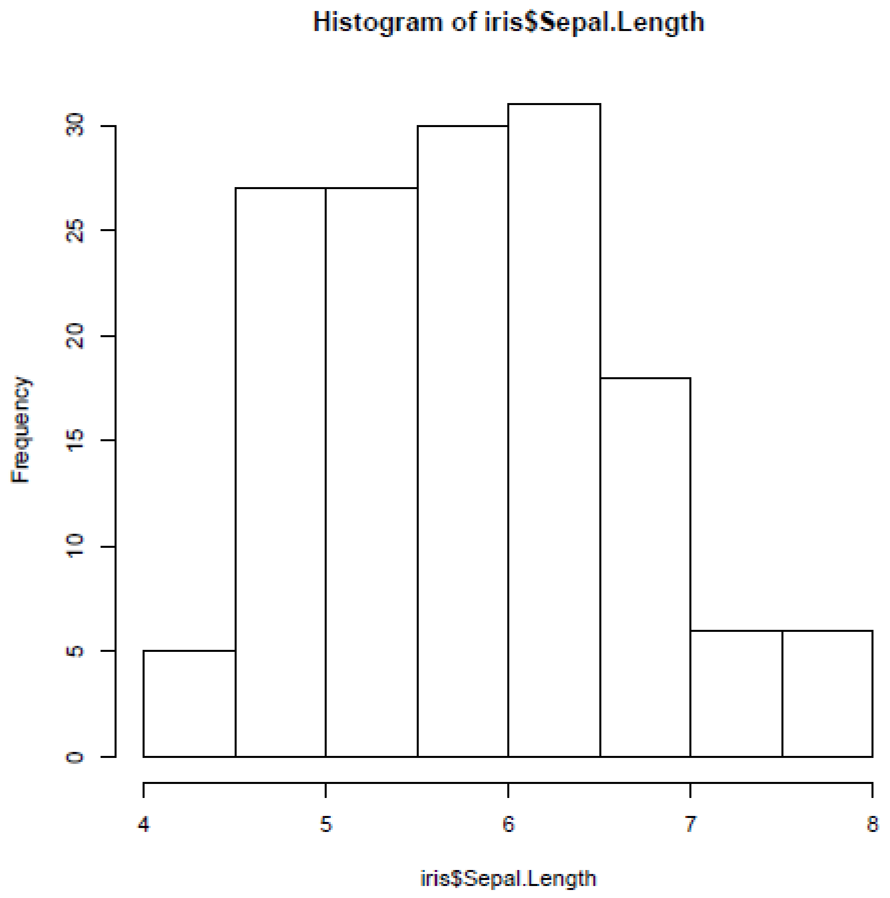
Mean, median and range: mean(), median(), range()

Quartiles and percentiles: quantile()



```
range(iris$Sepal.Length)

## [1] 4.3 7.9

quantile(iris$Sepal.Length)

##   0%  25%  50%  75% 100%
##  4.3  5.1  5.8  6.4  7.9

quantile(iris$Sepal.Length, c(0.1, 0.3, 0.65))

##  10%  30%  65%
## 4.80 5.27 6.20
```

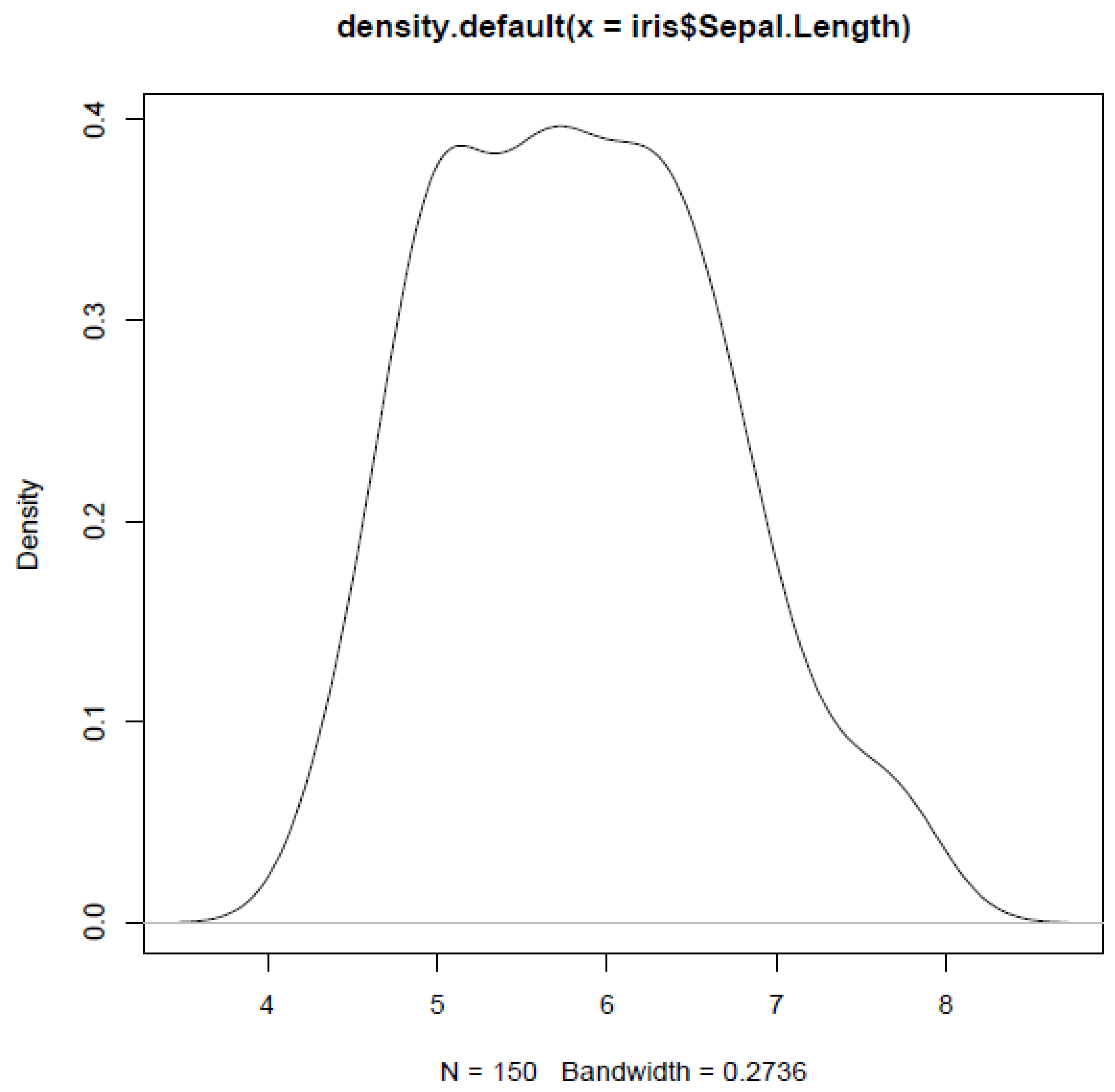
VARIANCE AND HISTOGRAM



```
var(iris$Sepal.Length)

## [1] 0.6857

hist(iris$Sepal.Length)
```

DENSITY

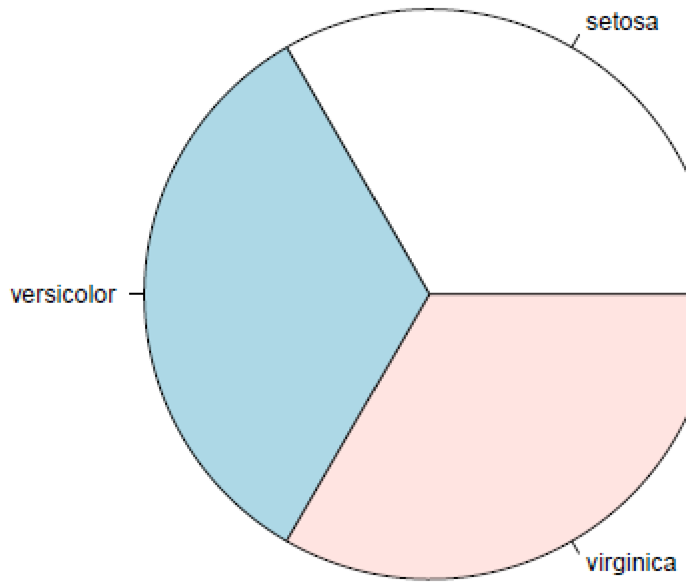
```
plot(density(iris$Sepal.Length))
```



PIE CHART

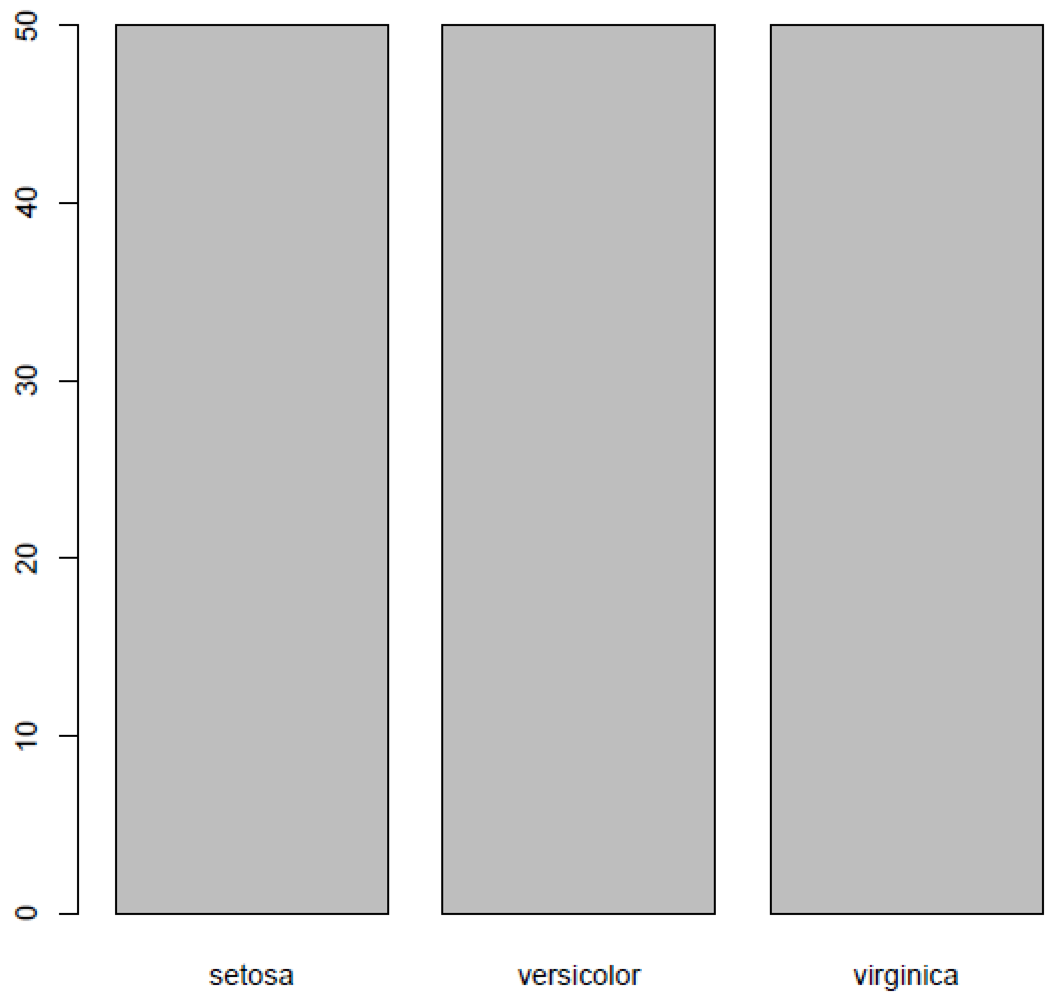
Frequency of factors: table()

```
table(iris$Species)

##
##     setosa versicolor  virginica
##         50         50         50

pie(table(iris$Species))
```



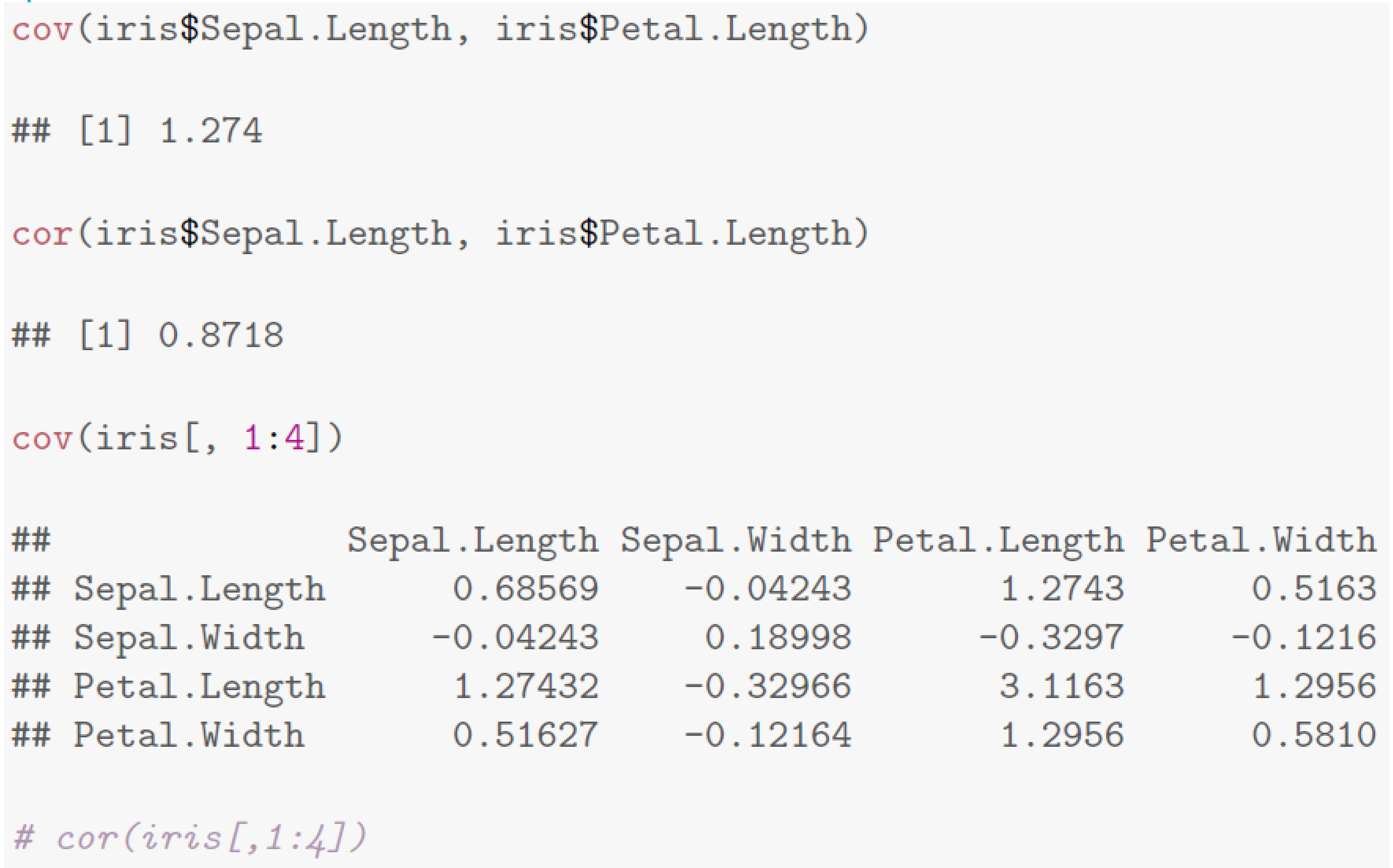
BAR CHART

```
barplot(table(iris$Species))
```



