

Basic Elements of Computational Statistics in R

Wolfgang Härdle

Ostap Okhrin

Yarema Okhrin

Chair of Statistics and Econometrics

Humboldt-Universität zu Berlin

Chair of Statistics and Econometrics

Technische Universität Dresden

Chair of Statistics

Universität Augsburg

<http://www.wiwi.hu-berlin.de/>

<http://tu-dresden.de>

<http://www.wiwi.uni-augsburg.de>



The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

The Basics of R

Introduction

R on your computer

First steps

Fundamental Operations

Vectors and Matrices

Data frames and lists

Reading and Writing Data

Programming in R

History of R language

- *S* - the statistical programming language of Bell Laboratories
(Becker and Chamber, 1984)
- R - implementation of *S*
- R was realized in 1992 by R. Ihaka and R. Gentleman
(University of Auckland NZ)
- Name R - from the first names of the authors



History of R language

R is a multi-platform statistical package widely used in many scientific fields from mathematics to medicine and biology. The source-code of R is freely available since 1995.

Possibilities of R :

- Handling and storage of data
- Statistical analysis
- Importing packages
- ...



The Basics of R

Introduction

R on your computer

First steps

Fundamental Operations

Vectors and Matrices

Data frames and lists

Reading and Writing Data

Programming in R

Installing and updating R

- R is a free software which can be downloaded from page:
<http://cran.r-project.org/mirrors.html>
- A full installation guide for all systems is available at
<http://cran.r-project.org/doc/manuals/R-admin.html>

Precompiled binary distributions:

- Installing R under Unix:
<http://mirrors.softliste.de/cran/>
- Installing R under Windows:
<http://mirrors.softliste.de/cran/bin/windows/base/>
- Installing R under Mac:
<http://cran.r-project.org/bin/macosx/>



Installing and updating R

Updating:

- Uninstall the previous version of R
- Install the new version
- Copy the old installed packages to the library folder of the new installation.
- Run command

```
1 > update.packages(checkBuilt = TRUE, ask = FALSE)
```



Packages

Packages enable us to get more functions or data sets for current session.

Installing packages:

- UNIX or Mac OS - run command

```
1 > install.packages("package.name")
```

- Windows

In the RGui interface use menu *Packages* -> *Install Packages*



Using installed packages

- To see the installed libraries, run

```
1 > library()
```

- To see the installed package, run one of the following commands:

```
1 > library(splines) # package name = "splines"
2 > require(splines)
```

- To *detach* a loaded package, run

```
1 > detach(package:splines)
```



The Basics of R

Introduction

R on your computer

First steps

Fundamental Operations

Vectors and Matrices

Data frames and lists

Reading and Writing Data

Programming in R

First steps

Printing "Hello world!" and getting help

```
1 > print("Hello world!")      # print "Hello world!"  
2 > help()                   # global help menu  
3 > help(sin)                # help for "sin"  
4 > ?sin                     # help for "sin"  
5 > library(help = splines)  # help about "splines"  
6 > example(print)           # find examples for "print"  
7 > help.search("table")     # help for "table"
```



Working Place

The **working directory** is a current directory where all pictures and tables are saved by default, and also from which all the data might be read.

```
1 > getwd()                      # get working directory
2 > setwd("your/own/path")        # set working directory
3 > rm(t1, t2)                   # erase variables t1 and t2
4 > rm(list = ls())              # clean memory
5 > ls()                         # get the names of all defined objects
6 > save.image()                 # save the workspace
7 > savehistory()                # save the history of the commands
```



Working Place

```
1 > load(".Rdata") # load saved workspace  
2 > loadhistory() # load saved history
```

- ☐ A recommendable way of writing programs is to split them into different modules in order not to mess up the main file.

```
1 > source("my.r") # accept the input from the file  
2 > sink("my.txt") # write the output to the file  
3 > sink() # finish writing the output  
4 > quit() # quit R  
5 > q() # quit R
```



The Basics of R

Introduction

R on your computer

First steps

Fundamental Operations

Vectors and Matrices

Data frames and lists

Reading and Writing Data

Programming in R

R as calculator

```
1 > 1+2    # Addition  
2 [1] 3  
3 > 2-1    # Subtraction  
4 [1] 1  
5 > 1*2    # Multiplication  
6 [1] 2  
7 > 1/2    # Division  
8 [1] 0.5  
9 > 3^2    # Power  
10 [1] 9  
11 > 5%/%2  # Integer division  
12 [1] 2  
13 > 5%2  # Modulo division  
14 [1] 1
```



R as calculator

```
1 > a = pi + 0.5 # a=3.641593
2 > # largest integer number that is smaller than a
3 > floor(a)
4 [1] 3
5 > # smallest integer number that is larger than a
6 > ceiling(a)
7 [1] 4
8 > trunc(a)    # truncate value
9 [1] 3
10 > round(a)   # round value
11 [1] 4
```



Basic functions

Function name	Example	Result
square root	<code>sqrt(2)</code>	1.414214
sine	<code>sin(pi)</code>	1.224606e-16
cosine	<code>cos(pi)</code>	-1
tangent	<code>tan(pi/4)</code>	1
arcsine	<code>asin(pi/4)</code>	0.903339
arccosine	<code>acos(0)</code>	1.570796
arctangent	<code>atan(1)</code>	0.785398
arctan(y/x)	<code>atan2(1, 2)</code>	0.463647
hyperbolic sine	<code>sinh(1)</code>	1.175201
hyperbolic cosine	<code>cosh(0)</code>	1
hyperbolic tangent	<code>tanh(pi)</code>	0.9962721
exponential	<code>exp(1)</code>	2.718282
logarithm	<code>log(1)</code>	0

Table 1: Basic functions



Comparison relations

```
1 > 5 < 5          # smaller
2 [1] FALSE
3 > 3 <= 4         # smaller or equal
4 [1] TRUE
5 > 7 > 2          # greater
6 [1] TRUE
7 > 5 >= 5         # greater or equal
8 [1] TRUE
9 > 2 != 1          # unequal
10 [1] TRUE
11 > pi == acos(-1) # logical equal
12 [1] TRUE
```



Assigning variables

- Different ways to assign variables to formal letters.

```
1 > a = pi + 0.5 # a = 3.641593
2 > b = a          # b = 3.641593
3 > d = 2^{1/2}    # d = 1.414214
4 > f <- d        # f = 1.414214
5 > d -> g        # g = 1.414214
```

- Variable names must not begin with a digit or a period followed by a digit.
- Additionally, names should not begin with a dot as this is common only for system variables.



Working with variables

- Different types of variables in R:
 - ▶ numeric (or double) - vector of real numbers
 - ▶ logical - vector of TRUE and FALSE
 - ▶ character - vector of strings
 - ▶ integer - vector of integers
 - ▶ matrix - matrix
 - ▶ list - vector of R objects
- To know the type of an object run

```
1 > typeof(object name)
```



The Basics of R

Introduction

R on your computer

First steps

Fundamental Operations

Vectors and Matrices

Data frames and lists

Reading and Writing Data

Programming in R

Basic manipulations with vectors

```
1 > v = c(1, pi, sqrt(2)) # create vector v
2 > v[i]          # i-th element of v
3 > length(v)    # length of v
4 > q = v + 3    # add 3 to every element of v
5 > p = v ^ (-1)  # elementwise inverse vector
```



Different ways to construct vectors

```
1 > a = c(1, 2, 3)                      # a = 1 2 3
2 > b=array(1:3)                         # b = 1 2 3
3 > c = 1:3                               # c = 1 2 3
4 > d=seq(1, 3)                           # d = 1 2 3
5 > e=seq(1, 3, by = 2)                   # e = 1 3
6 > f=seq(1, 4, length.out = 3) # f = 1.0 2.5 4.0
7 > v=1:2
8 > rep(v, 2)      # the vector v two times
9 [1] 1 2 1 2
10 > rep(v, c(3, 1)) # 1st value 3, 2nd 1 times
11 [1] 1 1 1 2
12 > rep(v, each = 2) # each value two times
13 [1] 1 1 2 2
```



Comparison relations for vectors

```
1 > v = c(1, sqrt(2), pi)
2 > v > 1           # create boolean vector,
3 [1] FALSE TRUE TRUE
4 > v[1] != 1       # logical NOT,
5 [1] FALSE
6 > (v[1]==1) & (v[3]==pi) # logical AND,
7 [1] TRUE
8 > v[1]==0 | 1     # logical OR,
9 [1] TRUE
10 > (v != 1) & (v>0)
11 [1] FALSE TRUE TRUE
12 > v[c(1, 3)]    # select elements 1,3 of v
13 > v[-1]          # eliminate element 1 in v
```



Some manipulations with vectors

```
1 > v = c(4, 2, 10)
2 > min(v)          # minimal value of v
3 [1] 2
4 > max(v)          # maximal value of v
5 [1] 10
6 > range(v)        # extremal values of v
7 [1] 2 10
8 > sort(v)         # sort values of v
9 [1] 2 4 10
```



Some manipulations with vectors

```
1 > sum(v)           # sum of all elements in v
2 [1] 16
3 > rev(v)          # elements of v in the reversed order
4 [1] 10  2  4
5 > which(v == 2)  # returning index of 2
6 [1] 2
7 > t(v)            # transpose of v
8     [,1] [,2] [,3]
9 [1,]    4    2   10
```



Some manipulations with vectors

```
1 > x = c(4, 2, 5, 7, 1, 9, 0, 3)
2 # positions of elements in the sorted vector
3 > rank(x)
4 [1] 5 3 6 7 2 8 1 4
5 # positions of elements in increasing order
6 > order(x)
7 [1] 7 5 2 8 1 3 4 6
```

Replacing some values in a vector

```
1 > x = 1:8                      # x = 1 2 3 4 5 6 7 8
2 > y = replace(x, x<3, 12)      # y = 12 12 3 4 5 6 7 8
3 > z = replace(x, 6, 12)        # z = 1 2 3 4 5 12 7 8
```



Creating a matrix

```
1 > matrix(1:12, nrow = 3) # create a matrix
2   [,1] [,2] [,3] [,4]
3 [1,]    1    4    7   10
4 [2,]    2    5    8   11
5 [3,]    3    6    9   12
6 > diag(2)    # identity matrix of size 2 * 2
7 > diag(3,2)  # diagonal matrix
8   [,1] [,2]
9 [1,]    3    0
10 [2,]    0    3
11 > matrix(0, 5, 4) # matrix of zeros of size 5 * 4
```



Creating a matrix

```
1 > diag(2, 3, 4)      # rectangular diagonal matrix
2   [,1] [,2] [,3] [,4]
3 [1,]  2     0     0     0
4 [2,]  0     2     0     0
5 [3,]  0     0     2     0
6 > m = 1:6            # m = 1 2 3 4 5 6
7 > dim(m) = c(2, 3) # matrix with values of m
8 > m
9   [,1] [,2] [,3]
10 [1,]  1     3     5
11 [2,]  2     4     6
```



Basic manipulations with matrices

```
1 > x = 1:6
2 > y = LETTERS[1:6] # "A" "B" "C" "D" "E" "F"
3 > rbind(x,y) # joins vectors rowwise into a matrix
4 [,1] [,2] [,3] [,4] [,5] [,6]
5 x "1" "2" "3" "4" "5" "6"
6 y "A" "B" "C" "D" "E" "F"
7 > cbind(x,y) # joins vectors columnwise into a matrix
8 > m = matrix(1:6,2)
9 > col(m)      # column indices of all elements
10    [,1] [,2] [,3]
11 [1,] 1     2     3
12 [2,] 1     2     3
13 > row(m)      # row indices of all elements
```



Extracting a submatrix

```
1 > y = matrix (1:16, 4 ,4)
2 > y[2, ]          # 2 6 10 14
3 > y[, 2]         # 5 6 7 8
4 > y[2]           # 2
5 > y[1:2, 3:4]   # extracts a submatrix
6      [,3] [,4]
7 [1,]  9     13
8 [2,] 10    14
```



Assigning dimension names

```
1 > A = matrix(1:20, 4, 5)
2 > dimnames(A) = list(letters[10:13], letters[1:5])
3   a b c d e
4 j 1 5 9 13 17
5 k 2 6 10 14 18
6 l 3 7 11 15 19
7 m 4 8 12 16 20
8 > A[2, 2]      # 6
9 > A["k", "b"]  # 6
10 > # alphabetic letters 1-5 as col.names
11 > colnames(A) = letters[1:5]
12 > # letters 10-13 as row names
13 > rownames(A) = letters[10:13]
```



Elementwise matrix operations

```
1 > A = matrix(1:20, 4, 5)
2 > sum(A)      # sum of all elements
3 [1] 210
4 > prod(A)     # product of all elements
5 [1] 2.432902e+18
6 > colSums(A)  # sums by columns
7 [1] 10 26 42 58 74
8 > rowSums(A)  # sums by rows
9 [1] 45 50 55 60
```



The Basics of R

Introduction

R on your computer

First steps

Fundamental Operations

Vectors and Matrices

Data frames and lists

Reading and Writing Data

Programming in R

Creating a data frame

- A data frame is a very useful object, which enables to collect sample-data of different types (numeric, logical etc.).
- Let us construct a data frame.

```
1 > cities      = c("Berlin", "New York",
2 +                  "Paris", "Tokyo")
3 > area        = c(892, 1214, 105, 2188)
4 > population  = c(3.4, 8.1, 2.1, 12.9)
5 > continent   = factor(c("Europe", "North America",
6 +                           "Europe", "Asia"))
7 > myframe     = data.frame(cities, area,
8 +                           population, continent)
```



Creating a data frame

- Let us take a look at the data frame.

```
1 > rownames(myframe) =c("Berlin", "New York",
2 +                      "Paris", "Tokyo")
3 > colnames(myframe) =c("City", "Area",
4 +                      "Pop.", "Continent")
5 > myframe
6
7   City Area Pop.      Continent
8 Berlin    Berlin  892  3.4        Europe
9 New York New York 1214  8.1 North America
10 Paris     Paris   105  2.1        Europe
11 Tokyo     Tokyo  2188 12.9       Asia
12 > # check if object is dataframe
13 > is.data.frame(myframe)
14 [1] TRUE
```



Factor variable

```
1 > e = c(2, 0, 2, 0)
2 > f = factor(e, level = 0:2)
3 > # levels of the variable f are "sorted" in decreasing
   order of the proximity to the sea
4 > levels(f) = c("Coastal", "Middle", "Inland")
5 > f
6 [1] Inland Coastal Inland Coastal
7 Levels: Coastal Middle Inland
8 > class(f)      # class is "factor"
9 [1] "factor"
10 > as.numeric(f) # numerical coding of the levels
11 [1] 3 1 3 1
```



Adding a new column to data frames

```
1 > f = c("Inland", "Coastal", "Inland", "Coastal")
2 > myframe = data.frame(myframe, f) # adds a new column
3 > colnames(myframe)[5] = "Sea.Env."
4 > # alternative way
5 Language.Spoken = c("German", "English",
6 +                  "French", "Japanese")
7 > myframe = cbind(myframe,
8                   "Language.Spoken"= Language.Spoken)
```



Addressing a particular column to data frames

```
1 > myframe[3]      # returns a data frame  
2           Pop.  
3 Berlin      3.4  
4 New York    8.1  
5 Paris       2.1  
6 Tokyo       12.9  
7 > a = myframe$Pop.      # a = 3.4 8.1 2.1 12.9  
8 > b = myframe[,3]       # b = 3.4 8.1 2.1 12.9  
9 > c = myframe[, "Pop ."] # c = 3.4 8.1 2.1 12.9  
10 > d = myframe[3,2]      # d = 105
```



Some useful functions for data frames

The attach() function enables to access objects in the database by simply giving their names.

```
1 > attach(myframe) # myframe added to search path
2 > detach(myframe) # removed from search path
3 > # editing a data frame through interactive tables
4 > edit(myframe)   # the changes of data are not stored
5 > fix(myframe)   # the changes in the data are stored
```



A data frame as database

There is a possibility to select subsets of data frames using the logical operators.

```
1 > myframe[(myframe$Language.Spoken == "French") |  
2 +         (myframe$Pop. > 10), -1]  
3  
4   Area Pop. Continent Prox.Sea Language.Spoken  
5 Paris 105    2.1      Europe   Inland       French  
5 Tokyo 2188   12.9     Asia     Coastal      Japanese  
6 > myframe[,-c(1,3)]  
7  
8   Area          Continent Prox.Sea Language.Spoken  
8 Berlin        892        Europe   Inland       German  
9 New York    1214    North America Coastal      English  
10 Paris        105        Europe   Inland       French  
11 Tokyo        2188       Asia     Coastal      Japanese
```



Conditional selection and transformation

```
1 > # conditional selection
2 > subset(myframe[,-1], Area > 1000)
3   Area Pop.      Continent Prox.Sea Language.Spoken
4 New York 1214 8.1 North America Coastal           English
5 Tokyo     2188 12.9          Asia Coastal           Japanese
6 > # creating a new column using data from myframe
7 > myframe.trans = transform(myframe,
8 +           Density = Pop. * 10^6 / Area)
9 > myframe.trans[,-(1:3)]
10  Continent Prox.Sea Language.Spoken    Density
11 Berlin      Europe Inland           German    3811.659
12 New York   North America Coastal       English    6672.158
13 Paris        Europe Inland           French   20000.000
14 Tokyo        Asia   Coastal           Japanese  5895.795
```



Extracting and sorting data

```
1 > Area.Seasiders = myframe$Area[myframe$Prox.Sea == "Middle"
2 +                                     | myframe$Prox.Sea == "Coastal"]
3 > Area.Seasiders
4 [1] 1214 2188
5 > sum(Area.Seasiders)
6 [1] 3402
7 > myframe[order(myframe$Pop.,partial = myframe$Area),
8 +          -(4:6)]
9             City Area Pop.
10 Paris      Paris 105  2.1
11 Berlin     Berlin 892  3.4
12 New York  New York 1214  8.1
13 Tokyo      Tokyo 2188 12.9
```



Lists

- Lists are very flexible objects which may be considered as a container of different type of data.
- The simplest way to construct a list is through

```
1 > a = c(2, 7)
2 > b = "Hello"
3 > stirling=function(x){x*x}
4 > d = list(example = stirling, a, end = b)
```



Lists

```
1 > d
2 $example
3 function (x)
4 {
5     x * x
6 }
7
8 [[2]]
9 [1] 2 7
10
11 $end
12 [1] "Hello"
```



Constructing a list

```
1 > z = c(stirling, a) # construct a list
2 > typeof(z) # "list"
3 > d$end      # use $ to address elements
4 [1] "Hello"
5 > # transforming a list into a 1-length-element list
6 > unlist(z)
7 [[1]]
8 > function(x){x*x}
9 [[2]]
10 [1] 2
11 [[3]]
12 [1] 7
```



Converting objects

```
1 # returning a list of the splitted objects
2 > split(myframe[,-1], myframe$Continent)
3 $Asia
4     Area Pop. Continent Prox.Sea Language.Spoken
5 Tokyo 2188 12.9      Asia   Coastal           Japanese
6
7 $Europe
8     Area Pop. Continent Prox.Sea Language.Spoken
9 Berlin 892 3.4      Europe  Inland           German
10 Paris 105 2.1      Europe  Inland           French
11
12 $'North America'
13     Area Pop.      Continent Prox.Sea Language.Spoken
14 New York 1214 8.1 North America  Coastal           English
```



The Basics of R

Introduction

R on your computer

First steps

Fundamental Operations

Vectors and Matrices

Data frames and lists

Reading and Writing Data

Programming in R

Writing Data

```
1 > # creating the file from the data frame
2 > write.table(myframe, "mydata.txt")
3 > # the names for columns and rows are not defined
4 > write.table(Orange, "example.txt", col.names=F, row.names=F)
5 > # the tab separation between cells
6 > write.table(Orange, "example2.txt", sep="\t")
7 > # dot as decimal point, comma as separator
8 > write.csv(myframe, "mydata.csv")
9 > # comma as decimal point, semicolon as separator
10 > write.csv2(myframe, "mydata.csv")
```



Reading Data

```
1 > data = read.table("mydata.txt", header = TRUE)
2 > names(data)
3 [1] "City"                  "Area"                   "Pop."
     "Continent"              "Prox.Sea"                "Language.Spoken"
4 > str(data, strict.width = "cut", width = 40)
5 'data.frame':   4 obs. of  6 variables:
6 $ City                    : Factor w/ 4 level"..
7 $ Area                     : int  892 1214 105 ..
8 $ Pop.                     : num  3.4 8.1 2.1 1..
9 $ Continent                : Factor w/ 3 level"..
10 $ Prox.Sea                 : Factor w/ 2 level"..
11 $ Language.Spoken: Factor w/ 4 level"..
```



Reading Data

```
1 > # providing separation between the columns
2 > data = read.table("file name", sep="\t")
3 > # the NA values were implemented by "hello"
4 > data = read.table("file name", na.strings="hello")
5 > # decimal separator: ".", variables separator: ","
6 > data = read.csv("file name")
7 > # decimal separator: ",", variables separator: "."
8 > data = read.csv2("file name")
9 > scan("file name") # read the data from file
```



The Basics of R

Introduction

R on your computer

First steps

Fundamental Operations

Vectors and Matrices

Data frames and lists

Reading and Writing Data

Programming in R

Programming in R

R has a lot of programming possibilities and allows to create powerful routines with

- functions,
- loops,
- conditions,
- packages,
- objects.

Providing a list of all arguments that can be used in the function

```
1 args(function name)
```



Functions

Below a simple function is presented, which returns

$$f(x, a) = \begin{cases} a \sin(x) \\ a \cos(x) \end{cases}$$

as a list with a and x as arguments.

```
1 > myfun4 = function (x, a = 1){r1 = a * sin(x)
2 +                               r2 = a * cos(x)
3 +                               list (r1, r2)}
4 > myfun4 (pi /2)
5 [[1]]
6 [1] 1
7 [[2]]
8 [1] 6.123032e-17
```



Loops and conditions

```
1 > x = 1
2 > if (x == 2){print("x = 2")}
3 > if (x == 2){print("x = 2")}\ else {print("x != 2")}
4 [1] "x != 2"
5 > x = numeric(1)
6 > for (i in 0:9) x[i+1] = i
7 > x
8 [1] 0 1 2 3 4 5 6 7 8 9
9 > i = 0
10 > while (i<15) {x[i+1] = i
11 + i = i+1 }
12 > x
13 [1] 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14
```



Some useful functions

The function `apply()` applies a deterministic function to the rows or to the columns of a matrix. The second argument determines whether it is applied to rows or columns.

```
1 > A = matrix(1:12, 6, 2, byrow = TRUE)
2 > apply(A, 1, mean) # "mean" is applied to rows
3 [1] 1.5 3.5 5.5 7.5 9.5 11.5
4 > apply(A, 2, mean) # "mean" is applied to columns
5 [1] 6 7
6 > a=apply(A, 2, sum)# a = 36 42
7 > b=apply(A, 1, sum)# b = 3 7 11 15 19 23
```



Some useful functions

The functions `lapply()` and `sapply()` could apply some function to each element of the list object.

```
1 # the function "sin" is applied, returning a list
2 > lapply(A[1, ], sin)
3 [[1]]
4 [1] 0.841471
5 [[2]]
6 [1] 0.9092974
7 # the function "sin" is applied, returning a vector
8 > sapply(A[1, ], sin) #
9 [1] 0.8414710 0.9092974
```



Some useful functions

The function `switch(i, expression1, expression2,...)` chooses the i-th expression in the given expression arguments.

```
1 > rootsquare = function (x, type){ switch(type,
2 +                               square = x*x,
3 +                               root = sqrt(x)) }
4 > a = rootsquare (10 ,1)          # a = 100
5 > b = rootsquare (10 ,2)          # b = 3.162278
6 > c = rootsquare (10 ,3)          # c = NULL
7 > d = rootsquare (10 , "square") # d = 100
8 > e = rootsquare (10 , "root")   # e = 3.162278
9 > f = rootsquare (10 , "ROOT")   # f = NULL
```



Some useful functions

Comparing the efficiency of two different commands by the time they need to be computed.

```
1 > x = c(1:500000)
2 > system.time(for(i in 1:500000){x[i] = rnorm(1)})
3 [1] user  system elapsed
4     3.252   0.037   3.329
5 > system.time({x = rnorm(500000)})
6 [1] user  system elapsed
7     0.058   0.000   0.058
```



Date types

```
1 > Sys.time() # current date and time values  
2 > date()      # current date and time values
```