CORRELATION

Covariance and correlation: cov() and cor()



```
cov(iris$Sepal.Length, iris$Petal.Length)

## [1] 1.274

cor(iris$Sepal.Length, iris$Petal.Length)

## [1] 0.8718

cov(iris[, 1:4])

##              Sepal.Length Sepal.Width Petal.Length Petal.Width
## Sepal.Length      0.68569    -0.04243       1.2743      0.5163
## Sepal.Width      -0.04243     0.18998      -0.3297     -0.1216
## Petal.Length      1.27432    -0.32966       3.1163      1.2956
## Petal.Width       0.51627    -0.12164       1.2956      0.5810

# cor(iris[,1:4])
```

AGGREATION

Stats of Sepal.Length for every Species with aggregate()

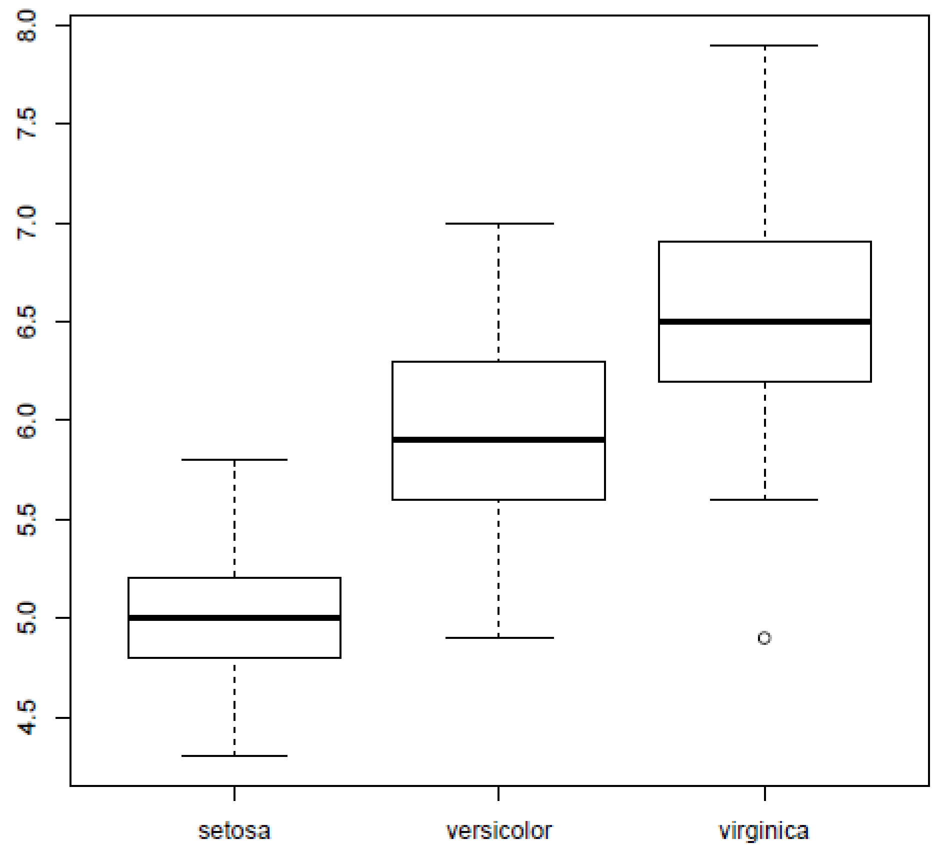
```
aggregate(Sepal.Length ~ Species, summary, data = iris)

##      Species Sepal.Length.Min. Sepal.Length.1st Qu.
## 1     setosa              4.30                 4.80
## 2 versicolor              4.90                 5.60
## 3  virginica              4.90                 6.22
##   Sepal.Length.Median Sepal.Length.Mean Sepal.Length.3rd Qu.
## 1                5.00              5.01                 5.20
## 2                5.90              5.94                 6.30
## 3                6.50              6.59                 6.90
##   Sepal.Length.Max.
## 1              5.80
## 2              7.00
## 3              7.90
```

BOXPLOT

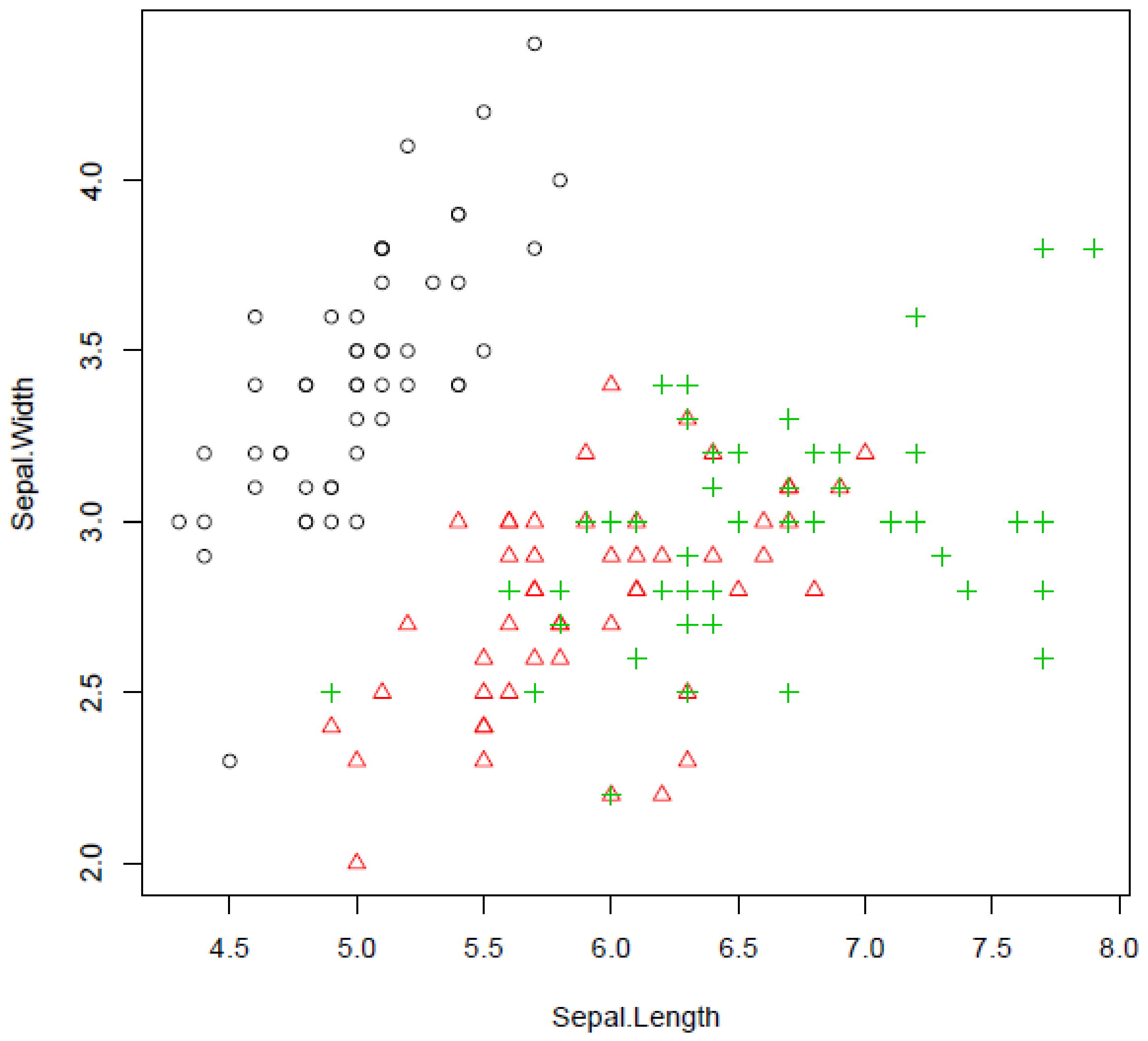
The bar in the middle is median.

The box shows the interquartile range (IQR), i.e., range between the 75% and 25% observation.

```
boxplot(Sepal.Length ~ Species, data = iris)
```



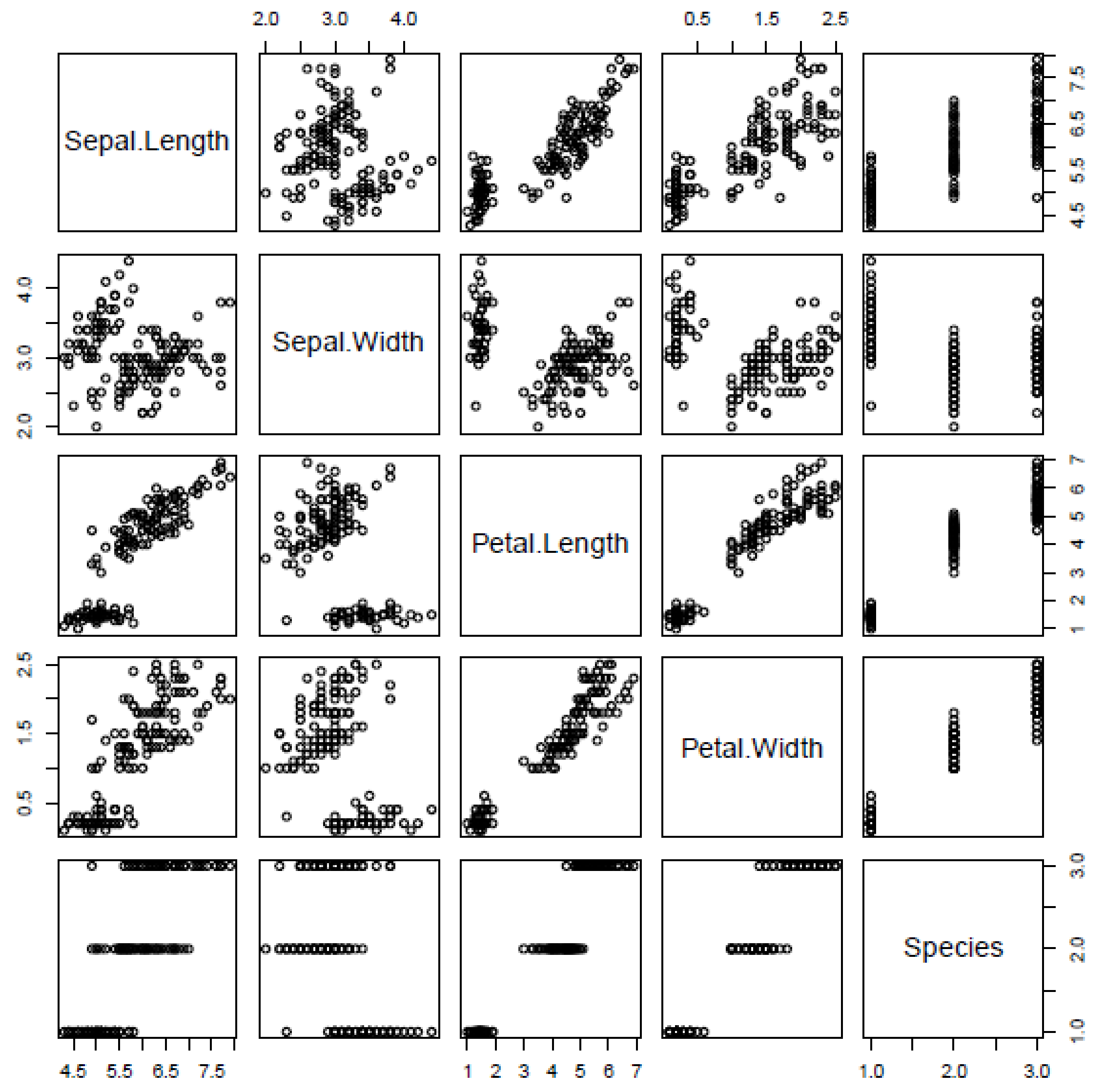
SCATTER PLOT



```
with(iris, plot(Sepal.Length, Sepal.Width, col = Species,
                pch = as.numeric(Species)))
```

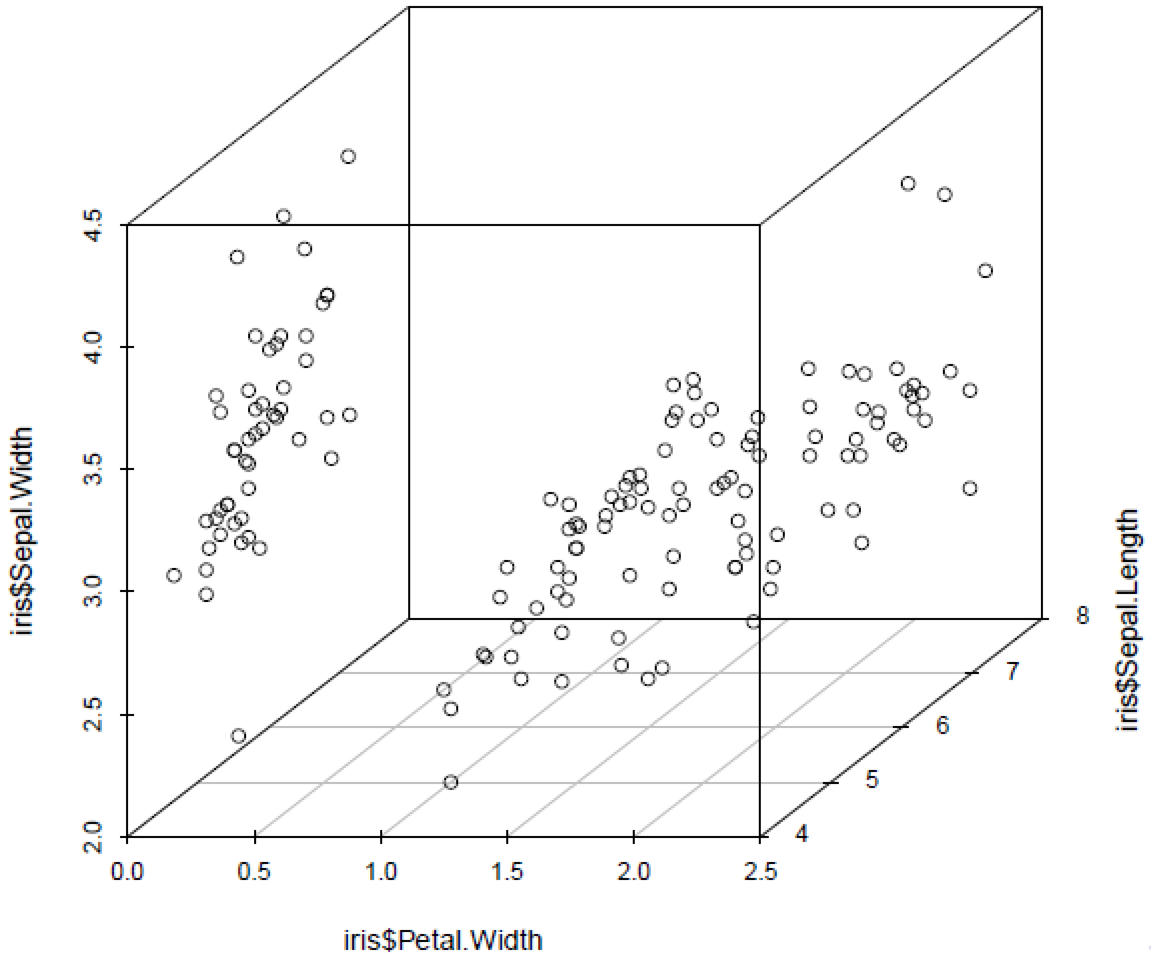
A MATRIX OF SCATTER PLOTS

```
pairs(iris)
```



3D SCATTER PLOT

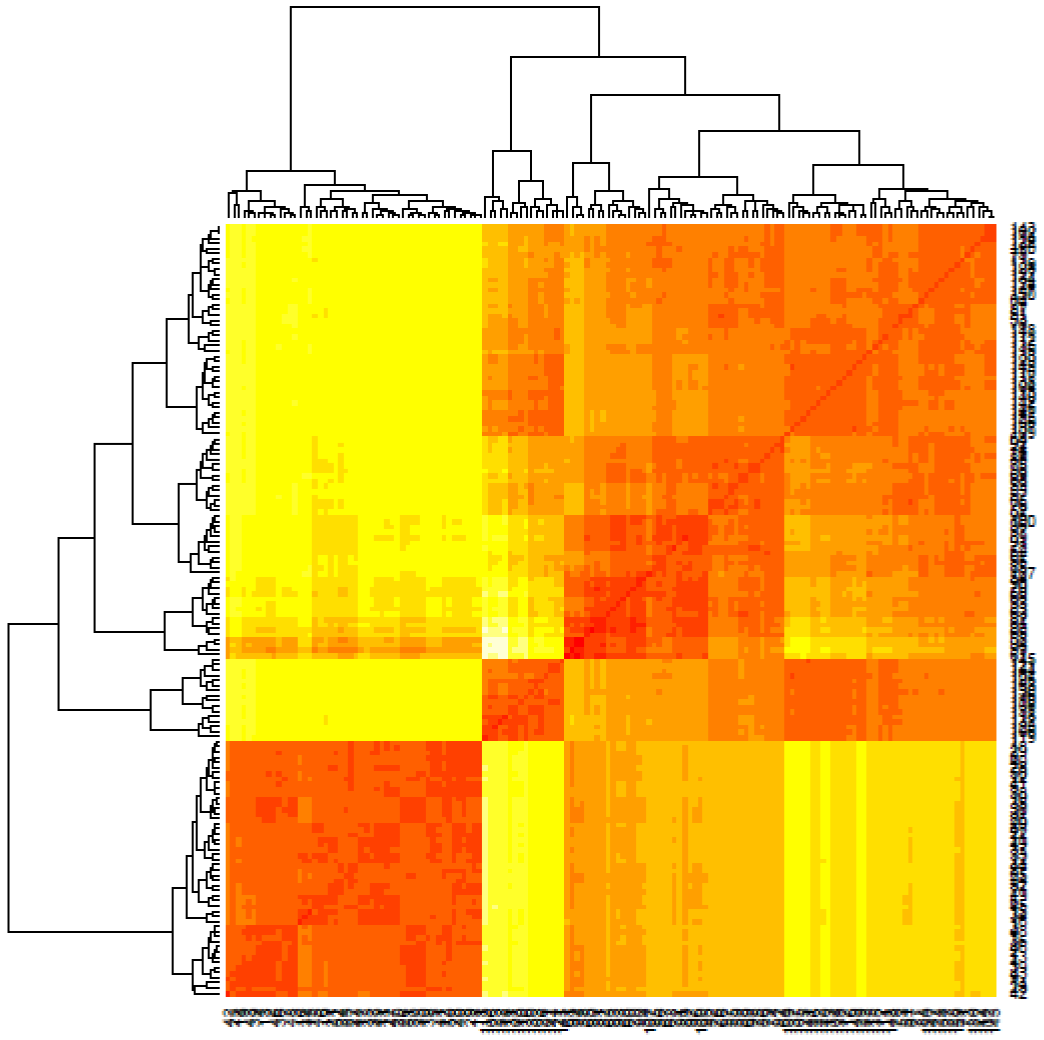
```
library(scatterplot3d)
scatterplot3d(iris$Petal.Width, iris$Sepal.Length, iris$Sepal.Width)
```



HEAT MAP

Calculate the similarity between different flowers in the iris data with dist() and then plot it with a heat map

```
dist.matrix <- as.matrix(dist(iris[, 1:4]))
heatmap(dist.matrix)
```

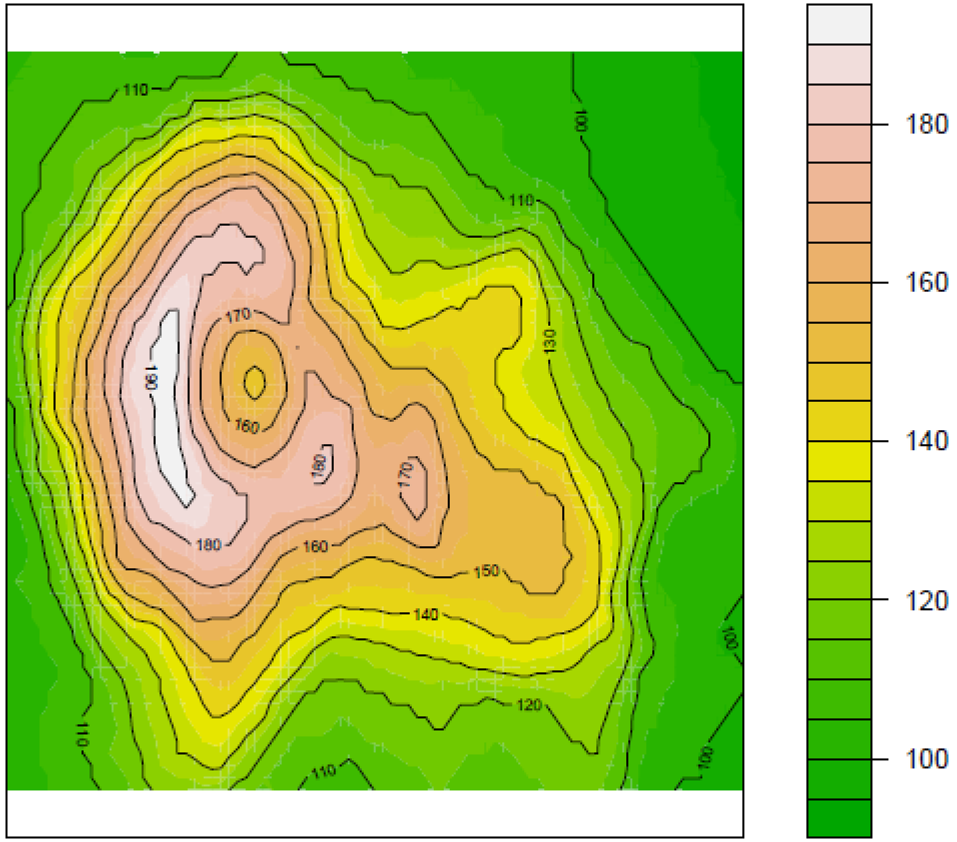


CONTOUR

```
filled.contour(volcano, color = terrain.colors, asp = 1, plot.axes = co
    add = T))
```

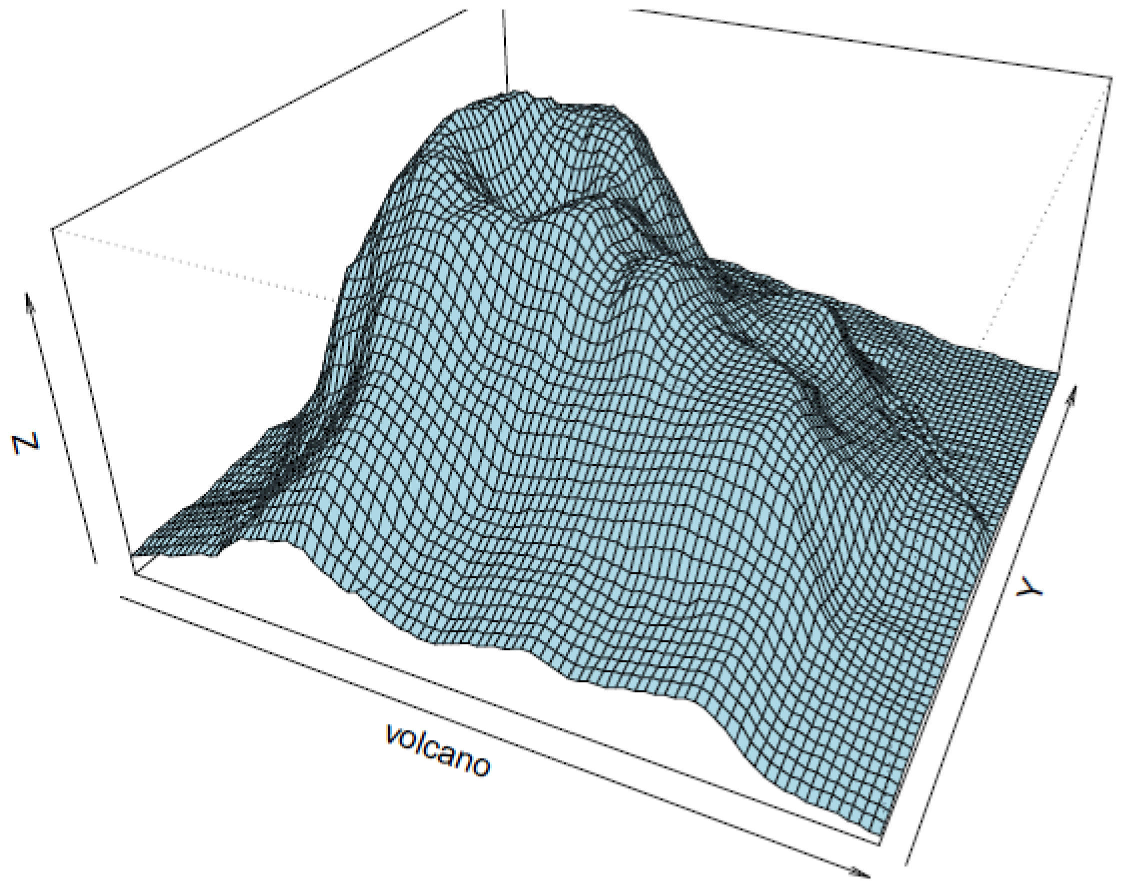
contour() and filled.contour() in package graphics

contourplot() in package lattice



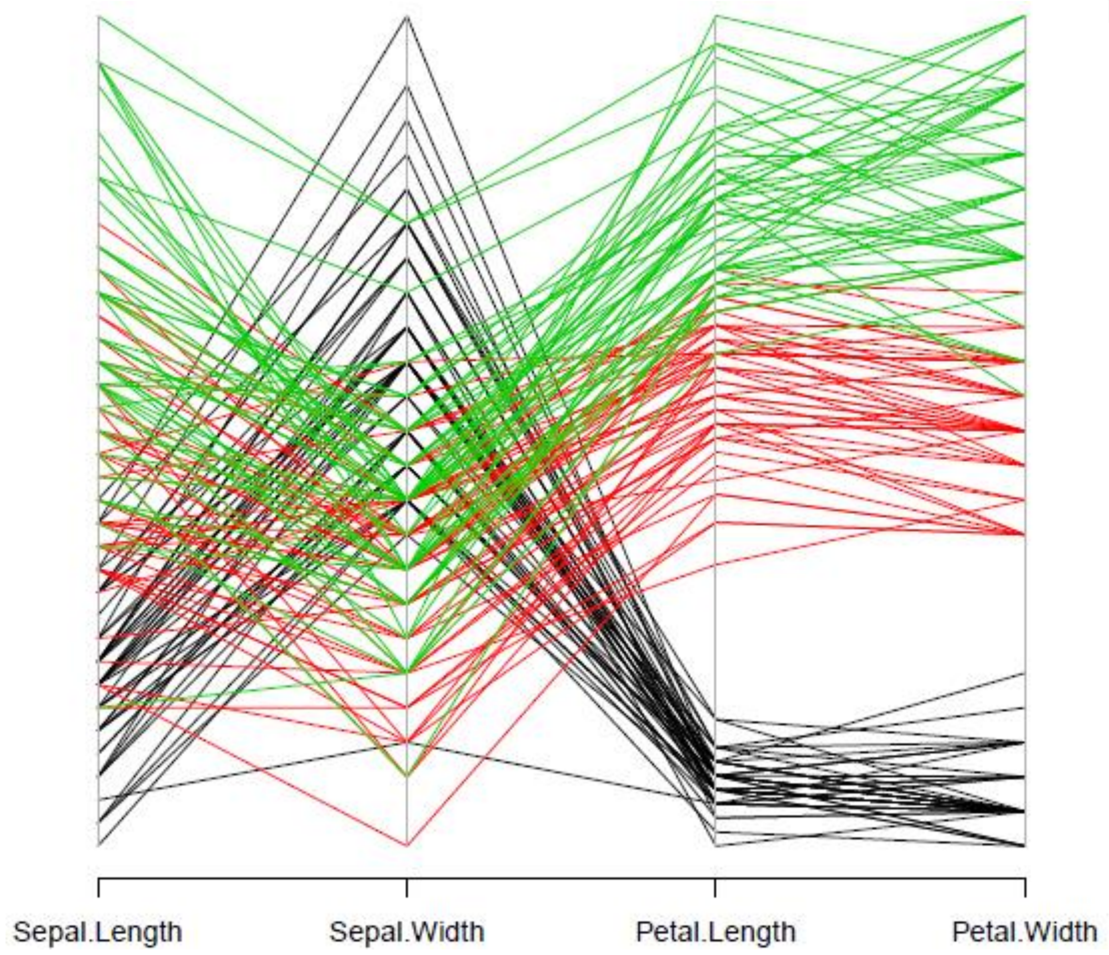
3D SURFACE

```r
persp(volcano, theta = 25, phi = 30, expand = 0.5, col = "lightblue")
```



PARALLEL COORDINATES

```
library(MASS)
parcoord(iris[1:4], col = iris$Species)
```