One can change the format of the dates.

```
1 > dates = c("23.05.1984", "2000/01/01", "05.03.1256")  
2 > as.Date(dates[1], "%d.%m.%Y")  
3 [1] "1984-05-23"  
4 > as.Date(dates[2], "%Y/%m/%d")  
5 [1] "2000-01-01"  
6 > as.Date(dates[3], "%m.%d.%Y")  
7 [1] "1256-05-03"  
8 > dates.a = c(dates1, dates2, dates3)  
9 > format (dates.a, "%Y")  
10 [1] "1984" "2000" "1256"
```



Date types

- The difference between two dates is calculated by the function

```
1 > difftime(Sys.time(), dates.a)
2 [1] Time differences in days
3 11165.421   5464.421  277081.421
```

- There is also a possibility to go through day, months, weeks and weekdays.

```
1 > m = months(dates.a)    # m = "May" "January" "May"
2 > weekdays(dates.a)
3 [1] "Wednesday" "Saturday" "Wednesday"
4 > q = quarters(dates.a) # q = "Q2" "Q1" "Q2"
```



The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

Motivation

- applied mathematics appears in different disciplines e.g.: chemistry, biology, geology, management, economics
- the demand for numerical computation has considerably increased
- these problems frequently have no analytical solution or the exact result is time-consuming to derive
- to solve these problems numerical techniques are used to approximate the result



Numerical Techniques

Matrix algebra

Numerical Integration

Numerical Differentiation

Root Finding

Matrix

- A matrix $\mathcal{A}_{(n \times p)}$ is a system of numbers with n rows and p columns:

$$\mathcal{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1p} \\ a_{21} & a_{22} & \dots & a_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & \dots & \dots & a_{np} \end{pmatrix} = (a_{ij}).$$



Diagonal matrices

```
1 > A = matrix(1:9, nrow = 3, ncol = 3, byrow = T)
2 > diag(x = A)    # extracts diagonal elements
3 [1] 1 5 9
4 > C = diag(3)    # creates identity matrix
5 > C
6      [,1] [,2] [,3]
7 [1,]     1     0     0
8 [2,]     0     1     0
9 [3,]     0     0     1
```



Rank

- The rank of matrix \mathcal{A} denoted by $\text{rank}(\mathcal{A})$ returns the maximum number of linearly independent rows or columns.
- Linear independence for a set of h rows a_j of matrix $\mathcal{A}_{(n \times p)}$ means, that $\sum_{j=1}^h c_j a_j = 0_p$ if and only if $c_j = 0$, for all j .
- If the rank is equal to the number of rows or columns, the matrix is called a full-rank matrix.
- In R the rank can be calculated using the function `qr()` with the function value `rank`.

```
1 > A = matrix(1:9, nrow = 3, ncol = 3, byrow = T)
2 > qr(A)$rank # returns the rank of matrix A
3 [1] 2
```



Trace

- The trace of a matrix $\text{tr}(A)$ is the sum of its diagonal elements,

$$\text{tr}(A) = \sum_{i=1}^{\min(n,p)} a_{ii}.$$

- The trace of a scalar just equals the scalar itself. One obtains the trace in R by combining the functions `diag()` and `sum()`.

```
1 > A = matrix(1:12, nrow = 4, ncol = 3)
2 > sum(diag(A))      # computes the trace
3 [1] 18
```



Determinant

- The formal definition of the determinant of a square matrix

$\mathcal{A}_{(p \times p)}$ is

$$\det(\mathcal{A}) = \sum (-1)^{|\tau|} a_{1\tau_1} \dots a_{p\tau_p}. \quad (1)$$

- For a square matrix $\mathcal{A}_{(2 \times 2)}$ of dimension two, the determinant is

$$\det(\mathcal{A}_{(2 \times 2)}) = a_{11}a_{22} - a_{12}a_{21}.$$

```
1 > A = matrix(1:9, nrow = 3, ncol = 3)
2 > det(A)
3 [1] 0
```



Transpose

- A matrix $\mathcal{A}_{(n \times p)}$ has a transpose $\mathcal{A}_{(n \times p)}^\top$, which is obtained by reordering the elements of the original matrix.

$$\mathcal{A}_{(n \times p)}^\top = (a_{ij})^\top = (a_{ji}).$$

- The resulting matrix has now p rows and n columns and is obtained by function `t()` in R.

```
1 > A = matrix(1:9, nrow = 3, ncol = 3, byrow = T)
2 > t(A)
3      [,1] [,2] [,3]
4 [1,]     1     4     7
5 [2,]     2     5     8
6 [3,]     3     6     9
```



Conjugate transpose

- Every matrix $A_{(n \times p)}$ has a conjugate transpose $A_{p \times n}^C$.
- If a matrix entry $a_{ij} = \alpha + \beta i$ is a complex number with real numbers α, β and imaginary unit $i = \sqrt{-1}$, then its conjugate is $a_{ij}^C = \alpha - \beta i$ and vice versa.
- Therefore the conjugate transpose is:

$$A^C = \begin{pmatrix} a_{11}^C & a_{21}^C & \dots & a_{n1}^C \\ a_{12}^C & a_{22}^C & \dots & a_{n2}^C \\ \vdots & \vdots & \ddots & \vdots \\ a_{p1}^C & \dots & \dots & a_{np}^C \end{pmatrix}. \quad (2)$$



Conjugate Transpose

```
1 > a = c(1 + 0.5i, 1, 1,
2 +                   1, 1, 1 - 0.5i) # matrix entries
3 > A = matrix(a, nrow = 2,           # complex matrix
4 +                   ncol = 3,
5 +                   byrow = T)
6 > AC = Conj(t(A))                  # conjugate transpose
7 > AC
8      [,1]    [,2]
9 [1,] 1-0.5i 1+0.0i
10 [2,] 1+0.0i 1+0.0i
11 [3,] 1+0.0i 1+0.5i
```



Basic Operations

- For matrices $A_{(n \times p)}$ and $B_{(n \times p)}$ of the same dimensions, matrix addition and subtraction work elementwise as follows:

$$A + B = (a_{ij} + b_{ij}),$$

$$A - B = (a_{ij} - b_{ij}).$$

```
1 > A = matrix(3:11, nrow = 3, ncol = 3, byrow = T)
2 > B = matrix(-3:-11, nrow = 3, ncol = 3, byrow = T)
3 > A + B
4      [,1] [,2] [,3]
5 [1,]     0     0     0
6 [2,]     0     0     0
7 [3,]     0     0     0
```



Basic Operations

- Elementary operations including addition, subtraction, multiplication, and division can also be used with a scalar and a matrix in R.
- For example the modulo operation:

```
1 > A = matrix(1:9, nrow = 3, ncol = 3, byrow = T)
2 > A %% 2 # performs modulo operation
3     [,1] [,2] [,3]
4 [1,]     1     0     1
5 [2,]     0     1     0
6 [3,]     1     0     1
```

- Elementary operations including addition +, subtraction -, multiplication *, and division / between two matrices are interpreted in R as elementwise operations.



Basic Operations

- Matrix multiplication returns the matrix product of matrices $A_{(n \times p)}$ and $B_{(p \times m)}$.

$$A_{(n \times p)} \cdot B_{(p \times m)} = C_{(n \times m)} = \begin{pmatrix} \sum_{i=1}^p a_{1i} b_{i1} & \dots & \sum_{i=1}^p a_{1i} b_{im} \\ \sum_{i=1}^p a_{2i} b_{i1} & \dots & \sum_{i=1}^p a_{2i} b_{im} \\ \vdots & \ddots & \vdots \\ \sum_{i=1}^p a_{ni} b_{i1} & \dots & \sum_{i=1}^p a_{ni} b_{im} \end{pmatrix}$$



Basic Operations

- In R one uses the operator `%*%` between two objects for matrix multiplication.
- The objects have to be of class vector or matrix.

```
1 > A = matrix(3:11, nrow = 3, ncol = 3, byrow = T)
2 > B = matrix(-3:-11, nrow = 3, ncol = 3, byrow = T)
3 > A %*% B # performs matrix multiplication
4
5      [,1] [,2] [,3]
6 [1,]   -78   -90  -102
7 [2,]  -132  -153  -174
8 [3,]  -186  -216  -246
```



Inverse

- The division operation for square matrices is done by inverting a matrix.
- The inverse \mathcal{A}^{-1} of a square matrix $\mathcal{A}_{(p \times p)}$ exists if $\det(\mathcal{A}) \neq 0$ and it holds:

$$\mathcal{A}^{-1}\mathcal{A} = \mathcal{A}\mathcal{A}^{-1} = \mathcal{I}_p. \quad (3)$$

- The inverse of $\mathcal{A} = (a_{ij})$ can be calculated by

$$\mathcal{A}^{-1} = \frac{\mathcal{W}}{\det(\mathcal{A})},$$

where $\mathcal{W} = (w_{ij})$ is the adjoint matrix of \mathcal{A} .



Inverse

```
1 > A = matrix(nrow = 3, ncol = 3, byrow = T,
2 +             c(1, 2, 5,
3 +               3, 9, 2,
4 +               2, 2, 2))
5 > solve(A)
6      [,1]   [,2]   [,3]
7 [1,] -0.28 -0.12  0.82
8 [2,]  0.04  0.16 -0.26
9 [3,]  0.24 -0.04 -0.06
```



Generalized Inverse

- In practice one is often confronted with singular matrices, whose determinant is equal to zero.
- In this situation the inverse can be given by a generalized inverse \mathcal{A}^- satisfying:

$$\mathcal{A}\mathcal{A}^-\mathcal{A} = \mathcal{A}. \quad (4)$$

```
1 > require(MASS)
2 > A = matrix(c(1, 0, 0, 0), 2, 2)
3 > ginv(A)      # generalized inverse
4     [,1] [,2]
5 [1,]    1    0
6 [2,]    0    0
```



Vector Norm

Definition

Let V be a vector space, b be a scalar both lying either in \mathbb{R}^n or \mathbb{C}^n . Consider the vectors $x, y \in V$. Then a norm is a mapping, $\|\cdot\| : V \rightarrow \mathbb{R}_0^+$, with the following properties:

1. $\|bx\| = |b|\|x\|$,
2. $\|x + y\| \leq \|x\| + \|y\|$,
3. $\|x\| \geq 0$, where $\|x\| = 0$ if and only if $x = 0$.



Vector Norm

- In R the function `norm()` can return the following norms:

```
1 > x = matrix(c(2, 1, 2), nrow = 3, ncol = 1)
2 > norm(x, type=c("0")) # Manhattan norm
3 [1] 5
4 > norm(x, type=c("I")) # infinity norm
5 [1] 2
6 > norm(x, type=c("F")) # Frobenius norm
7 [1] 3
8 > norm(x, type=c("2")) # Euclidean norm
9 [1] 3
```

- The object `x` has to be of class `matrix` in R to compute all norms.



Matrix Norm

Definition

Let $U^{n \times p}$ be a set of $(n \times p)$ matrices and a be a scalar, which are either real or complex. $U^{n \times p}$ is a vector space equipped with matrix addition and scalar multiplication. Let $\mathcal{A}, \mathcal{B} \in U^{n \times p}$, then a matrix norm is a mapping, $\|\cdot\| : U^{n \times p} \rightarrow \mathbb{R}_0^+$, with the following properties:

1. $\|a\mathcal{A}\| = |a|\|\mathcal{A}\|,$
2. $\|\mathcal{A} + \mathcal{B}\| \leq \|\mathcal{A}\| + \|\mathcal{B}\|,$
3. $\|\mathcal{A}\| \geq 0$, where $\|\mathcal{A}\| = 0$ if and only if $\mathcal{A} = 0$.