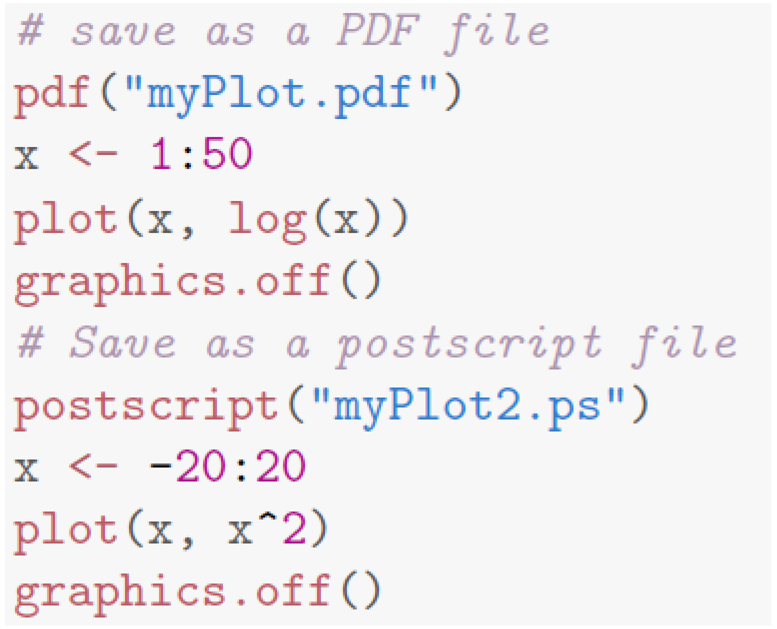


SAVE CHARTS TO FILES

Save charts to PDF and PS files: pdf() and postscript()

BMP, JPEG, PNG and TIFF files: bmp(), jpeg(), png() and tiff()

Close files (or graphics devices) with graphics.off() or dev.off() after plotting



```r
# save as a PDF file
pdf("myPlot.pdf")
x <- 1:50
plot(x, log(x))
graphics.off()
# Save as a postscript file
postscript("myPlot2.ps")
x <- -20:20
plot(x, x^2)
graphics.off()
```

CLASSIFICATION WITH R

CLASSIFICATION WITH R

Decision trees: rpart, party

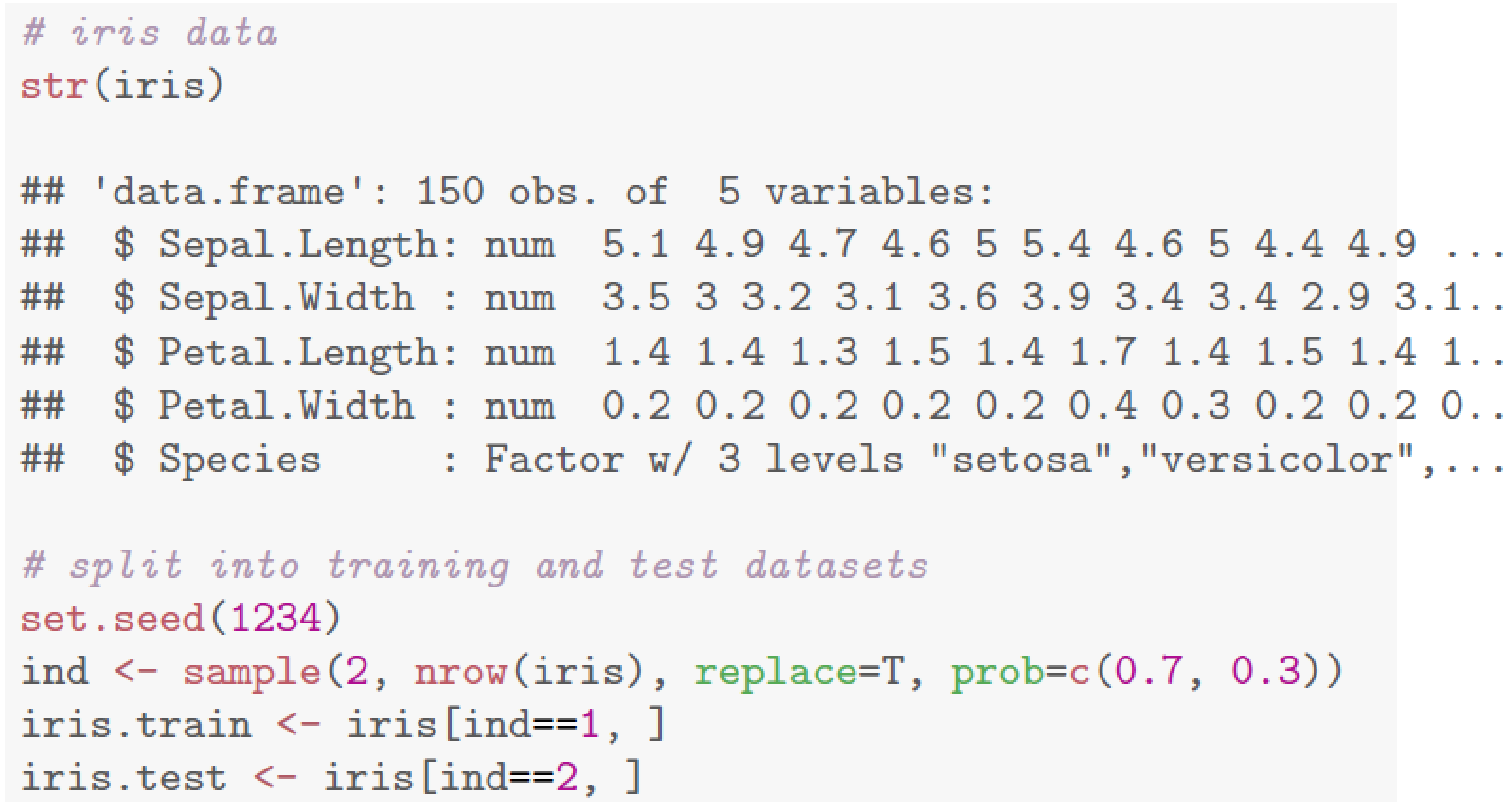
Random forest: randomForest, party

SVM: e1071, kernlab

Neural networks: nnet, neuralnet, RSNNS

Performance evaluation: ROCR

THE IRIS DATASET

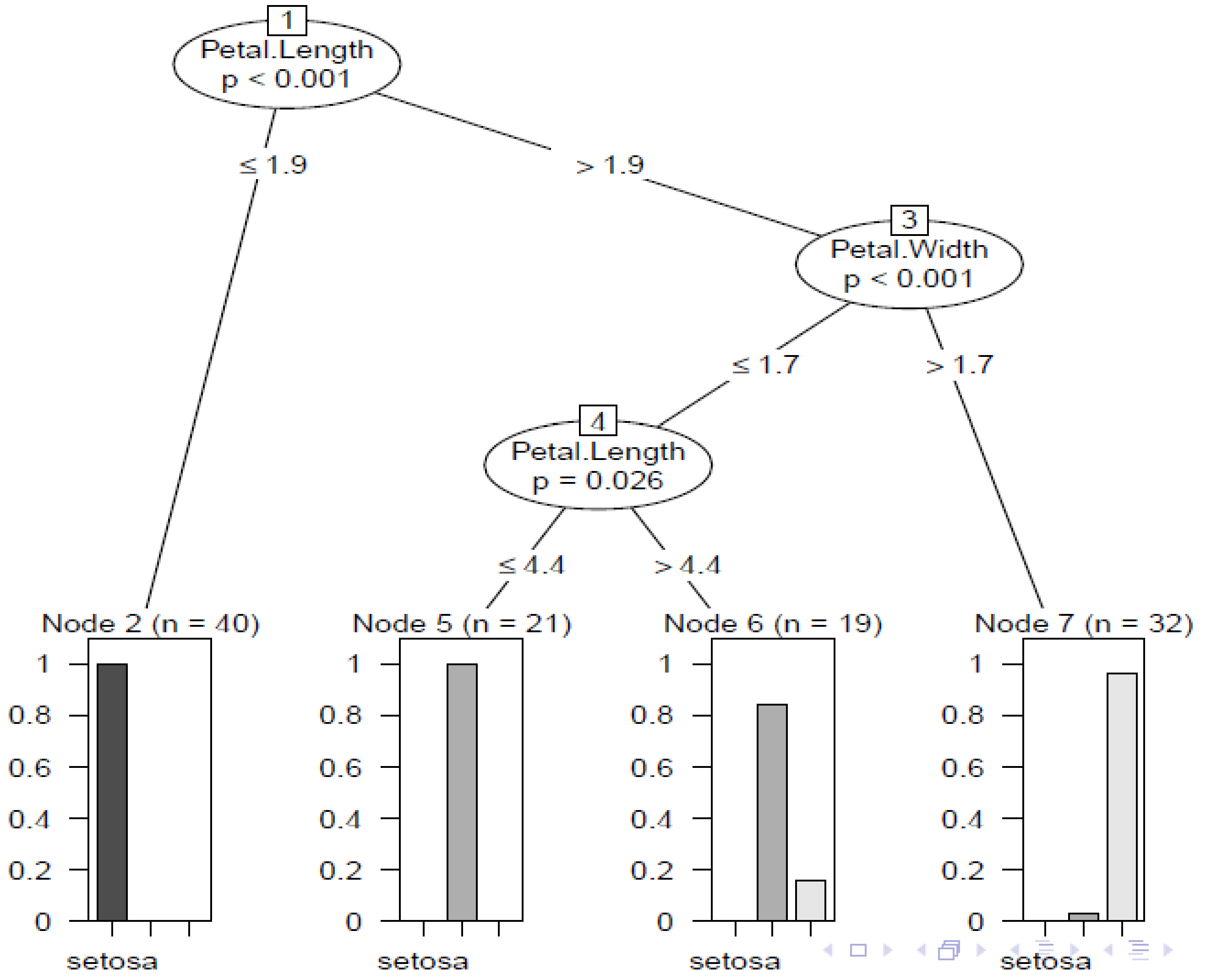


```
# iris data
str(iris)

## 'data.frame': 150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1..
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1..
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0..
##  $ Species     : Factor w/ 3 levels "setosa","versicolor",...

# split into training and test datasets
set.seed(1234)
ind <- sample(2, nrow(iris), replace=T, prob=c(0.7, 0.3))
iris.train <- iris[ind==1, ]
iris.test <- iris[ind==2, ]
```

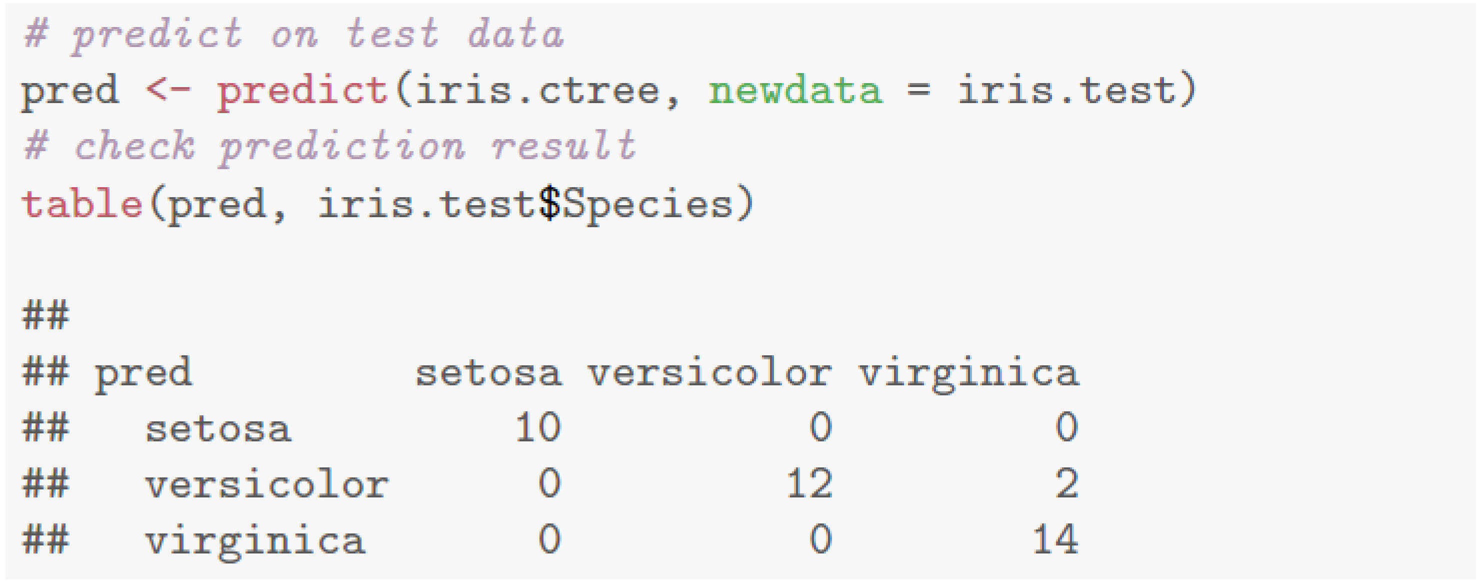
BUILD A DECISION TREE

```
# build a decision tree
library(party)
iris.formula <- Species ~ Sepal.Length + Sepal.Width +
                          Petal.Length + Petal.Width
iris.ctree <- ctree(iris.formula, data=iris.train)
```

```
plot(iris.ctree)
```



PREDICTION



```
# predict on test data
pred <- predict(iris.ctree, newdata = iris.test)
# check prediction result
table(pred, iris.test$Species)

##
## pred         setosa versicolor virginica
##   setosa         10          0         0
##   versicolor      0         12         2
##   virginica       0          0        14
```

R PACKAGES FOR RANDOM FOREST

Package randomForest

- very fast
- cannot handle data with missing values
- a limit of 32 to the maximum number of levels of each categorical attribute

Package party: cforest()

- not limited to the above maximum levels
- slow
- needs more memory

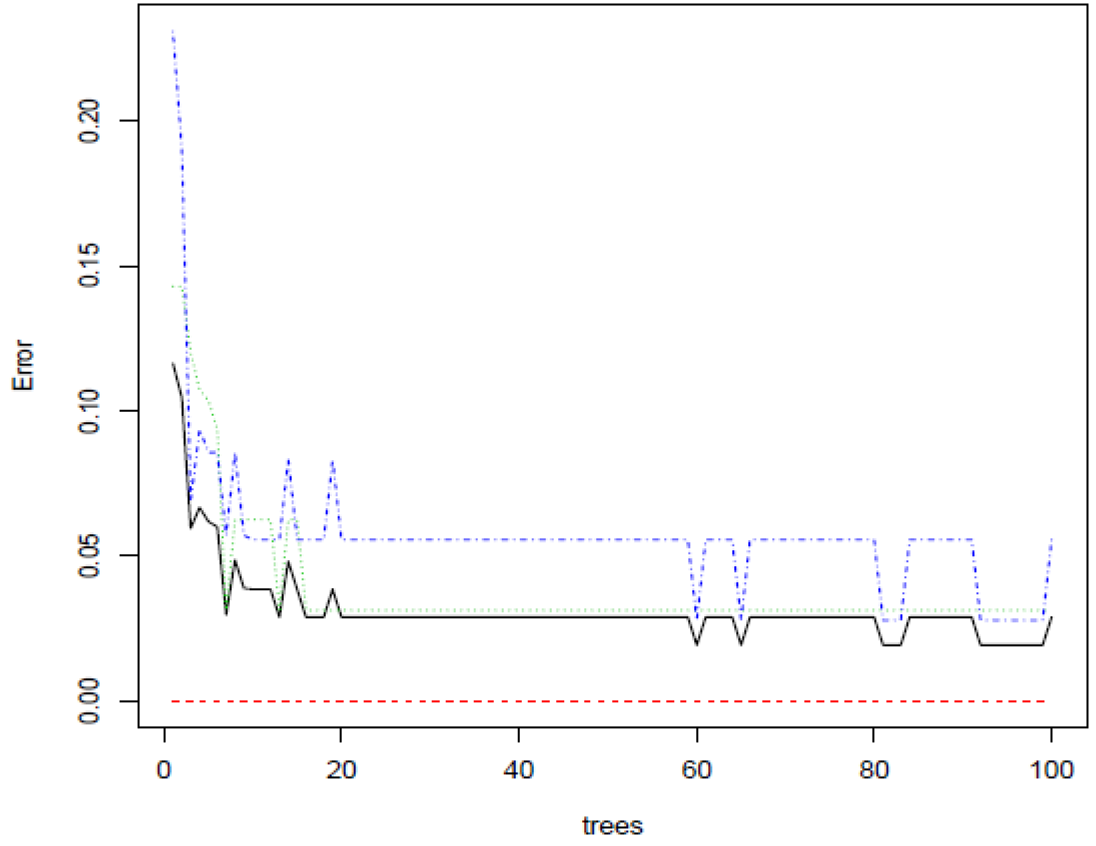
TRAIN A RANDOM FOREST

```
# split into two subsets: training (70%) and test (30%)
ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
train.data <- iris[ind==1,]
test.data <- iris[ind==2,]
# use all other variables to predict Species
library(randomForest)
rf <- randomForest(Species ~ ., data=train.data, ntree=100,
                   proximity=T)
```

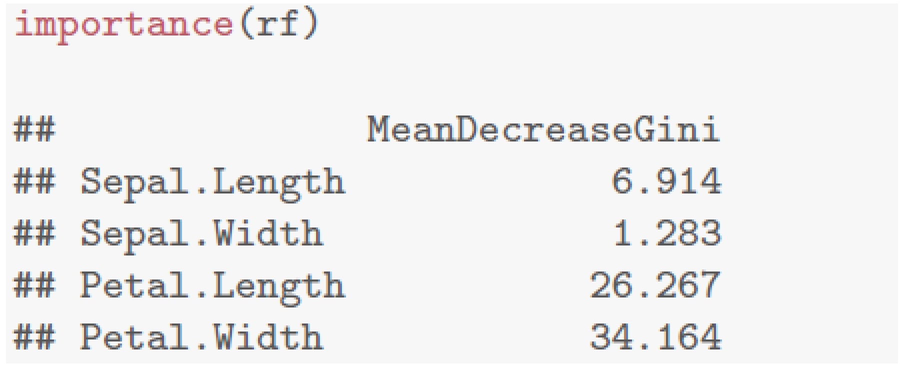
```
table(predict(rf), train.data$Species)

##
##               setosa versicolor virginica
##   setosa          36          0         0
##   versicolor       0         31         2
##   virginica        0          1        34

print(rf)

##
## Call:
##  randomForest(formula = Species ~ ., data = train.data, ntr...
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.88%
## Confusion matrix:
##            setosa versicolor virginica class.error
## setosa         36          0         0     0.00000
## versicolor      0         31         1     0.03125
## virginica       0          2        34     0.05556
```

ERROR RATE OF RANDOM FOREST

```r
plot(rf, main = "")
```



VARIABLE IMPORTANCE



```
importance(rf)

##              MeanDecreaseGini
## Sepal.Length            6.914
## Sepal.Width             1.283
## Petal.Length           26.267
## Petal.Width            34.164
```

CLUSTERING WITH R

CLUSTERING WITH R

k-means: kmeans(), kmeansruns()

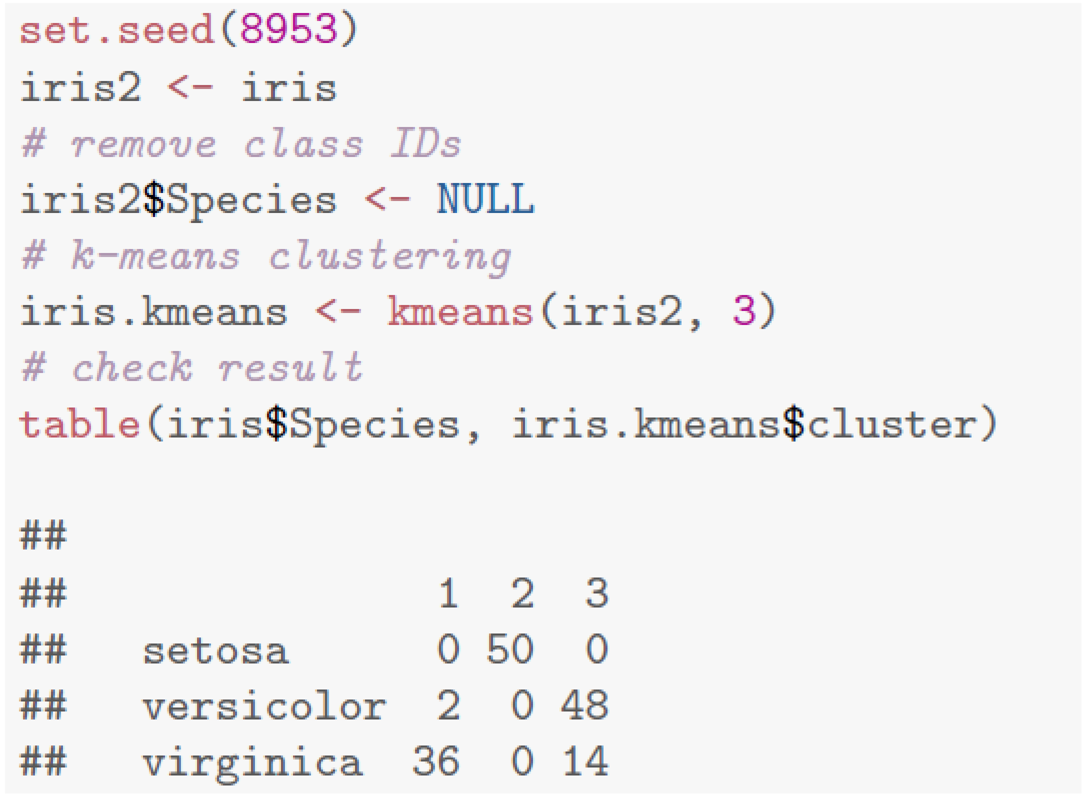
k-medoids: pam(), pamk()

Hierarchical clustering: hclust(), agnes(), diana()

DBSCAN: fpc

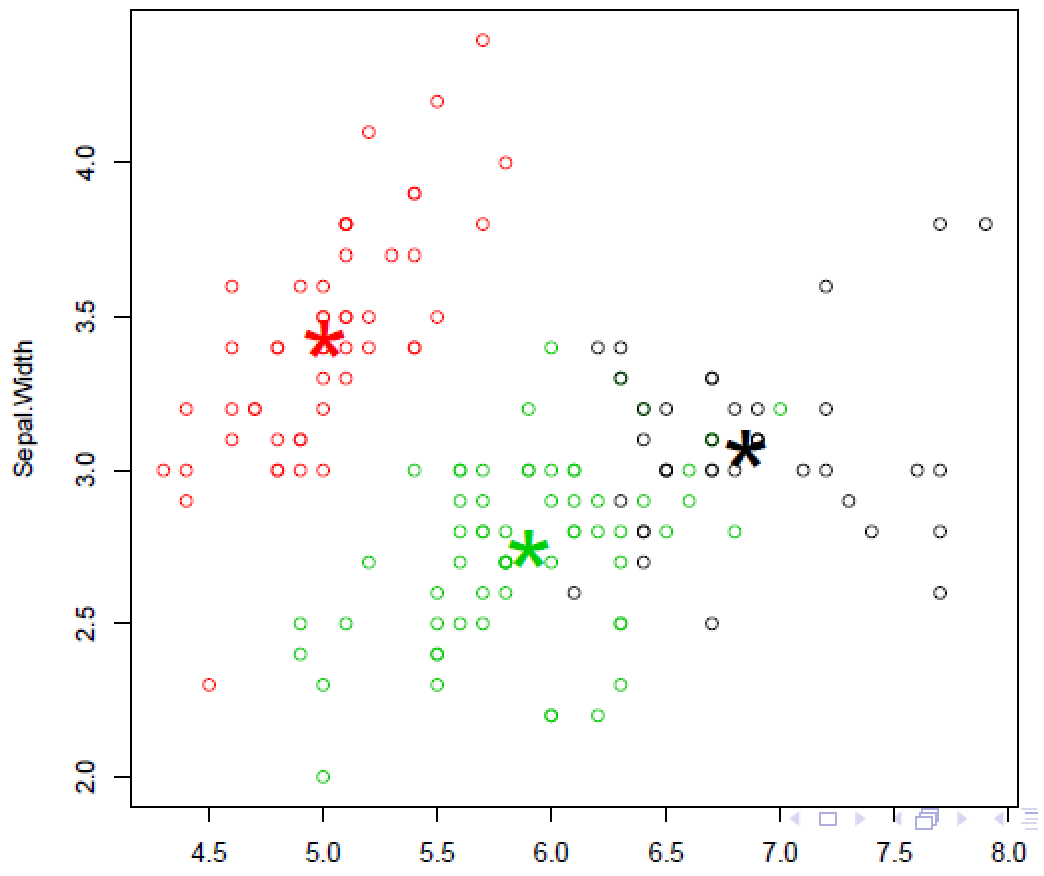
BIRCH: birch

K-MEANS CLUSTERING



```
set.seed(8953)
iris2 <- iris
# remove class IDs
iris2$Species <- NULL
# k-means clustering
iris.kmeans <- kmeans(iris2, 3)
# check result
table(iris$Species, iris.kmeans$cluster)

##
##                 1  2  3
##   setosa        0 50  0
##   versicolor    2  0 48
##   virginica    36  0 14
```

```r
# plot clusters and their centers
plot(iris2[c("Sepal.Length", "Sepal.Width")], col=iris.kmeans$cluster)
points(iris.kmeans$centers[, c("Sepal.Length", "Sepal.Width")],
       col=1:3, pch="*", cex=5)
```



THE K-MEDOIDS CLUSTERING

Difference from k-means: a cluster is represented with its center in the k-means algorithm, but with the object closest to the center of the cluster in the k-medoids clustering.

more robust than k-means in presence of outliers

PAM (Partitioning Around Medoids) is a classic algorithm for k-medoids clustering.

The CLARA algorithm is an enhanced technique of PAM by drawing multiple samples of data, applying PAM on each sample and then returning the best clustering. It performs better than PAM on larger data.

Functions pam() and clara() in package cluster

Function pamk() in package fpc does not require a user to choose k.

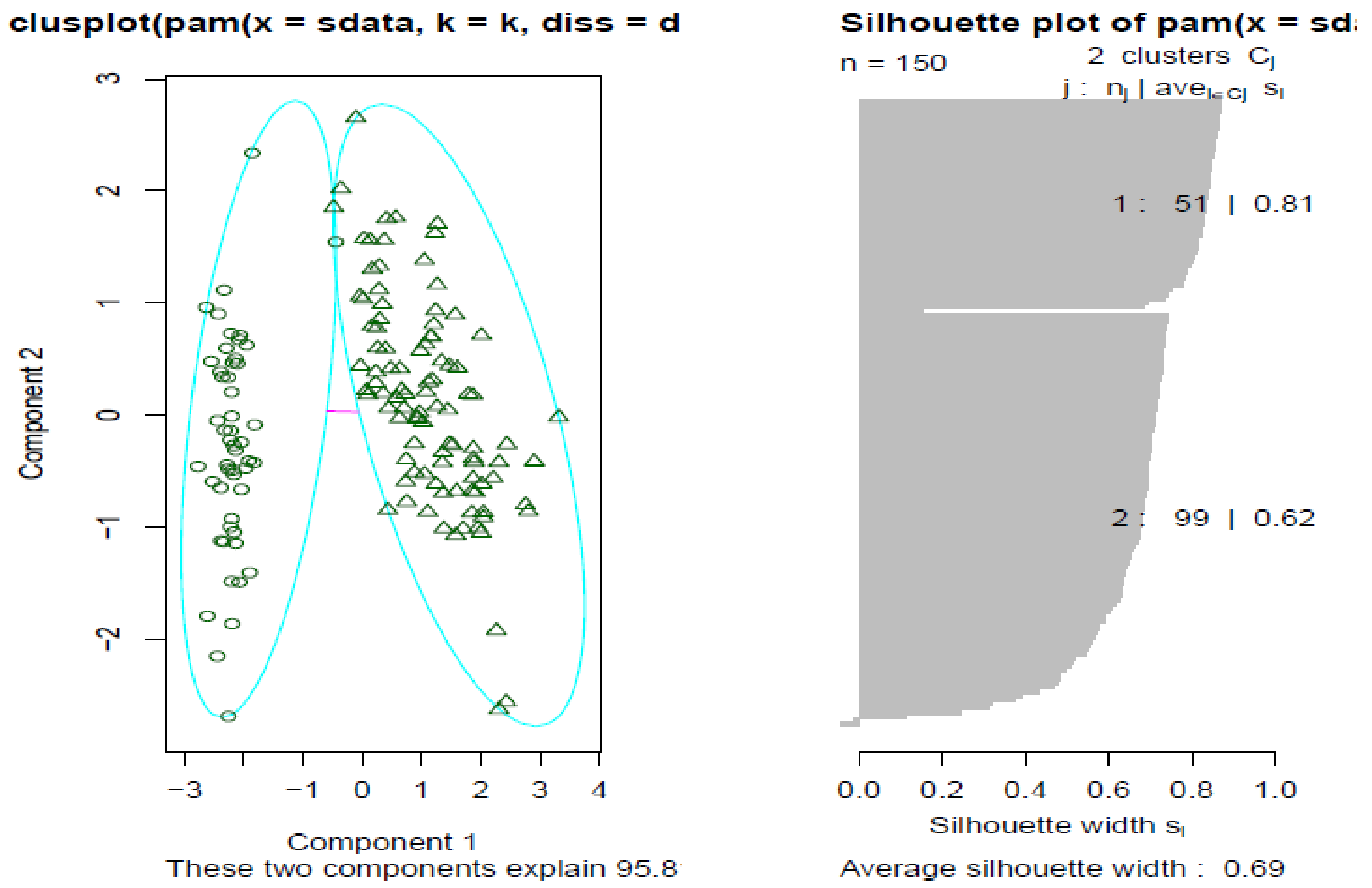
CLUSTERING WITH PAMK()

Two clusters:

- “setosa"
- a mixture of “versicolor" and “virginica"

```
library(fpc)
pamk.result <- pamk(iris2)
# number of clusters
pamk.result$nc

## [1] 2

# check clustering against actual species
table(pamk.result$pamobject$clustering, iris$Species)

##
##     setosa versicolor virginica
##   1     50          1         0
##   2      0         49        50
```