Matrix Norm

- The five matrix norms are computed with the function `norm()` in R.

```
1 > A = matrix(c(1, 2, 3, 4),  
2     ncol = 2, nrow = 2, byrow = T)  
3 > norm(A, type = c("1")) # one norm  
4 [1] 6                      # maximum of column sums  
5 > norm(A, type = c("I")) # infinity norm  
6 [1] 7                      # maximum of row sums  
7 > norm(A, type = c("F")) # Frobenius norm  
8 [1] 5.477226  
9 > norm(A, type = c("2")) # Euclidean norm  
10 [1] 5.464986
```



Eigenvalues and -vectors

Definition

Let V be any real vector space and $L : V \rightarrow V$ be a linear transformation mapping. Then a non-zero vector $\gamma \in V$ is called an eigenvector, if and only if there exists a scalar $\lambda \in \mathbb{R}$ such that $L(\gamma) = \mathcal{A}\gamma = \lambda\gamma$. Therefore it holds that:

$$\lambda \text{ is eigenvalue of } \mathcal{A} \Leftrightarrow \det(\mathcal{A} - \lambda\mathcal{I}) = 0.$$



Eigenvalues and -vectors

- In R the eigenvalues and eigenvectors of matrix \mathcal{A} can be calculated using the function `eigen`.

```
1 > A = matrix(c(2, 0, 1, 0, 3, 1, 0, 6, 2),  
2 +                 nrow = 3, ncol = 3, byrow = T)  
3 > Eigen = eigen(A)      # eigenvectors and -values  
4 > Eigen$values          # returns eigenvalues  
5 [1] 5 2 0  
6 > Eigen$vectors         # returns eigenvectors  
7          [,1] [,2]      [,3]  
8 [1,] 0.2857143     1 -0.4285714  
9 [2,] 0.4285714     0 -0.2857143  
10 [3,] 0.8571429    0  0.8571429
```



Spectral Decomposition

Theorem

Let $\mathcal{A}_{(p \times p)}$ be a matrix with real-valued entries and eigenvalues $\lambda_1, \dots, \lambda_p$ and the corresponding eigenvectors $\gamma_1, \dots, \gamma_p$. Let \mathcal{P} be the eigenvector matrix with $\gamma_1, \dots, \gamma_p$ as columns and Λ be the eigenvalue matrix with $\lambda_1, \dots, \lambda_p$ as diagonal entries. Then

$$\mathcal{A} = \mathcal{P}\Lambda\mathcal{P}^{-1}. \quad (5)$$



Spectral Decomposition

- It is possible to decompose matrix \mathcal{A} by (5) in R with the following code:

```
1 > A = matrix(c(2, 0, 1, 0, 3, 1, 0, 6, 2),
2 +           nrow = 3, ncol = 3, byrow = T)
3 > P = eigen(A)$vectors      # eigenvector matrix
4 > L = diag(eigen(A)$values) # eigenvalue matrix
5 [1] 3
6 > P %*% L %*% solve(P)      # spectral decomposition
7     [,1]          [,2]  [,3]
8 [1,]    2 4.440892e-16    1
9 [2,]    0 3.000000e+00    1
10 [3,]   0 6.000000e+00    2
```



Numerical Techniques

Matrix algebra

Numerical Integration

Numerical Differentiation

Root Finding

Univariate Integration

- ◻ In calculus it is not possible to obtain the analytical representation of the corresponding indefinite integral for every function $f \in C[a, b]$

$$\int \exp(-x^2) dx. \quad (6)$$

- ◻ A corresponding definite integral has to be computed numerically using numerical integration or the so-called quadrature.



Newton Cotes Rule

- If a function $f \in C[a, b]$ and nodes $a = x_0 < x_1 < \dots < x_n = b$ are given, one looks for a polynomial $p_n(x) = \sum_{j=1}^n c_j x^{j-1} \in P_n$, where P_n are basis polynomials, with the condition:

$$p_n(x_k) = f(x_k), \text{ for all } k \in \{0, \dots, n\}. \quad (7)$$

- To construct a polynomial that satisfies the above condition, the following basis polynomials P_n are used:

$$L_k(x) = \prod_{i=0, i \neq k}^n \frac{x - x_i}{x_k - x_i}. \quad (8)$$

- This leads to the so-called Lagrange polynomial, which satisfies the condition in (7) (assuming $\frac{0}{0} = 1$).



Newton Cotes Rule

- For example consider the integral of the cosine function on $[-\frac{\pi}{2}, \frac{\pi}{2}]$ and split the interval in 10 subintervals, where the *trapezoidal rule* is applied:

```
1 > require("caTools")
2 > x = (-5:5)*(pi/2)/5 # set interval
3 > c = trapz(x, cos(x)) # integration
4 > c # result of the integration
5 [1] 1.983524
6 > abs(c - 2) # absolute error of the solution
7 [1] 0.01647646
```

- The integral of the cosine function on $[-\frac{\pi}{2}, \frac{\pi}{2}]$ is 2.
- The absolute error is almost 0.02.



Gaussian Quadrature

Definition

A method of numerical integration for a function $f : [a, b] \rightarrow \mathbb{R}$ with the formula

$$I(f) = \int_a^b f(x)dx \approx \int_a^b w(x)p(x)dx \approx I_n^G(f) = \sum_{k=1}^n f(x_k)\alpha_k,$$

and n nodes x_1, \dots, x_n is called Gaussian quadrature, if an arbitrary weighting function $w(x)$ and a polynomial $p(x)$ of degree $2n - 1$ exactly approximate $f(x)$, such that $f(x) = w(x)p(x)$.



Gaussian Quadrature

- Consider again the cosine function on the interval $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

```
1 > require("stats")
2 > integrate(cos, -pi/2, pi/2)
3 with absolute error < 2.2e-14
```

- The output of the `integrate()` function delivers the computed value of the definite integral and an upper bound on the absolute error.



Multivariate Integration

Example 1

Integrate the bivariate function

$$\int_0^1 \int_0^1 x^2 y^3 dx dy. \quad (9)$$

- The analytical solution of this integral is $1/12$.
- The surface of $x^2 y^3$ for the interval $[0, 1]$ is depicted in Figure 1.
- For the computation of multiple integrals R package R2Cuba is used.



Multivariate Integration

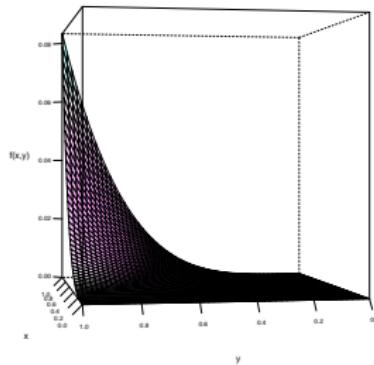


Figure 1: Plot of the multivariate function (9).

BCS_Integrand



Set Up

```
1 > NoDim = 2 # dimension of integration
2 > NoInt = 1 # number of integrand
3 > integrand = function(arg, phase){
4 +   x = arg[1] # function argument x
5 +   y = arg[2] # function argument y
6 +   f = x^{2} * y^{3} # function
7 +   return(f)}
```



The Adaptive Method

- The adaptive method is applied to example (9) via the function cuhre.

```
1 > cuhre(NoDim, NoInt, integrand,
2 +   lower = rep(0, 2), upper = rep(1, 2),
3 +   rel.tol = 1e-3, abs.tol = 1e-12,
4 +   flags = list(verbose = 0))
5 integral: 0.083333333 (+-1.3e-15)
6 nregions: 2 number of evaluations: 195
7 probability: 0
```



The Monte-Carlo Method

- The Monte-Carlo method is applied to example (9) via the function vegas.

```
1 > vegas(NoDim, NoInt, integrand,
2 +   lower = rep(0, 2), upper = rep(1, 2),
3 +   rel.tol = 1e-3, abs.tol = 1e-12,
4 +   flags = list(verbose = 0))
5 integral: 0.08329357 (+-7.5e-05)
6 number of evaluations: 17500
7 probability: 0.1201993
```



Numerical Techniques

Matrix algebra

Numerical Integration

Numerical Differentiation

Root Finding

The Gradient

Definition

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be a differentiable function and

$x = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$. Then the vector

$$\nabla f(x) \stackrel{\text{def}}{=} \left\{ \frac{\partial f}{\partial x_1}(x), \dots, \frac{\partial f}{\partial x_n}(x) \right\}^\top$$

is called the gradient of f at point x .



The Jacobian

Definition

Let $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ be a differentiable function with $F = (f_1, \dots, f_m)$ and the coordinates $x = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$, then the Jacobian matrix in a point $x \in \mathbb{R}^n$ is defined as follows:

$$J_F(x) \stackrel{\text{def}}{=} \left\{ \frac{\partial f_i(x)}{\partial x_j} \right\}_{i=1,\dots,m; j=1,\dots,n}.$$



The Hessian

Definition

Let $f : \mathbb{R}^n \rightarrow \mathbb{R}$ be twice continuously differentiable. Then the Hessian matrix of f at a point x is defined as follows:

$$H_f(x) \stackrel{\text{def}}{=} \left\{ \frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right\}_{i,j=1,\dots,n}. \quad (10)$$



Analytical Differentiation

- The function `deriv()` returns for an expression the gradient vector.

```
1 > f = expression(x^2 + y^2) # function
2 > euclid2 = deriv(f, c("x", "y"),
3 +   function.arg=c("x", "y"), hessian = T)
4 > attr(euclid2(2, 2), "gradient") # gradient vector
5     x  y
6 [1,] 4  4
```



Analytical Differentiation

- The function `deriv()` returns for an expression the Hessian matrix.

```
1 > f = expression(x^2 + y^2) # function
2 > euclid2 = deriv(f, c("x", "y"),
3 +   function.arg=c("x", "y"), hessian = T)
4 > h = attr(euclid2(2, 2), "hessian")
5 > H = rbind(h[, , 1], h[, , 2]) # Hessian matrix
6 > H
7
8      x  y
9 [1,] 2  0
10 [2,] 0  2
```



Numerical Differentiation

- To develop numerical methods for determining derivatives of a function at a point x one uses the Taylor expansion

$$f(x+h) = f(x) + h \cdot f'(x) + \frac{h^2}{2!} f''(x) + \frac{h^3}{3!} f'''(x) + \mathcal{O}(h^3). \quad (11)$$

- If the Taylor expansion is truncated after the linear term, then (11) can be solved for $f'(x)$:

$$f'(x) = \frac{f(x+h) - f(x)}{h} + \mathcal{O}(h^2). \quad (12)$$

- Therefore an approximation for the derivative at point x could be

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}. \quad (13)$$



Numerical Differentiation

- The R package numDeriv incorporates the functions grad(), jacobian() and hessian().

```
1 > require(numDeriv)
2 > func1 = function(x){c(sin(sum(x)), cos(sum(x)))}
3 > x      = (0:1) * 2 * pi
4 > jacobian(func1, x)
5     [,1] [,2]
6 [1,]    1    1
7 [2,]    0    0
```



Numerical Differentiation

```
1 > require(numDeriv)
2 > func = function(x){sqrt(sum(x^2))}
3 > # gradient vector
4 > grad(func, c(1,0,0), method = "Richardson")
5 [1] 1 0 0
6 > # Hessian matrix
7 > hessian(func, c(0,0,0))
8 [1,]      [,1]      [,2]      [,3]
9 [1,] 194419.75 -56944.23 -56944.23
10 [2,] -56944.23 194419.75 -56944.23
11 [3,] -56944.23 -56944.23 194419.75
```



Automatic Differentiation

- Automatic differentiation is more accurate than numerical differentiation.

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad (14)$$

or

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}. \quad (15)$$

- The accuracy of numerical differentiation is related to the choice of h .
- If h is small, then divided difference has errors introduced by rounding-off the floating point numbers.



Automatic Differentiation

- Consider the following multivariate function: $f(x) = \prod_{i=1}^{10} x_i$.
- All arguments of the respective function are redefined as dual numbers $x_i + x'_i \varepsilon$ for automatic differentiation.
- Here, ε has the property that $\varepsilon^2 \approx 0$.
- The automatic differentiation for the presented function looks as follows:

$$\nabla f(x) = \prod_{i=1}^{10} x_i + \varepsilon \left(x_1' \prod_{i=2}^{10} x_i + \cdots + x_j' \prod_{\substack{i=1 \\ i \neq j}}^{10} x_i + \cdots + x_{10}' \prod_{i=1}^9 x_i \right) \quad (16)$$



Automatic Differentiation

- It is necessary to install `radx`

```
install_github(radx,"quantumelixir")".
```

- Function `radxeval()` implements automatic differentiation.

```
1 > require(radx)                      # not provided by CRAN
2 > f = function(x) {(x^2 + x)^3} # function
3 > radxeval(f, c(2), 1)          # automatic
   differentiation
4
5 [,1]
5 [1,] 540
```



Numerical Techniques

Matrix algebra

Numerical Integration

Numerical Differentiation

Root Finding

Solving Systems of Linear Equations

Example 2

Solve the following system of linear equations in R with the Gaussian algorithm and the back-substitution,

$$\mathcal{A}x = b$$

$$\mathcal{A}^e = \begin{pmatrix} 2 & -\frac{1}{2} & -\frac{1}{2} & 0 & 0 \\ -\frac{1}{2} & 0 & 2 & -\frac{1}{2} & 3 \\ -\frac{1}{2} & 2 & 0 & -\frac{1}{2} & 3 \\ 0 & -\frac{1}{2} & -\frac{1}{2} & 2 & 0 \end{pmatrix}. \quad (17)$$



Solving Systems of Linear Equations

- Two parameters are required the coefficient matrix A and the vector of constraints b to use solve().

```
1 > A = matrix( byrow = TRUE,      # coefficient matrix
2 +           c( 2, -1/2, -1/2,      0,
3 +             -1/2,      0,       2, -1/2,
4 +             -1/2,      2,       0, -1/2,
5 +               0, -1/2, -1/2,      2),
6 +           ncol = 4, nrow = 4)
7 > b = c(0, 3, 3, 0)            # vector of constants
8 > solve(A, b)
9 [1] 1 2 2 1                   # x1, x2, x3, x4
```



Solving Systems of Nonlinear Equations

- A system of nonlinear equations is represented by a function $F = (f_1, \dots, f_n) : \mathbb{R}^n \rightarrow \mathbb{R}^n$.
- Any nonlinear system has a general extended form:

$$\begin{cases} f_1(x_1, \dots, x_n) = 0, \\ \vdots \\ f_n(x_1, \dots, x_n) = 0. \end{cases}$$



Solving Systems of Nonlinear Equations

- ◻ There are many different numerical methods to solve a system of nonlinear equations.
- ◻ In general one distinguishes between gradient and non-gradient methods.
- ◻ The *Newton method* or *Newton-Raphson method* is introduced.
- ◻ To get a better illustration of the idea behind the *Newton method*, consider a continuous differentiable function $F : \mathbb{R} \rightarrow \mathbb{R}$.
- ◻ One tries to find x^* with $F(x^*) = 0$ and $\frac{\partial F(x)}{\partial x} \Big|_{x=x^*} \neq 0$.



Solving Systems of Nonlinear Equations

- start by choosing a starting value $x_0 \in \mathbb{R}$ and define the tangent line

$$p(x) = F(x_0) + \frac{\partial F(x)}{\partial x} \Big|_{x=x_0} (x - x_0). \quad (18)$$

- the tangent $p(x)$ is a good approximation for F in a sufficiently small neighbourhood of x_0
- if $\frac{\partial F(x)}{\partial x} \Big|_{x=x_0} \neq 0$, the root x_1 of the tangent in (18) can be computed as follows:

$$x_1 = x_0 - \frac{F(x_0)}{\frac{\partial F(x)}{\partial x} \Big|_{x=x_0}}.$$



Solving Systems of Nonlinear Equations

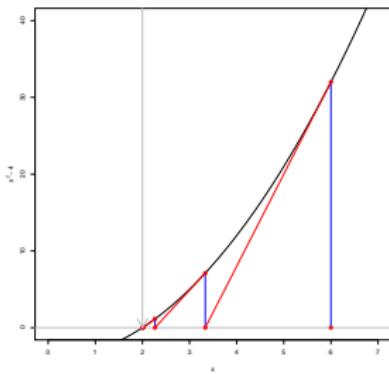


Figure 2: Illustration for the iteration steps of *Newton's method* to find the root for $f(x) = x^2 - 4$ with $x_0 = 6$.

 BCS_Newton



Solving Systems of Nonlinear Equations

- In R the function `uniroot()` finds the root for nonlinear equations.

```
1 > f = function(x)
2     {-x^4 - cos(x) + 9*x^2 - x - 5} # objective function
3 > uniroot(f, c(0, 2))$root # root in [0,2]
4 [1] 0.8913574
5 > uniroot(f, c(-3, 2))$root # root in [-3,2]
6 [1] -2.980569
```



Maximization and Minimization

Definition (Global Extrema)

A real function f defined on a domain M has a global maximum point at x_{opt} if $f(x_{opt}) \geq f(x)$ for all x in M . Then $f(x_{opt})$, the function value at the maximum point x_{opt} is called the maximum value of the function. Analogously, the function has a global minimum point at x_{opt} if $f(x_{opt}) \leq f(x)$ for all x in M . Then $f(x_{opt})$, the function value at the minimum point x_{opt} , is called the minimum value of the function.



Maximization and Minimization

Definition (Local Extrema)

If the domain M is a metric space then f is said to have a local maximum point at the point x_{opt} if there exists some neighbourhood $\epsilon > 0$ such that $f(x_{opt}) \geq f(x)$ for all x in M within distance ϵ centring on x_{opt} . Analogously, the function has a local minimum point at x_{opt} if $f(x_{opt}) \leq f(x)$ for all x in M within distance ϵ centring on x_{opt} .



Maximization and Minimization

Example 3

The following function possesses multiple local maximum, local minimum, global maximum and global minimum points.

$$\begin{aligned} f(x, y) = & 0.03 \sin(x) \sin(y) - 0.05 \sin(2x) \sin(y) \\ & + 0.01 \sin(x) \sin(2y) + 0.09 \sin(2x) \sin(2y). \end{aligned} \quad (19)$$

The function is plotted in Figure 3 with its extrema depicted by points.



Maximization and Minimization

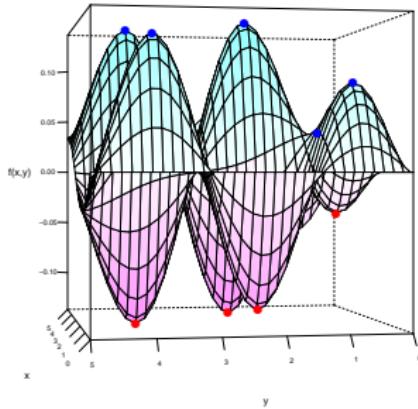


Figure 3: 3D plot for function (19). All maxima are blue points, and all minima are red points.

Golden Section Search Method

1. Define upper bound x_U and lower bound x_L , $x_U > x_L$.
2. Set $r = (\sqrt{5} + 1)/2$, $d = (x_U - x_L)r$ and ε .
3. Choose x_1 and $x_2 \in [x_L, x_U]$. Set $x_1 = x_L + d$ and $x_2 = x_U - d$.
4. Stop, if $|f(x_U) - f(x_L)| < \varepsilon$ or $|x_U - x_L| < \varepsilon$. If $f(x_1) < f(x_2) \Rightarrow x_{min} = x_1$, otherwise $x_{min} = x_2$.
5. Update x_U or x_L , if $|f(x_U) - f(x_L)| \geq \varepsilon$ or $|x_U - x_L| \geq \varepsilon$. If $f(x_1) > f(x_2) \Rightarrow x_U = x_2$, otherwise $x_L = x_1$.
6. Return to 2.



Golden Section Search Method

- Apply the golden ratio search method to find the maximum of $f(x) = -(x - 3)^2 + 10$.

```
1 > require("stats")
2 > f = function(x){-(x - 3)^2 + 10}
3 > optimize(f, c(-10, 10),
4           tol = 0.0001, maximum = T)
5 $maximum
6 [1] 3
7 $objective
8 [1] 10
```



Nelder-Mead Method

1. Choose x_1, x_2, x_3 such that $f(x_1) < f(x_2) < f(x_3)$ and set ϵ_x and/or ϵ_f .
2. Stop, if $\|x_i - x_j\| < \epsilon_x$ and/or $\|f(x_i) - f(x_j)\| < \epsilon_f$, for $i \neq j$, $i, j \in \{1, 2, 3\}$ and set $x_{min} = x_1$.



Nelder-Mead Method

3. Else, compute $z = \frac{1}{2}(x_1 + x_2)$ and $d = 2z - x_3$.

If $f(x_1) < f(d) < f(x_2) \Rightarrow x_3 = d$.

If $f(d) \leq f(x_1)$, compute $k = 2d - z$.

If $f(k) < f(x_1) \Rightarrow x_3 = k$.

Else, $x_3 = d$.

If $f(x_3) > f(d) \geq f(x_2) \Rightarrow x_3 = d$.

Else, compute $t = [t | f(t) = \min\{f(t_1), f(t_2)\}]$, where

$t_1 = \frac{1}{2}(x_3 + z)$ and $t_2 = \frac{1}{2}(d + z)$.

If $f(t) < f(x_3) \Rightarrow x_3 = t$.

Else, $x_3 = s = 1/2(x_1 + x_3)$ and $x_2 = z$.

4. Return to 2.



Nelder-Mead Method

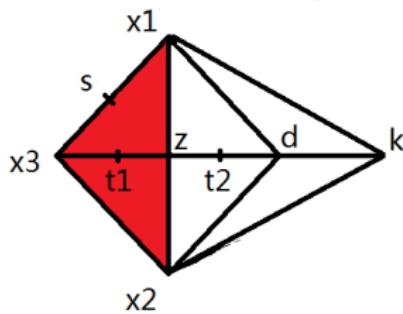


Figure 4: Algorithm graph for Nelder-Mead Method. The variables x_1 , x_2 and x_3 are the search region at the specific iteration step. All other variables, d , k , s , t_1 and t_2 , are possible updates for one x_i .



The Rosenbrock Function

Example 4

The function to be minimized is the Rosenbrock function, which has an analytic solution with global minimum in $(1, 1)$ and global minimum value of the function $f(1, 1) = 0$.

$$f(x_1, x_2) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2. \quad (20)$$



The Rosenbrock Function

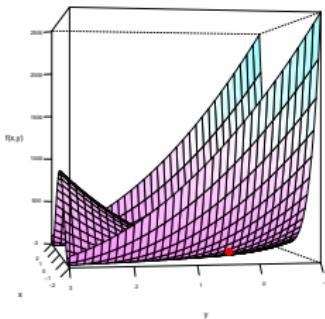


Figure 5: Plot for the Rosenbrock function with its minimum depicted by a red point.

BCS_Rosenbrock



The Nelder Mead Method

```
1 > require(neldermead)
2 > fct = function(x){
3 + y = 100*(x[2] - x[1]^2)^2 + (1 - x[1])^2
4 +
5 > answer = fminsearch(fun = fct,
6 +           x0 = c(-1.2, 1), verbose = F)
7 > neldermead.get(answer, key = "xopt")
8     [,1]
9 [1,] 1.000022
10 [2,] 1.000042
11 > neldermead.get(answer, key = "fopt")
12 [1] 8.177661e-10
```



The BFGS Method

- The main idea of this method originated from the Newton's method.
- A second-order Taylor expansion for a twice differentiable function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ at $x = x_i \in \mathbb{R}^n$ is employed, such that

$$f(x) = f(x_i) + \nabla f^\top(x_i)q + \frac{1}{2}q^\top H(x_i)q.$$

- Here q , and $\nabla f(x_i)$ is the value of the partial derivative of function f at point x_i , and $H(x_i)$ is the Hessian matrix.
- According to the first-order condition one obtains
$$\nabla f(x) = \nabla f(x_i) + H(x_i)q = 0.$$
- If $H(x_i)$ is invertible then,

$$q = -H^{-1}(x_i)\nabla f(x_i),$$

$$x - x_i = -H^{-1}(x_i)\nabla f(x_i),$$

$$x_{i+1} = x_i - H^{-1}(x_i)\nabla f(x_i).$$



The BFGS Method

- The problem is that Newton's method requires the computation of the exact Hessian at each iteration, which is computationally expensive.
- Therefore the BFGS method overcomes this disadvantage with an approximation of Hessian's inverse obtained from the following optimization problem,

$$H(x_i) = \arg \min_H \|H^{-1} - H^{-1}(x_{i-1})\|_{\mathcal{W}}, \quad (21)$$

subject to:

$$\begin{aligned}\mathcal{B}^{-1} &= (H^{-1})^\top, \\ \mathcal{B}^{-1}(\nabla f_i - \nabla f_{i-1}) &= x_i - x_{i-1}.\end{aligned}$$



The BFGS Method

```
1 > require("optimx")
2 > answer = optimx(fn=fct,                                     # optimization
3 +                         par=c(-1.2, 1),
4 +                         method=c("BFGS"))
5 > print(data.frame(answer$p1,
6 +                         answer$p2,
7 +                         answer$value))
8   answer.p1 answer.p2 answer.value
9 1 0.9998044 0.9996084 3.827383e-08      # minimum
```



Conjugate Gradient Method

- The Conjugate Gradient method is applied to solve a positive definite symmetric linear equation system:

$$\mathcal{A}x = b, \quad \mathcal{A} \in \mathbb{R}^{n \times n}, \quad \mathcal{A} = \mathcal{A}^T, \quad \text{and } \mathcal{A} \text{ positive definite.}$$

- The main idea behind this method is to utilize iterations stepwise to approach the optimum of the linear system.