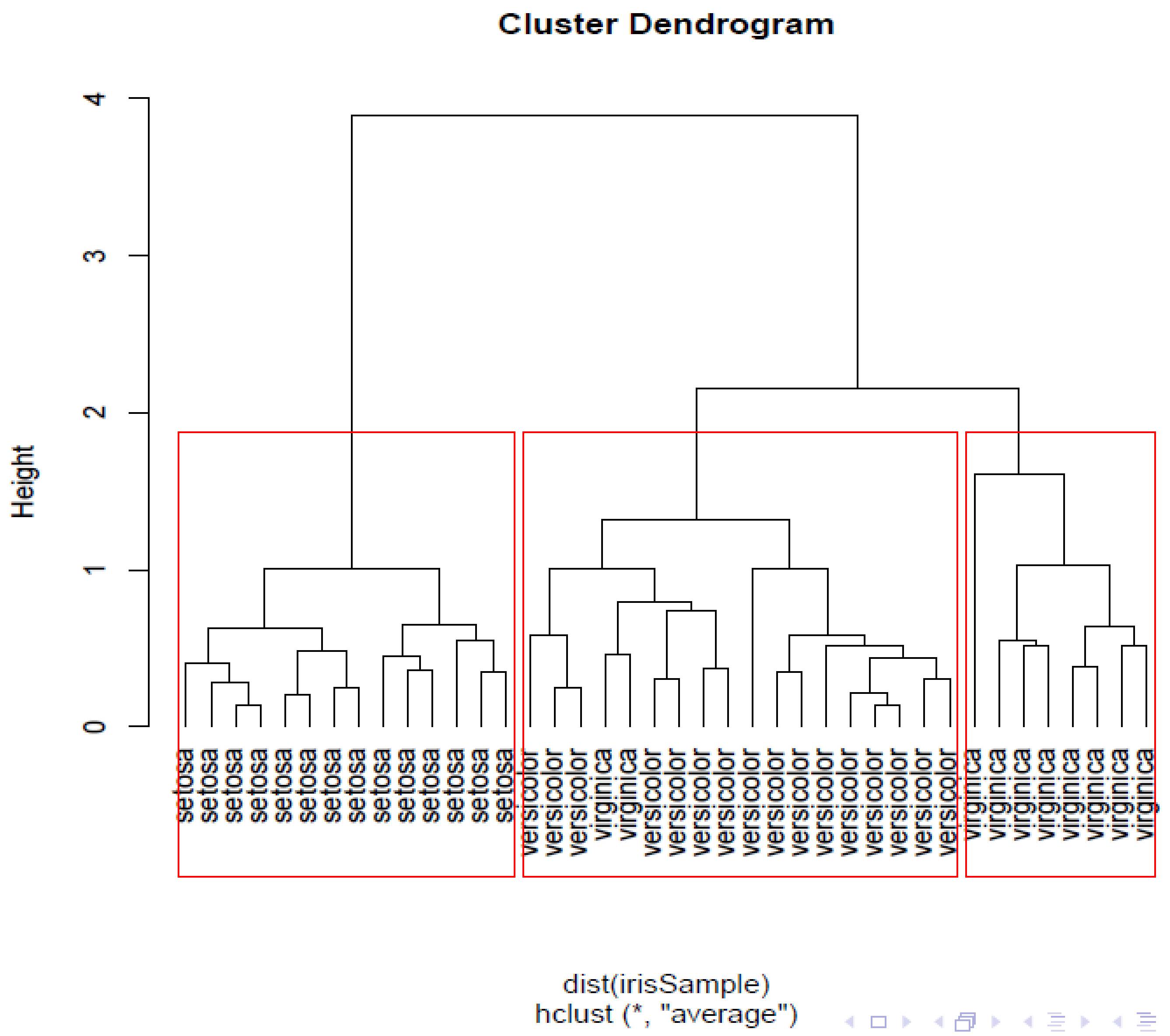
```
layout(matrix(c(1, 2), 1, 2))  # 2 graphs per page
plot(pamk.result$pamobject)
```



```
layout(matrix(1))  # change back to one graph per page
```

HIERARCHICAL CLUSTERING OF THE IRIS DATA

```r
set.seed(2835)
# draw a sample of 40 records from the iris data, so that the
# clustering plot will not be over crowded
idx <- sample(1:dim(iris)[1], 40)
irisSample <- iris[idx, ]
# remove class label
irisSample$Species <- NULL
# hierarchical clustering
hc <- hclust(dist(irisSample), method = "ave")
# plot clusters
plot(hc, hang = -1, labels = iris$Species[idx])
# cut tree into 3 clusters
rect.hclust(hc, k = 3)
# get cluster IDs
groups <- cutree(hc, k = 3)
```


Cluster Dendrogram
Height
0
1
2
3
4
setosa
versicolor
virginica
dist(irisSample)
hclust (*, "average")

TEXT MINING

TEXT MINING WITH R

Text mining: tm

Topic modelling: topicmodels, lda

Word cloud: wordcloud

Twitter data access: twitteR

TEXT MINING

unstructured text data

text categorization

text clustering

entity extraction

sentiment analysis

document summarization

. . .

TEXT MINING OF TWITTER DATA WITH R

1. extract data from Twitter
2. clean extracted data and build a document-term matrix
3. find frequent words and associations
4. create a word cloud to visualize important words
5. text clustering
6. topic modelling

RETRIEVE TWEETS

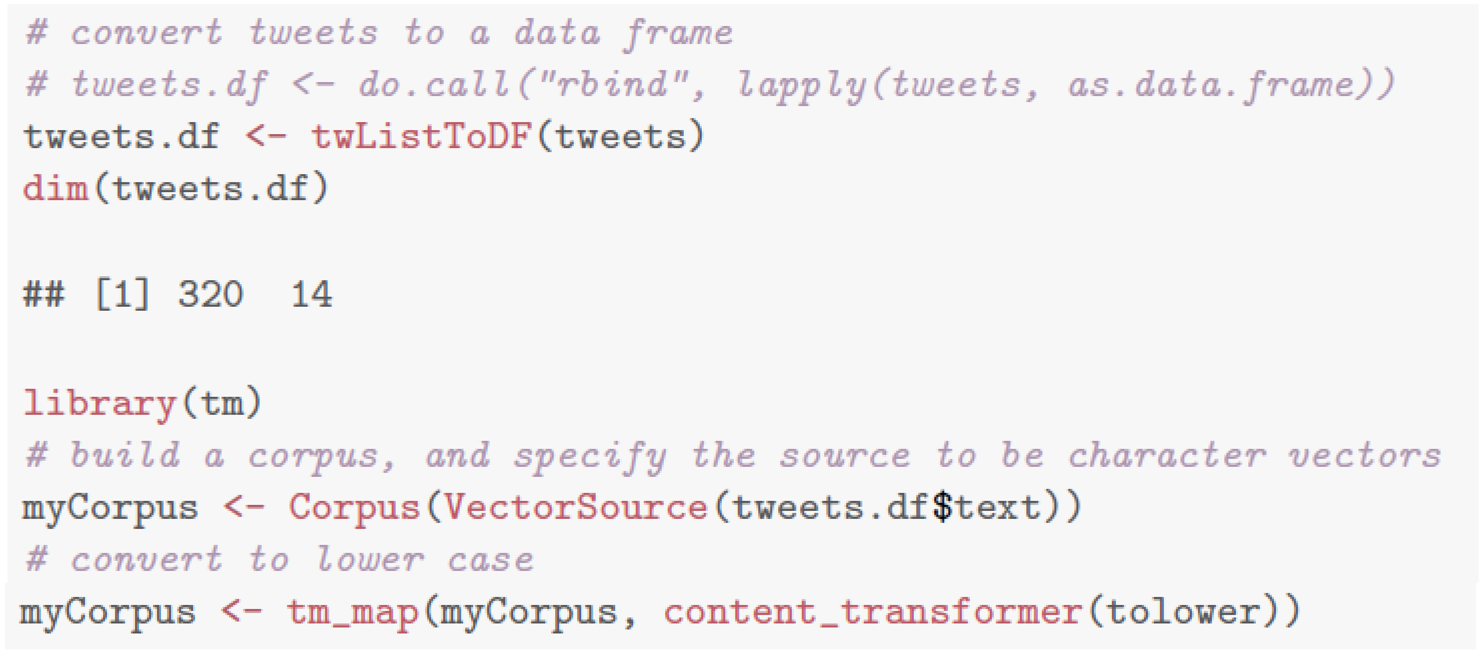
Retrieve recent tweets by @RDataMining

```
## Option 1: retrieve tweets from Twitter
library(twitteR)
tweets <- userTimeline("RDataMining", n = 3200)
```

```
## Option 2: download @RDataMining tweets from RDataMining.com
url <- "http://www.rdatamining.com/data/rdmTweets.RData"
download.file(url, destfile = "./data/rdmTweets.RData")
```

```
## load tweets into R
load(file = "./data/rdmTweets.RData")
```

TEXT CLEANING



```r
# convert tweets to a data frame
# tweets.df <- do.call("rbind", lapply(tweets, as.data.frame))
tweets.df <- twListToDF(tweets)
dim(tweets.df)

## [1] 320  14

library(tm)
# build a corpus, and specify the source to be character vectors
myCorpus <- Corpus(VectorSource(tweets.df$text))
# convert to lower case
myCorpus <- tm_map(myCorpus, content_transformer(tolower))
```

```r
# remove punctuation
myCorpus <- tm_map(myCorpus, removePunctuation)
# remove numbers
myCorpus <- tm_map(myCorpus, removeNumbers)
# remove URLs
removeURL <- function(x) gsub("http[[:alnum:]]*", "", x)
myCorpus <- tm_map(myCorpus, removeURL)
# add two extra stop words: 'available' and 'via'
myStopwords <- c(stopwords("english"), "available", "via")
# remove 'r' and 'big' from stopwords
myStopwords <- setdiff(myStopwords, c("r", "big"))
# remove stopwords from corpus
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
```

```r
# keep a copy of corpus to use later as a dictionary for stem compl
myCorpusCopy <- myCorpus
# stem words
myCorpus <- tm_map(myCorpus, stemDocument)
```

```r
# inspect the first 5 documents (tweets) inspect(myCorpus[1:5])
# The code below is used for to make text fit for paper width
for (i in 1:5) {
    cat(paste("[[", i, "]] ", sep = ""))
    writeLines(myCorpus[[i]])
}

## [[1]] exampl  call java code  r
##
## [[2]] simul mapreduc  r  big data analysi use flight data  ...
## [[3]] job opportun senior analyst  big data  wesfarm indust...
## [[4]] clavin  open sourc softwar packag  document geotag  g...
## [[5]]  onlin book  natur languag process  python
```

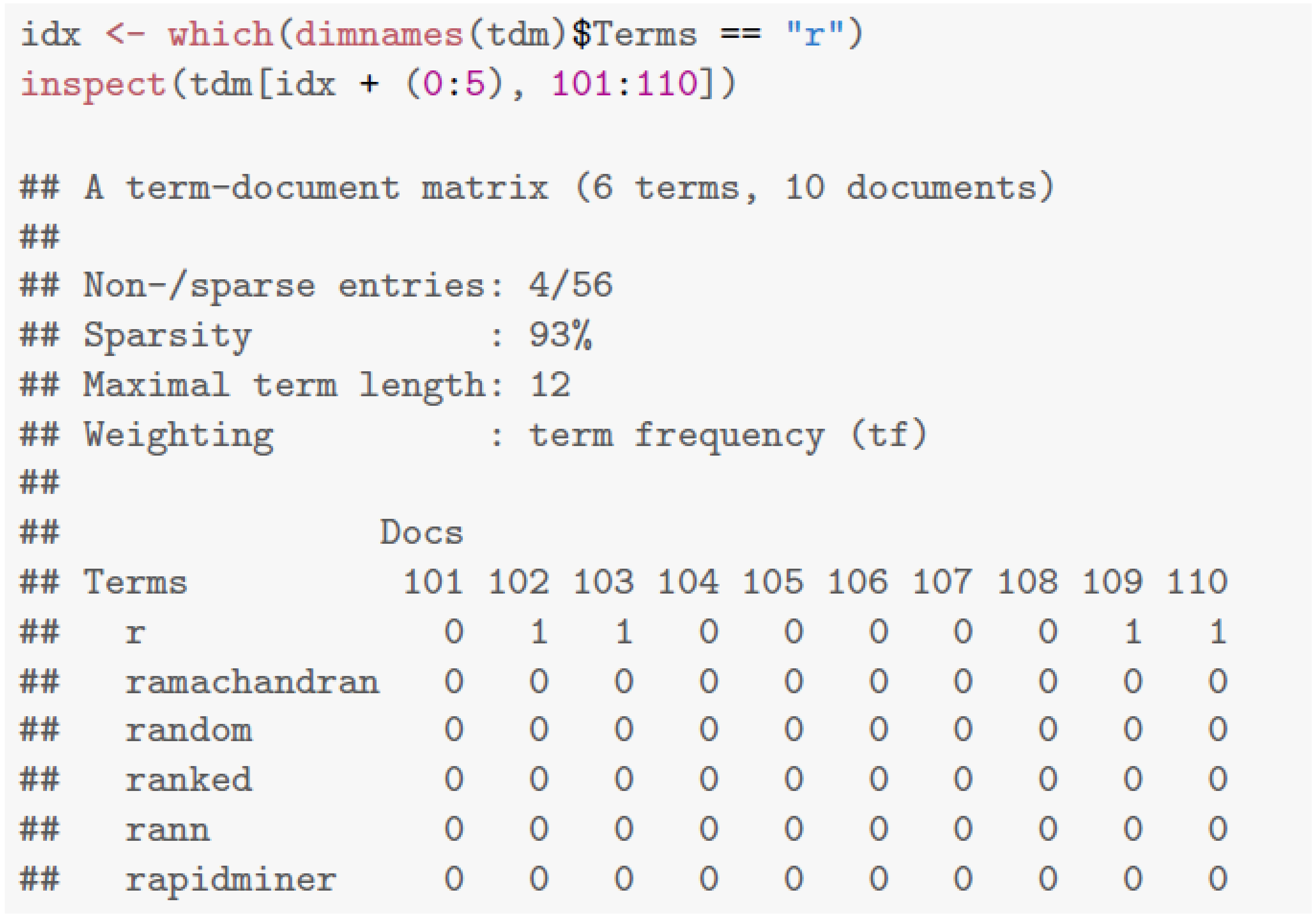
```r
# stem completion
myCorpus <- tm_map(myCorpus, stemCompletion,
                   dictionary = myCorpusCopy)
```

```
## [[1]] examples call java code r
## [[2]] simulating mapreduce r big data analysis used flights...
## [[3]] job opportunity senior analyst big data wesfarmers in...
## [[4]] clavin open source software package document geotaggi...
## [[5]] online book natural language processing python
```

```r
# count frequency of "mining"
miningCases <- tm_map(myCorpusCopy, grep, pattern = "\\<mining")
sum(unlist(miningCases))

## [1] 82

# count frequency of "miners"
minerCases <- tm_map(myCorpusCopy, grep, pattern = "\\<miners")
sum(unlist(minerCases))

## [1] 4

# replace "miners" with "mining"
myCorpus <- tm_map(myCorpus, gsub, pattern = "miners",
                   replacement = "mining")
```

```
tdm <- TermDocumentMatrix(myCorpus,
                               control = list(wordLengths = c(1, Inf)))
tdm

## A term-document matrix (790 terms, 320 documents)
##
## Non-/sparse entries: 2449/250351
## Sparsity           : 99%
## Maximal term length: 27
## Weighting          : term frequency (tf)
```