Conjugate Gradient Method

1. Set x_0 and ε , then compute $p_0 = r_0 = b - \mathcal{A}x_0$.
2. Stop, if $r_i < \varepsilon$ and set $x_{opt} = x_{i+1}$.
3. Else, compute:

$$\begin{aligned}\alpha_i &= \frac{r_i^\top r_i}{p_i^\top \mathcal{A} p_i}, \\ x_{i+1} &= x_i + \alpha_i p_i, \\ r_{i+1} &= r_i - \alpha_i \mathcal{A} p_i, \\ \beta_i &= \frac{r_{i+1}^\top r_{i+1}}{r_i^\top r_i}, \\ p_{i+1} &= r_{i+1} + \beta_i p_i.\end{aligned}$$

4. Update x_i , r_i and p_i .
5. Return to 2.



Conjugate Gradient Method

```
1 > require("optimx")
2 > epsilon = list(reltol = 10^-7) # set tolerance
3 > fct = function (x){
4 +   y = 100*(x[2] - x[1]^2)^2 + (1 - x[1])^2
5 > answer = optimx(fn = fct, # objective function
6 +   par= c(1.2, 1), # initial guess (x_0)
7 +   control = epsilon, # relative tolerance
8 +   method = c("CG")) # use CG method
9 >
10 > print(data.frame(answer$p1, answer$p2, answer$value))
11   answer.p1 answer.p2 answer.value
12 1 1.030077 1.061209 0.0009036108
```



Constrained Optimization

- Constrained optimization problems can be categorized into 2 classes with respect to the linearity of the objective function and the constraints.
- Linear programming (LP) problems have a linear objective function and linear constraints.
- Otherwise one faces a nonlinear programming problem (NLP).



Constrained Optimization LP

$$\arg \max_x a^\top x,$$

$$\begin{aligned} \text{subject to: } & Cx \leq b, \\ & x \geq 0, \end{aligned}$$

Example 5

Solve the following linear programming optimization problem with R.

$$\arg \max_{x_1, x_2} 2x_1 + 4x_2, \tag{22}$$

$$\begin{aligned} \text{subject to: } & 3x_1 + 4x_2 \leq 60, \\ & x_1 \geq 0, \\ & x_2 \geq 0. \end{aligned}$$



Constrained Optimization LP

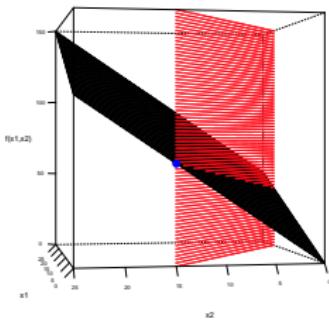


Figure 6: Plot for the linear programming problem with the constraint hyperplane depicted by the point grid and the optimum by a blue point.

BCS_LP



Constrained Optimization LP

```
1 > require("Rglpk")
2 > fct = c(2, 4)                      # the objective function
3 > st  = matrix(c(3, 4), nrow=1)       # coefficients
4 > leq = c("<=")                      # kind of constraint
5 > rhs = c(60)                        # constraints vector
6 > Rglpk_solve_LP(fct, st, leq, rhs, max=T)
7 $optimum
8 [1] 60
9 $solution
10 [1] 0 15
11 $status
12 [1] 0
```



Constrained Optimization NLP

Example 6

Next consider a constrained nonlinear optimization problem, which can be solved in R using `constrOptim()`. Solve the following non-linear optimization problem with R,

$$\arg \max_{x_1, x_2} \sqrt{5x_1} + \sqrt{3x_2}, \quad (23)$$

$$\text{subject to: } 3x_1 + 5x_2 \leq 10,$$

$$x_1 \geq 0,$$

$$x_2 \geq 0.$$



Constrained Optimization NLP

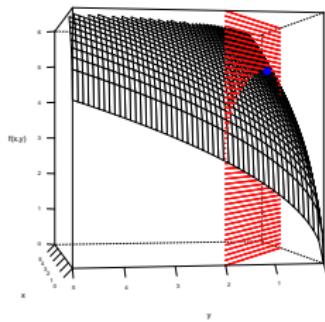


Figure 7: Plot for the objective function with its constraint from (23) and the optimum depicted by the blue point.

BCS_NLP



Constrained Optimization NLP

```
1 > require("stats")
2 > fct = function(x)                      # objective function
3 +     {sqrt(5*x[1]) + sqrt(3*x[2])}
4 > A = matrix(c(-3, -5), nrow = 1,
5 +     ncol = 2, byrow = TRUE)           # coefficients matrix
6 > b = c(-10)                           # constraints
7 > answer = constrOptim(c(1, 1), fct, # optimization
8 +     NULL, ui = A, ci = b,
9 +     control = list(fnscale = -1))
10 > c(answer$par, answer$value)         # optimum
11 [1] 2.4510595 0.5293643 4.7609523
```



The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

Combinatorics and Discrete Distributions

Set Theory

Probabilistic Experiment with Finite Sample Space

Binomial Distribution

Multinomial Distribution

Hypergeometric Distribution

Poisson Distribution

Set Theory

- Analysis of order and magnitude in mathematics by Georg Cantor (end of 19th century)
- 1908 axiomatic system by Ernst Zermelo, known today as Zermelo-Fraenkel set theory
- Sets: \mathbb{R} , \mathbb{N} , \mathbb{Q} , ...
- Basic properties:
 - ▶ The empty set \emptyset : $A \cup \emptyset = A$, $A \cap \emptyset = \emptyset$.
 - ▶ The full set Ω : $A \cup \Omega = \Omega$, $A \cap \Omega = A$.
 - ▶ The complementary set A^C : $A \cup A^C = \Omega$, $A \cap A^C = \emptyset$, $(A^C)^C = A$.
- Commutative, associative and distributive property



Creating Sets

```
1 > require("sets")
2 > x = c(1, 3, 3, 4, 2) # not sorted
3 > x
4 [1] 1 3 3 4 2
5 > subset(x, x < 4)
6 [1] 1 3 3 2
7 > seq(1, 2, by = 1/4)
8 [1] 1.00 1.25 1.50 1.75 2.00
9 > set(5, 7, 6, 5)      # sorted by set()
10 {5, 6, 7}
```

- Functions in base package imitate sets with lists, vectors
- Creating sets by either enumerating elements or by stating form of the set
- Create sets with `set()` from the `sets` package



Set Relations

operation	R basics	sets package
$x \in A$	<code>%in%</code>	<code>set_contains_element(A,x)</code>
$A = B$	<code>setequal(A,B)</code>	<code>set_is_equal(A,B), ==</code>
$A \cup B$	<code>union(A,B)</code>	<code>set_union(A,B), A B</code>
$A \cap B$	<code>intersect(A,B)</code>	<code>& B</code>
$A \setminus B$	<code>setdiff(A,B)</code>	<code>A-B</code>

Table 2: Set Operations in R



R Package sets

- Provides basic operations for:
 - ▶ Ordinary sets
 - ▶ Generalized sets including fuzzy sets and multisets



R Package sets

```
1 > set("a", "b", "c")
2 {"a", "b", "c"}
3 > gset(c(1.0, 2.0, 3.0),
4 +       c(0.2, 0.5, 0.9)) # fuzzy set
5 {1 [0.2], 2 [0.5], 3 [0.9]}
6 > A = c("x", "y", "z", "z", "z", "x")
7 > as.gset(A)           # multiset
8 {"x" [2], "y" [1], "z" [3]}
9 > table(A)
10 A
11 x y z
12 2 1 3
```



R Package sets

- Some useful functions for combinatoric issues

```
1 > s = set(1, 2, 4)
2 > t = set(3, 4)
3 > s * t
4 {(1, 3), (1, 4), (2, 3), (2, 4), (4, 3), (4, 4)}
5 > set_similarity(s, t)
6 [1] 0.25
7 > gset_cardinality(s * t)
8 [1] 6
9 > set_combn(set(2, 4, 6), 2)
10 {{2, 4}, {2, 6}, {4, 6}}
```



Combinatorics and Discrete Distributions

Set Theory

Probabilistic Experiment with Finite Sample Space

Binomial Distribution

Multinomial Distribution

Hypergeometric Distribution

Poisson Distribution

Probabilistic Experiments with Finite Sample Space

- Focus on random or nondeterministic experiments with finite number of outcomes
- Random trial yields outcome, all possible outcomes form sample space
- Subsets of sample space are events
- Examples: rolling a die, tossing a coin and urn sample



Probabilistic Experiments with Finite Sample Space

```
1 > require(prob)
2 > probspace(tosscoin(2))
3   toss1 toss2  probs
4   1      H      H  0.25
5   2      T      H  0.25
6   3      H      T  0.25
7   4      T      T  0.25
```



Axioms of Probability

- The function P assigns a probability to an event $a \in A$, i.e.
 $P : A \rightarrow [0, 1] \subset \mathbb{R}$, where A is the sample space
- $P(\Omega) = 1$, the entire sample space will occur with certainty
- $P(e_1 \cup e_2 \cup \dots) = P(e_1) + P(e_2) + \dots$ for disjunct events e_i
- $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
- $P(A \setminus B) = P(A) - P(A \cap B)$



Experiments with R

- The package prob has many useful functions for conducting experiments

```
1 > urnsample(x, size, replace = F, ordered = F)
2 > tosscoin(nCoins, makespace = F)
3 > roldie(ndies, nsides = 6, makespace = F)
4 > cards(jokers = F, makespace = F)
5 > roulette(european = F, makespace = F)
```

- For computing number of possible outcomes.
- Sampling Design:
 - ▶ with or without replacement
 - ▶ ordered or unordered



Sample Space

- Size of the sample space is:

	R with replacement	without replacement
ordered	n^k	$\frac{n!}{(n-k)!}$
unordered	$\frac{(n+k-1)!}{k!(n-1)!}$	$\frac{n!}{k!(n-k)!}$

Table 3: Choosing k elements out of set with n elements for different sample designs.

- Taking samples with the base function

```
sample(x, size, replace = T)
```



Sampling from Urns

- Taking samples with function
`urnsample(x, size, replace = F, ordered = F)`
- In contrast to `sample()`, here the order can be disregarded
- The argument `size` determines the number of balls to be drawn



Sample Examples

```
1 > u = urnsamples(1:5, 2, replace = F, ordered = T)
2 > head(probspace(u))
3   X1 X2 probs
4 1  1  2  0.05
5 2  2  1  0.05
6 3  1  3  0.05
7 4  3  1  0.05
8 5  1  4  0.05
9 6  4  1  0.05
```

- To get the probability of one outcome, compute one over the number of samples, as in table 2.



Sampling Procedures

- Famous: simple random sample (SRS)
- Sample according to the probability, e.g.
probability-proportional-to-size (PPS)
- Systematic sampling



Random Variables

- Outcomes of a probabilistic experiment might be described by a random variable (rv) X .
- All possible events ω are elements of Ω the event space.
- These outcomes of an probabilistic experiment are associated with distinct values x_j of X , for $j = \{1, \dots, k\}$.



Random Variables

Definition

A real valued random variable X on the probability space (Ω, \mathcal{F}, P) , is a real valued function $X(\omega)$ defined on Ω , such that for every Borel subset \mathcal{B} of the real numbers:

$$\{\omega : X(\omega) \in \mathcal{B}\} \in \mathcal{F}.$$

- The σ -algebra \mathcal{F} for real valued rv is the Borel σ -algebra.
- The probability function P assigns probabilities to the events.



Random Variables

- For the probabilistic experiment of tossing a fair coin $\Omega = \{H, T\}$ the random variable X is defined as follows:

$$X = \begin{cases} 1, & \text{if head } H \text{ shows up,} \\ 0, & \text{if tail } T \text{ shows up.} \end{cases}$$



Random Variables

- There are two types of random variables discrete and continuous.
- The distinction between both is very important for the analysis.

Definition

A rv X is said to be discrete if the possible distinct values x_j of X are either countably infinite or finite.

- Distributions of discrete random variables are described by the probability mass function $P(X = x_j)$.
- Consider the following two examples:
 - ▶ The outcomes of tossing a fair coin with finite distinct values.
 - ▶ Drawing randomly a person and its number of descendants with countably infinite distinct values.



Expectation of Discrete Random Variables

Definition

Let X be a discrete random variable with distinct values $\{x_1, \dots, x_k\}$ and a probability function $P(X = x_j) \in [0, 1]$ for $j \in \{1, \dots, k\}$. Then the expected value of X is defined as

$$E X = \sum_{j=1}^k P(X = x_j)x_j. \quad (24)$$

- For infinite possible outcomes the finite sum becomes an infinite sum.
- The expectation is not defined for all random variables.
- An example is the Cauchy distribution introduced in Section .



Variance of Discrete Random Variables

Definition

Let X be a discrete random variable with distinct values $\{x_1, \dots, x_k\}$ and a probability function $P(X = x_j) \in [0, 1]$ for $j \in \{1, \dots, k\}$. Then the variance of X is defined as

$$\text{Var } X = E(X - EX)^2 = \sum_{j=1}^k P(X = x_j)(x_j - EX)^2. \quad (25)$$

- As for the expectation the variance is not defined for all random variables.
- The variance measures the expected dispersion of a random variable to its expected value.



Continuous Random Variables

Definition

A rv X is said to be continuous if the possible distinct values x_j are uncountably infinite.

- For a continuous random variable the probability density function pdf describes its distribution.
- Selecting randomly a person and its weight is a typical example for a probabilistic experiment, which can be described by a continuous random variable.



Combinatorics and Discrete Distributions

Set Theory

Probabilistic Experiment with Finite Sample Space

Binomial Distribution

Multinomial Distribution

Hypergeometric Distribution

Poisson Distribution

Binomial Distribution

- One of the basic probability distributions is the binomial.
- Examples of this distribution can be observed in daily life:
 - ▶ Tossing a coin to obtain heads or tails.
 - ▶ Trying to score a goal in a football game.



Bernoulli Random Variable

- A **Bernoulli experiment** is a random experiment with two outcomes: success or failure.
- Let p denote the probability of the success of each trial and the random variable X be equal to 1 if the outcome is a success and 0 for a failure.
- Then the probability mass function of X is

$$P(X = 0) = 1 - p,$$

$$P(X = 1) = p.$$

- The expected value and variance of a Bernoulli random variable are $E X = p$ and $\text{Var } X = p(1 - p)$ respectively.



Binomial Distribution

Definition

The **binomial distribution** is the distribution of a random variable X for which

$$P(X = x) = \binom{n}{x} p^x (1 - p)^{n-x}, \quad x \in \{0, 1, 2, \dots, n\}, \quad (26)$$

where n is the number of all trials, x is the number of successful outcomes, p is the probability of success, $\binom{n}{x}$ is the number of possibilities that n outcomes leading to x successes and $n - x$ failures.



Binomial Distribution

- In R, the command `dbinom()` creates a probability mass function.

```
1 > n = 10                      # number of trials
2 > p = 0.2                      # probability of success
3 > dbinom(2, n, p)            # probability mass function at
   X = 2
4 [1] 0.30199
5 > dbinom(0:10, n, p)        # probability at 0, 1, ..., 10
6 [1] 0.107374 0.268435 0.301989 0.201326 0.088080
7 [6] 0.026424 0.005505 0.000786 0.000074 0.000004
8 [11] 0.000000
```



Binomial Distribution

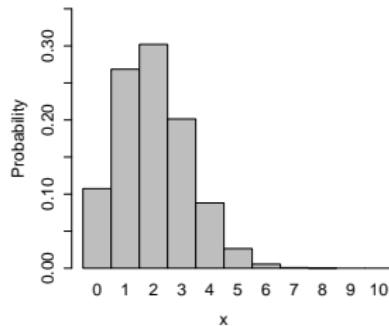


Figure 8: Probability mass function of the binomial distribution with number of trials $n = 10$ and probability of success $p = 0.2$

BCS_Binhist



Combinatorics and Discrete Distributions

Set Theory

Probabilistic Experiment with Finite Sample Space

Binomial Distribution

Multinomial Distribution

Hypergeometric Distribution

Poisson Distribution

Multinomial Distribution

- The binomial experiment always generates two possible outcomes (success or failure) on each trial.
- When tossing a die, the probability of getting the number six is $1/6$, and there is a $5/6$ chance to roll any other number.
- But if we toss several dice together each time, what is the probability to get only a certain number for all dice?



Multinomial Distribution

Definition

Let a random experiment be independently repeated n times, so that it returns one of the fixed k possible outcomes with the probabilities p_1, p_2, \dots, p_k each time. The **multinomial distribution** is a distribution of a vector of random variables $X = (X_1, X_2, \dots, X_k)^\top$ where the X_i 's denote the number of occurrences for which

$$P(X_1 = x_1, X_2 = x_2, \dots, X_k = x_k) = \frac{n!}{x_1! x_2! \cdots x_k!} p_1^{x_1} p_2^{x_2} \cdots p_k^{x_k}, \quad (27)$$

where $p_1, p_2, \dots, p_k > 0$, $\sum_{i=1}^k p_i = 1$, $\sum_{i=1}^k x_i = n$, and x_i is nonnegative for each i .



Multinomial Distribution

Example 7

Suppose we had a box with two red, three green and five blue marbles. We randomly draw three marbles with replacement. What is the probability of drawing one marble of each colour?

Here the realizations of the random variable are

$x_1 = 1, x_2 = 1, x_3 = 1$, and the corresponding probabilities are $p_1 = 0.2, p_2 = 0.3, p_3 = 0.5$. Therefore according to equation (27), the desired probability is

$$P(X_1 = 1, X_2 = 1, X_3 = 1) = \frac{3!}{1! 1! 1!} 0.2^1 0.3^1 0.5^1 = 0.18.$$



Multinomial Distribution

- In R this can be calculated as below.

```
1 > x = c(1, 1, 1)
2 > my.prob = c(0.2, 0.3, 0.5)    # set p's
3 > dmultinom(x, prob = my.prob)
4 [1] 0.18
```

- Each specific random variable X_i , for $i = 1, 2, \dots, k$, follows a binomial distribution, thus the expectation and the variance are $E(X_i) = np_i$ and $\text{Var}(X_i) = np_i(1 - p_i)$ respectively.
- The covariance between any two random variables is $\text{Cov}(X_i, X_j) = -np_i p_j$ for $i \neq j$.



Combinatorics and Discrete Distributions

Set Theory

Probabilistic Experiment with Finite Sample Space

Binomial Distribution

Multinomial Distribution

Hypergeometric Distribution

Poisson Distribution

Hypergeometric Distribution

- In the typical “6 from 49” lottery, 6 numbers from 1 to 49 are chosen without replacement.
- Every time when one number is drawn, the chance of the remaining numbers to be chosen will change.
- This is an example for the **hypergeometric experiment** which satisfies the following requirements:
 1. a sample is randomly selected *without replacement* from a population;
 2. each element of the population is from one of two different groups which can also be defined as success and failure.
- Because the sample is drawn without replacement, the trials in the hypergeometric experiment are not independent.



Hypergeometric Distribution

Definition

A random variable X from the hypergeometric experiment follows the **hypergeometric distribution** $H(n, M, N)$ which has the probability function

$$P(X = x) = H(x; n, M, N) = \frac{\binom{M}{x} \binom{N-M}{n-x}}{\binom{N}{n}}, \quad x = 0, 1, \dots, \min\{M, n\}, \quad (28)$$

where N is the population size, n is the sample size, M is the number of successes in the population and x is the number of successes in the sample.



Hypergeometric Distribution

Example 8

Having a box with 20 marbles, including 10 red and 10 blue marbles, we randomly select 6 marbles without replacement. The probability of getting two red marbles can be calculated using the $H(6, 10, 20)$ distribution. Here the experiment consists of 6 trials, so $n = 6$, and there are 20 marbles in the box, so $N = 20$. $M = 10$, since there are 10 red marbles inside, of which two should be selected, so $x = 2$. Then

$$P(X = 2) = \frac{\binom{10}{2} \binom{20-10}{6-2}}{\binom{20}{6}} = 0.2438.$$



Hypergeometric Distribution

- This example has a straightforward solution in R.

```
1 > M = 10      # number of successes in population
2 > x = 2      # number of successes in sample
3 > N = 20      # population size
4 > n = 6      # sample size
5 > dhyper(x, M, N - M, n)
6 [1] 0.243808
```



Hypergeometric Distribution

- The binomial distribution is a limiting form of the hypergeometric distribution, which pops up when the population size is very large compared to the sample size.
- In this case, it is possible to ignore the “no replacement” problem and approximate a hypergeometric distribution by the binomial distribution.



Hypergeometric Distribution

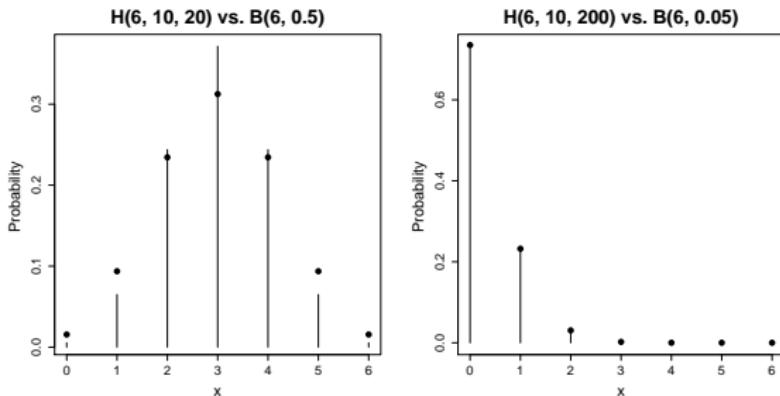


Figure 9: Probability functions of the hypergeometric (lines) vs. binomial distribution (dots)

BCS_Binhyper



Combinatorics and Discrete Distributions

Set Theory

Probabilistic Experiment with Finite Sample Space

Binomial Distribution

Multinomial Distribution

Hypergeometric Distribution

Poisson Distribution

Poisson Distribution

- The Poisson distribution was first discovered in 1898 by the extraordinary economist, statistician and mathematician Ladislaus von Bortkiewicz.
- If we simultaneously toss 100 or 1000 imaginary dices with 100 sides each, how can we calculate the probability of a certain number of dice showing the same face?



Figure 10: Ladislaus von Bortkiewicz (1868-1931)



Poisson Distribution

Definition

A random variable X follows a **Poisson distribution** with parameter λ , denoted as $\text{Pois}(\lambda)$, if

$$P(X = x) = \exp(-\lambda) \cdot \frac{\lambda^x}{x!}, \quad x = 0, 1, 2, \dots \text{ and } \lambda > 0. \quad (29)$$



Poisson Distribution

- Eventually, the limiting distribution for the binomial appears to be the Poisson distribution.

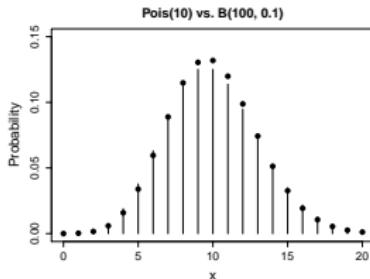


Figure 11: Probabilities of the Poisson (lines) and the Bernoulli distribution (dots)

BCS_Binpois



Poisson Distribution

Example 9

If a typist makes on average one typographical error per page, what is the probability of exactly two errors in a one-page text?