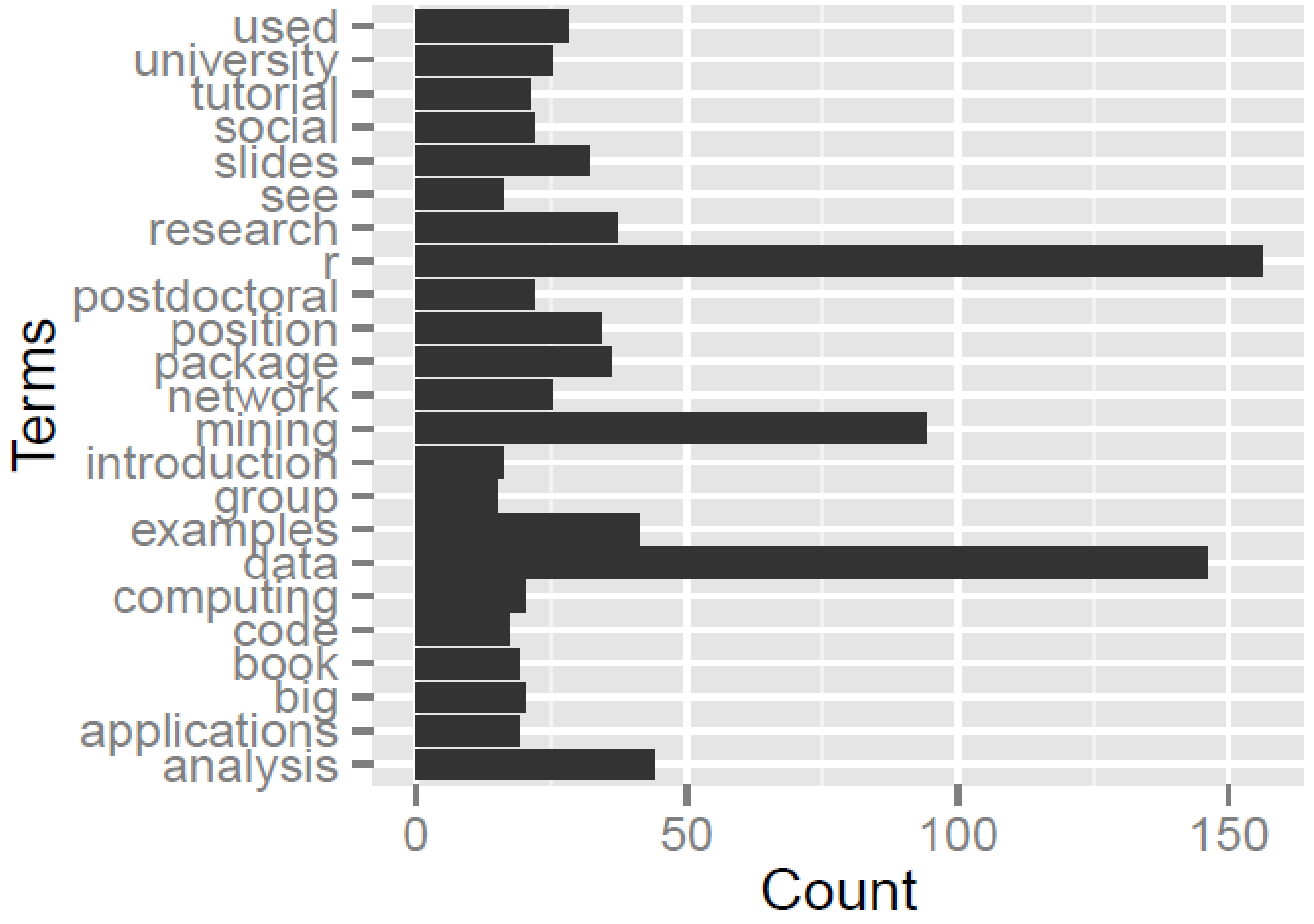
FREQUENT WORDS AND ASSOCIATIONS



```
idx <- which(dimnames(tdm)$Terms == "r")
inspect(tdm[idx + (0:5), 101:110])

## A term-document matrix (6 terms, 10 documents)
##
## Non-/sparse entries: 4/56
## Sparsity           : 93%
## Maximal term length: 12
## Weighting          : term frequency (tf)
##
##              Docs
## Terms          101 102 103 104 105 106 107 108 109 110
##   r              0   1   1   0   0   0   0   0   1   1
##   ramachandran   0   0   0   0   0   0   0   0   0   0
##   random         0   0   0   0   0   0   0   0   0   0
##   ranked         0   0   0   0   0   0   0   0   0   0
##   rann           0   0   0   0   0   0   0   0   0   0
##   rapidminer     0   0   0   0   0   0   0   0   0   0
```

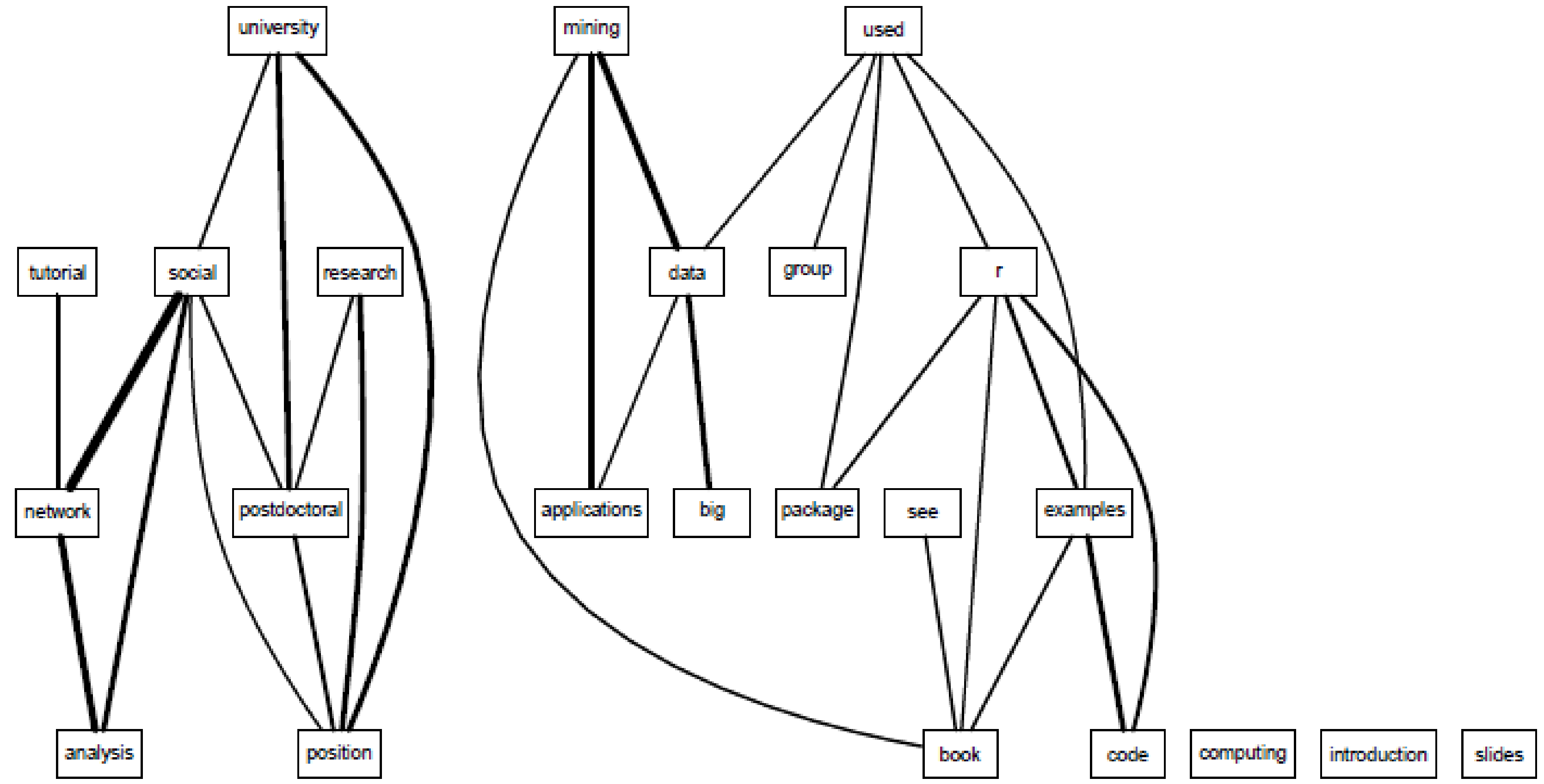
```r
# inspect frequent words
(freq.terms <- findFreqTerms(tdm, lowfreq = 15))

##  [1] "analysis"     "applications" "big"          "book"
##  [5] "code"         "computing"    "data"         "examples"
##  [9] "group"        "introduction" "mining"       "network"
## [13] "package"      "position"     "postdoctoral" "r"
## [17] "research"     "see"          "slides"       "social"
## [21] "tutorial"     "university"   "used"

term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq >= 15)
df <- data.frame(term = names(term.freq), freq = term.freq)
```

```r
library(ggplot2)
ggplot(df, aes(x = term, y = freq)) + geom_bar(stat = "identity") +
    xlab("Terms") + ylab("Count") + coord_flip()
```



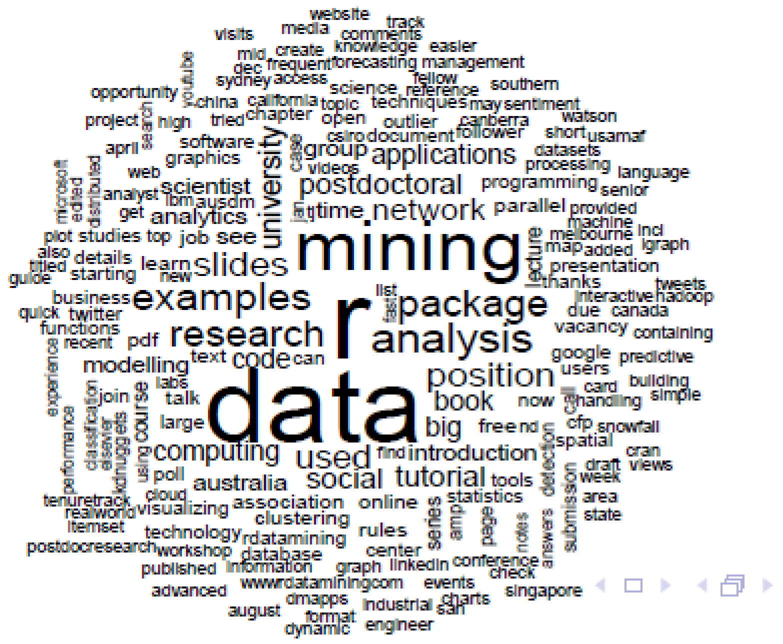
```
# which words are associated with 'r'?
findAssocs(tdm, "r", 0.2)

##              r
## examples 0.32
## code     0.29
## package  0.20

# which words are associated with 'mining'?
findAssocs(tdm, "mining", 0.25)

##                 mining
## data              0.47
## mahout            0.30
## recommendation    0.30
## sets              0.30
## supports          0.30
## frequent          0.26
## itemset           0.26
```

```r
library(graph)
library(Rgraphviz)
plot(tdm, term = freq.terms, corThreshold = 0.12, weighting = T)
```



WORD CLOUD

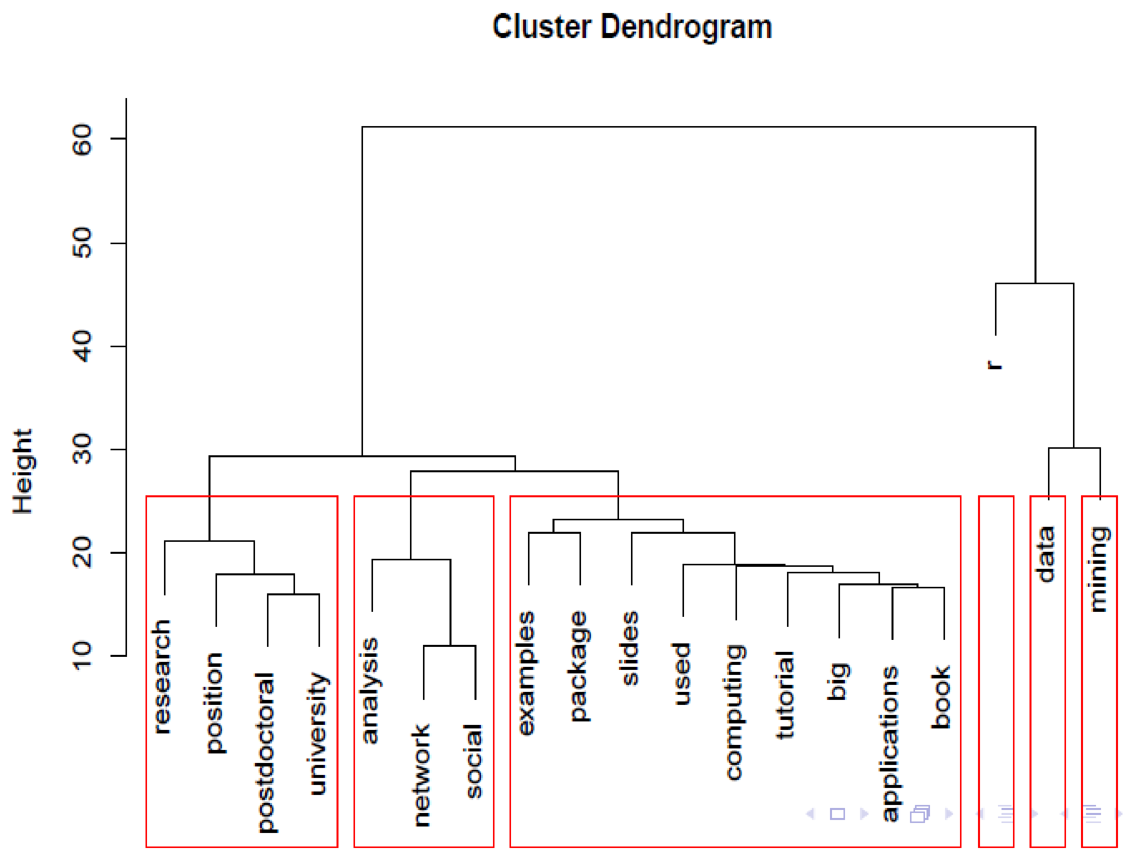
```r
library(wordcloud)
m <- as.matrix(tdm)
# calculate the frequency of words and sort it by frequency
word.freq <- sort(rowSums(m), decreasing = T)
wordcloud(words = names(word.freq), freq = word.freq, min.freq = 3,
    random.order = F)
```



CLUSTERING

```
# remove sparse terms
tdm2 <- removeSparseTerms(tdm, sparse = 0.95)
m2 <- as.matrix(tdm2)
# cluster terms
distMatrix <- dist(scale(m2))
fit <- hclust(distMatrix, method = "ward")
```

```
plot(fit)
rect.hclust(fit, k = 6)  # cut tree into 6 clusters
```

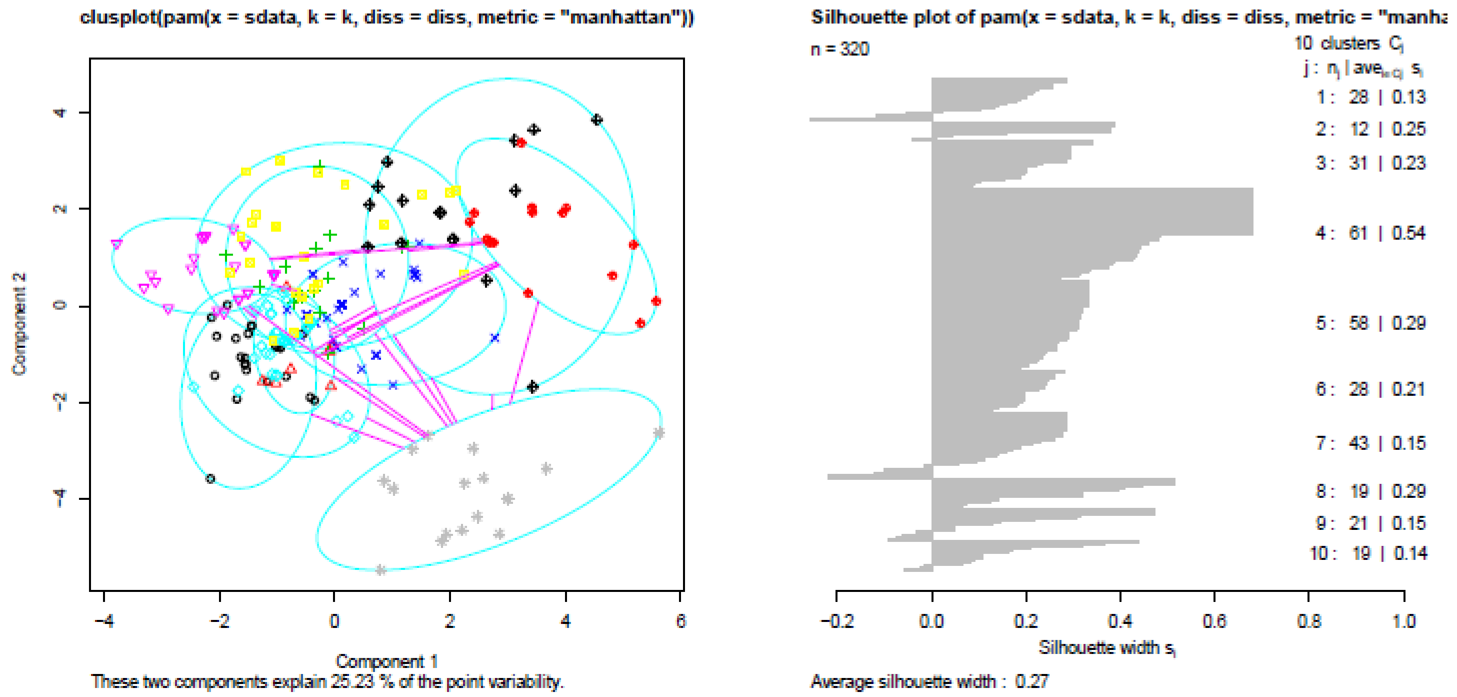


```r
m3 <- t(m2)  # transpose the matrix to cluster documents (tweets)
set.seed(122)  # set a fixed random seed
k <- 6  # number of clusters
kmeansResult <- kmeans(m3, k)
round(kmeansResult$centers, digits = 3)  # cluster centers

##   analysis applications   big  book computing  data examples
## 1    0.147        0.088 0.147 0.015     0.059 1.015    0.088
## 2    0.028        0.167 0.167 0.250     0.028 1.556    0.194
## 3    0.810        0.000 0.000 0.000     0.000 0.048    0.095
## 4    0.080        0.036 0.007 0.058     0.087 0.000    0.181
## 5    0.000        0.000 0.000 0.067     0.067 0.333    0.067
## 6    0.119        0.048 0.071 0.000     0.048 0.357    0.000
##   mining network package position postdoctoral     r research
## 1  0.338   0.015   0.015    0.059        0.074 0.235    0.074
## 2  1.056   0.000   0.222    0.000        0.000 1.000    0.028
## 3  0.048   1.000   0.095    0.143        0.095 0.286    0.048
## 4  0.065   0.022   0.174    0.000        0.007 0.703    0.000
## 5  1.200   0.000   0.000    0.000        0.067 0.067    0.000
## 6  0.119   0.000   0.024    0.643        0.310 0.000    0.714
##   slides social tutorial university  used
## 1  0.074  0.000    0.015      0.015 0.029
## 2  0.056  0.000    0.000      0.000 0.250
## 3  0.095  0.762    0.190      0.000 0.095
```

```
for (i in 1:k) {
    cat(paste("cluster ", i, ":  ", sep = ""))
    s <- sort(kmeansResult$centers[i, ], decreasing = T)
    cat(names(s)[1:5], "\n")
    # print the tweets of every cluster
    # print(tweets[which(kmeansResult£cluster==i)])
}
```

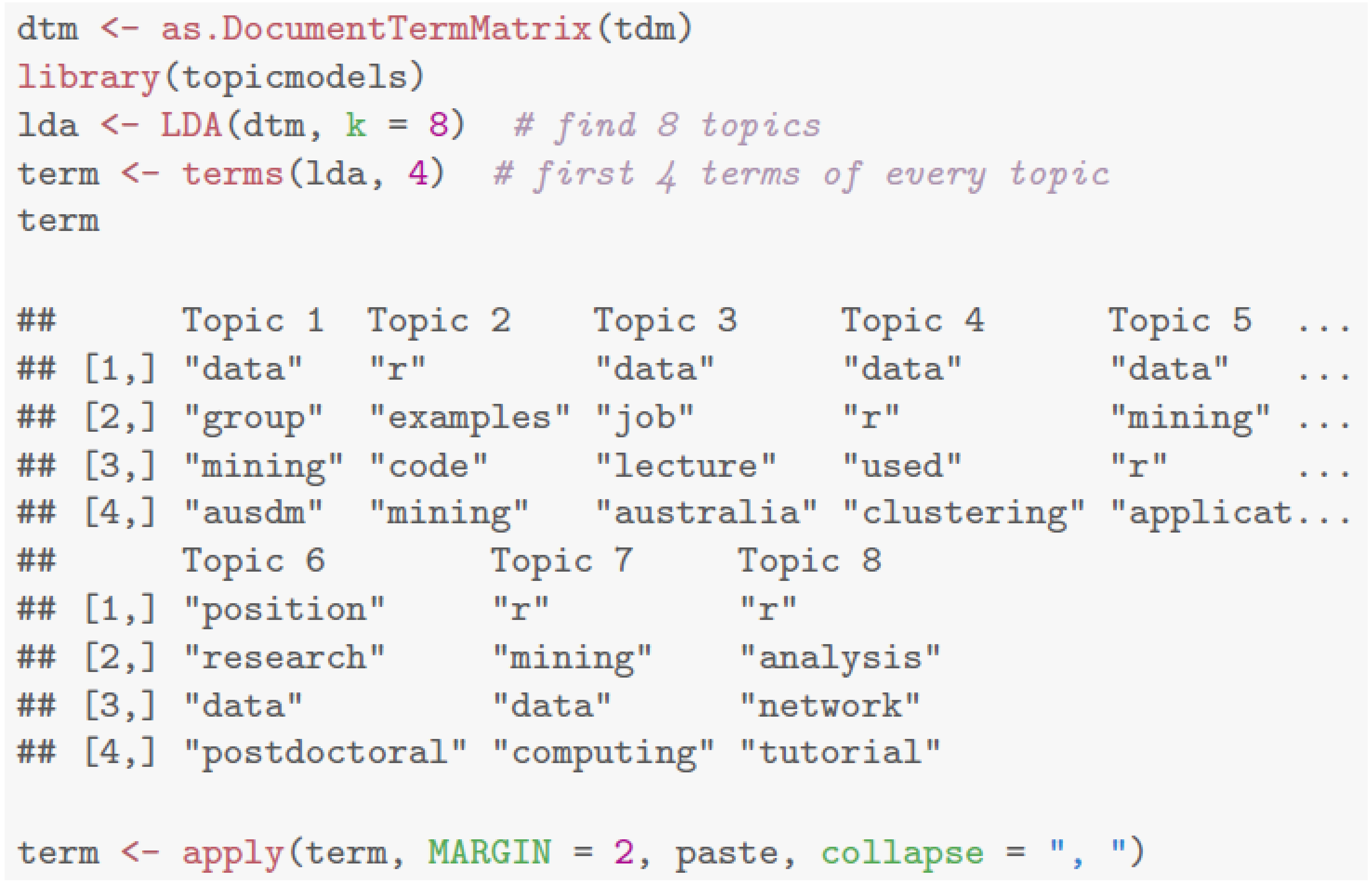
```
## cluster 1:  data mining r analysis big
## cluster 2:  data mining r book used
## cluster 3:  network analysis social r tutorial
## cluster 4:  r examples package slides used
## cluster 5:  mining tutorial slides data book
## cluster 6:  research position university data postdoctoral
```

```
# plot clustering result
layout(matrix(c(1, 2), 1, 2))  # set to two graphs per page
plot(pamResult, col.p = pamResult$clustering)
```



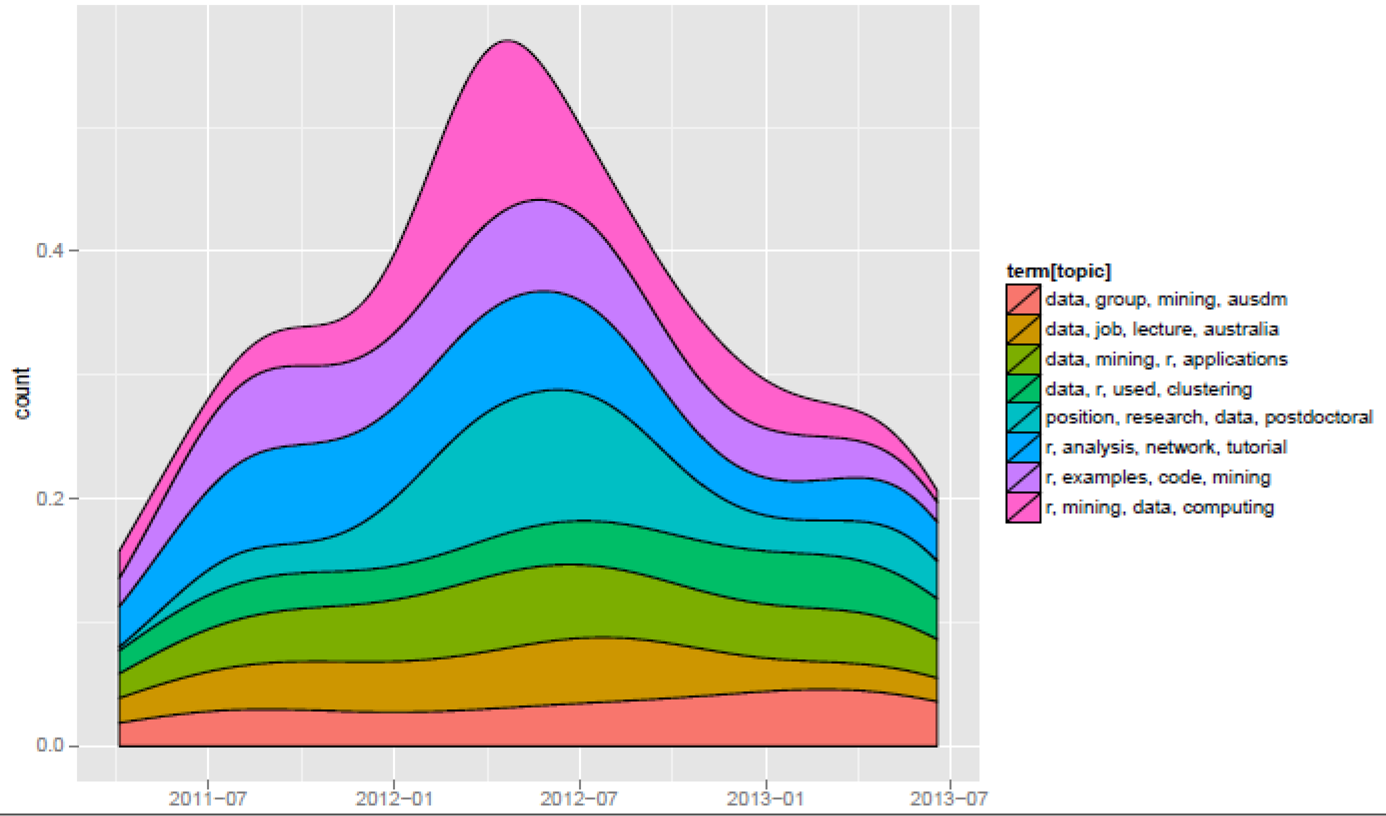
```
layout(matrix(1))  # change back to one graph per page
```

TOPIC MODELLING



```
dtm <- as.DocumentTermMatrix(tdm)
library(topicmodels)
lda <- LDA(dtm, k = 8)  # find 8 topics
term <- terms(lda, 4)  # first 4 terms of every topic
term

##      Topic 1  Topic 2    Topic 3     Topic 4      Topic 5  ...
## [1,] "data"   "r"        "data"      "data"       "data"   ...
## [2,] "group"  "examples" "job"       "r"          "mining" ...
## [3,] "mining" "code"     "lecture"   "used"       "r"      ...
## [4,] "ausdm"  "mining"   "australia" "clustering" "applicat...
##      Topic 6        Topic 7     Topic 8
## [1,] "position"     "r"         "r"
## [2,] "research"     "mining"    "analysis"
## [3,] "data"         "data"      "network"
## [4,] "postdoctoral" "computing" "tutorial"

term <- apply(term, MARGIN = 2, paste, collapse = ", ")
```

```r
# first topic identified for every document (tweet)
topic <- topics(lda, 1)
topics <- data.frame(date=as.IDate(tweets.df$created), topic)
qplot(date, ..count.., data=topics, geom="density",
      fill=term[topic], position="stack")
```


count
term[topic]
data, group, mining, ausdm
data, job, lecture, australia
data, mining, r, applications
data, r, used, clustering
position, research, data, postdoctoral
r, analysis, network, tutorial
r, examples, code, mining
r, mining, data, computing