Consider the number of typographical errors per page as rv $X \sim B(n, p)$. Here n is the number of words per page and p the probability of a word to contain a typographical error. It is plausible to assume that n is large and p small, which is sufficient for X to follow the Poisson distribution with $\lambda = E X = 1$. Using equation (29), the probability for two typographical errors per page ($X = 2$) is

$$P(X = 2) = \exp(-1) \cdot \frac{1^2}{2!} = 0.184.$$



Poisson Distribution

- For the example from above the solution can be calculated in R as follows:

```
1 > x = 2
2 > lambda = 1
3 > dpois(x,lambda)    # probability mass function
4 [1] 0.1839397
```



Poisson Distribution

Example 10

The Prussian horsekick fatality dataset from Ladislaus von Bortkiewicz gives us the number of soldiers killed by horsekick in 10 similar corps of the Prussian military over a period of 20 years. Observations on accidents over a total of 200 corps-years are available, as well as $\lambda = 0.61$ estimated by the mean of the 200 observations. What is the probability that there was exactly one fatal horsekick over 20 years in a corps? Since $n = 200$ and $\lambda = 0.61$, $p = \lambda/n = 0.00305$ and the probability can be calculated by

$$P(X = 1) = \text{Pois}(1; 0.61) = \exp(-0.61) \cdot \frac{0.61^1}{1!} = 0.33144,$$

$$P(X = 1) = B(1; 200, 0.00305),$$

$$= \binom{200}{1} 0.00305^1 (1 - 0.00305)^{200-1} = 0.33215.$$



Poisson Distribution

```
1 > x = 1
2 > n = 200
3 > lambda = 0.61
4 > p = lambda / n
5 > dbinom(x,n,p)      # binomial probability mass function
6 [1] 0.3321483
7 > dpois(x,lambda)    # Poisson probability mass function
8 [1] 0.331444
```



Summation of Poisson distributed random variables

- Let $X_i \sim \text{Pois}(\lambda_i)$ be a number of independent and Poisson distributed random variables with parameters $\lambda_1, \lambda_2, \dots, \lambda_n$.
- Then their sum is a rv also following the Poisson distribution with $\lambda = \lambda_1 + \lambda_2 + \dots + \lambda_n$:

$$X_i \sim \text{Pois}(\lambda_i), i = 1, \dots, n \text{ implies } \sum_{i=1}^n X_i \sim \text{Pois}(\lambda_1 + \lambda_2 + \dots + \lambda_n). \quad (30)$$



Summation of Poisson distributed random variables

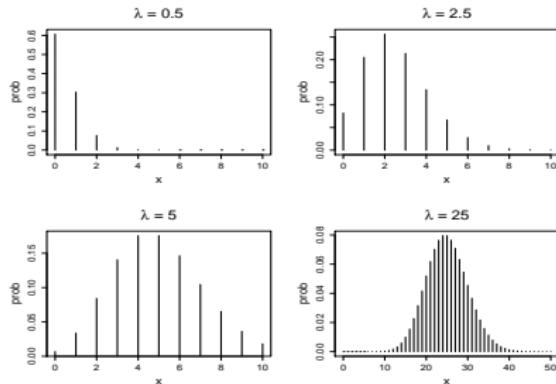


Figure 12: Probability mass functions of the Poisson distribution for different λ

The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

Univariate Distributions

Motivation

Continuous Distribution

The Normal Distribution

Distributions Related to the Normal Distribution

Other Univariate Distributions

Conclusions

Is Everything Normal?

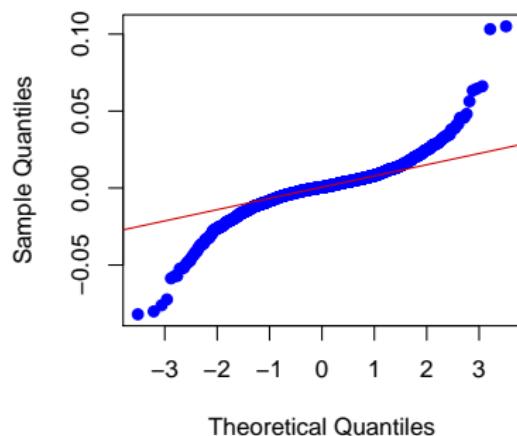
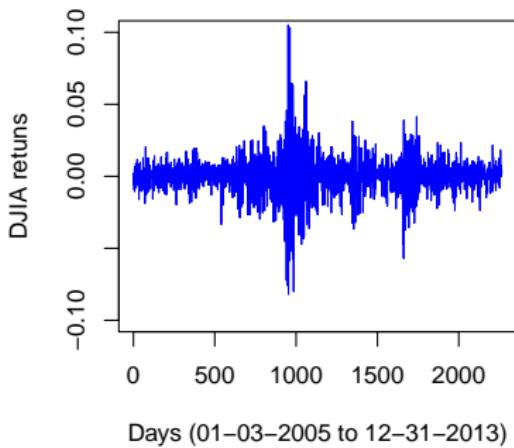


Figure 13: Log-returns of the DJIA daily closing values and QQ-plot for the normal distribution



Testing Mean of the Sample

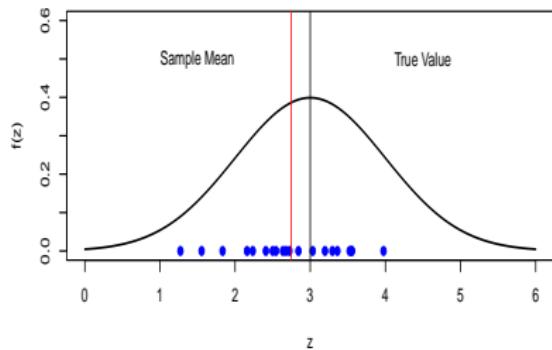


Figure 14: Sample Mean Test

- variance known
 $\frac{\bar{x}-\mu}{\sigma n^{1/2}} \sim N(0, 1)$
- variance unknown
 $\frac{\bar{x}-\mu}{\sigma n^{1/2}} \sim ?$



Univariate Distributions

Motivation

Continuous Distribution

The Normal Distribution

Distributions Related to the Normal Distribution

Other Univariate Distributions

Conclusions

The CDF and Density Function of a Random Variable

Definition

Let X be a continuous random variable. The mapping $F_X : \mathbb{R} \rightarrow [0, 1]$, defined by $F_X(x) = P(X \leq x)$, is called the cumulative distribution function (cdf) of the random variable X .

Definition

A mapping $f_X : \mathbb{R} \rightarrow \mathbb{R}_+$ is called the probability density function (pdf) of a random variable (rv) X if $f_X(x) = \frac{dF_X(x)}{dx}$ exists for all x and $\int_{-\infty}^{\infty} f_X(x) dx$ exists and takes on the value one.



Expectation and Variance

Definition

Let X be a continuous random variable with a density function $f_X(x)$. Then the expected value of X is defined as

$$\mathbb{E} X = \int_{-\infty}^{\infty} xf_X(x) dx. \quad (31)$$

Definition

If the expected value of X exists, the variance is defined as

$$\text{Var } X = \int_{-\infty}^{\infty} (x - \mathbb{E} X)^2 f_X(x) dx, \quad (32)$$

and the standard deviation is $\sigma_X = \sqrt{\text{Var } X}$.



Skewness and Excess Kurtosis

Definition

The skewness of a random variable X is defined as

$$S(X) = E \left\{ (X - E X)^3 \right\} / \sigma_X^3. \quad (33)$$

Definition

The excess kurtosis of a random variable X is defined as

$$K(X) = \frac{E \left\{ (X - E X)^4 \right\}}{\sigma_X^4} - 3. \quad (34)$$



Univariate Distributions

Motivation

Continuous Distribution

The Normal Distribution

Distributions Related to the Normal Distribution

Other Univariate Distributions

Conclusions

The Normal Distribution

- The normal distribution is considered the most prominent distribution in statistics.
- It is a continuous probability distribution that has a bell-shaped probability density function, also known as the Gaussian function.



Figure 15: Johann Carl Friedrich Gauss (1777–1855)



Normal Random Variable

Definition

An rv X with the Gaussian pdf

$$\varphi(x, \mu, \sigma^2) = (2\pi\sigma^2)^{-1/2} \exp \left\{ -(x - \mu)^2 / (2\sigma^2) \right\}$$

is said to be normally distributed with $\text{E } X = \mu$ and $\text{Var } X = \sigma^2$ and is written as $X \sim N(\mu, \sigma^2)$. If $\mu = 0$ and $\sigma^2 = 1$, then φ is called the standard normal distribution, and we abbreviate $\varphi(x, 0, 1)$ as $\varphi(x)$.



The CDF and PDF

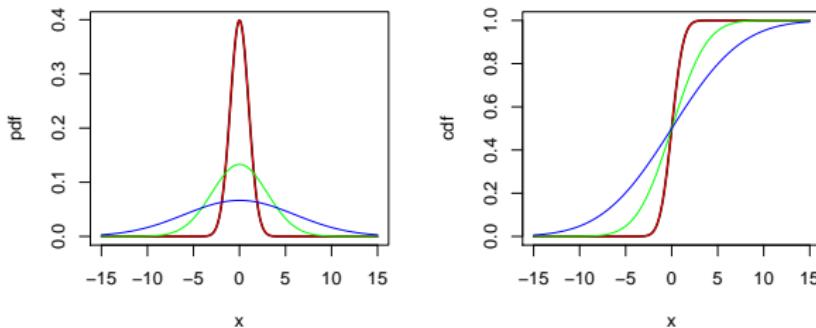


Figure 16: pdf and cdf of the normal distribution (for $\mu = 0$ and $\sigma^2 = 1$,
 $\sigma^2 = 3$, $\sigma^2 = 6$ respectively)

BCS_NormPdfCdf



The CDF and PDF

```
1 > z = -3:3           # values for the random variables
2 > round(dnorm(z), 2) # density or pdf of standard normal
3 [1] 0.00 0.05 0.24 0.40 0.24 0.05 0.00
4 > round(pnorm(z), 2) # cdf of standard normal
5 [1] 0.00 0.02 0.16 0.50 0.84 0.98 1.00
```

- The normal distribution is symmetric.
- For the standard normal distribution the expected value and the median are the same (0)
- It is true that $\varphi(z) = \varphi(-z)$.



Univariate Distributions

Motivation

Continuous Distribution

The Normal Distribution

Distributions Related to the Normal Distribution

Other Univariate Distributions

Conclusions

χ^2 distribution

- $X_i, i = 1, \dots, n$ is rv and $X_i \sim N(0, 1)$ for all i
- Then (with n degrees of freedom)

$$Z = \sum_{i=1}^n X_i^2 \sim \chi_n^2$$



χ^2 distribution

- Probability density function

$$f(z, n) = \frac{2^{-n/2} z^{n/2-1} e^{-z/2}}{\Gamma(n/2)}$$

- Gamma function, $\Gamma(k) = \int_0^\infty t^{k-1} e^{-t} dt.$

- Related R functions

- ▶ `dchisq(z,n)`
- ▶ `pchisq(z,n)`



χ^2 distribution

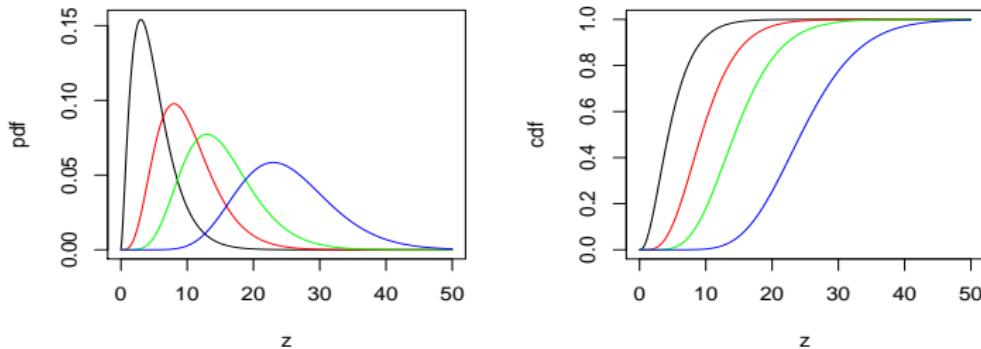


Figure 17: Distribution and density functions for $rv Z$. Degrees of freedom are $n = 5$, $n = 10$, $n = 15$ and $n = 25$ respectively.



BCS_ChiPdfCdf



χ^2 distribution

■ Distribution properties

- ▶ Mean and variance: $E[X] = n$, $\text{Var}[X] = 2n$
- ▶ Mode: $M = n - 2$, for $n \geq 2$
- ▶ Skewness and excess kurtosis: $S = 2\sqrt{\frac{2}{n}}$, $K = \frac{12}{n}$
- ▶ Stable distribution: sum of r chi-squared distributed independent rv x_1, x_2, \dots, x_r , with n_1, n_2, \dots, n_r degrees of freedom, is chi-squared with $n = n_1 + n_2 + \dots + n_r$



Student's *t*-distribution

- $X \sim N(0, 1)$ and $Y \sim \chi_n^2$ independent rv
- Random variable Z that is *t*-distributed, can be formalized as:

$$Z = \frac{X}{\sqrt{Y/n}} \sim t_{n-1}$$



Student's *t*-distribution

- Probability density function

$$f(z, n) = \frac{\Gamma\left(\frac{n+1}{2}\right)}{\sqrt{\pi n} \Gamma(n/2) \left(1 + \frac{z^2}{n}\right)^{\frac{n+1}{2}}}$$

- Related R functions

- ▶ `dt(z,n)`
- ▶ `pt(z,n)`



Student's t -distribution

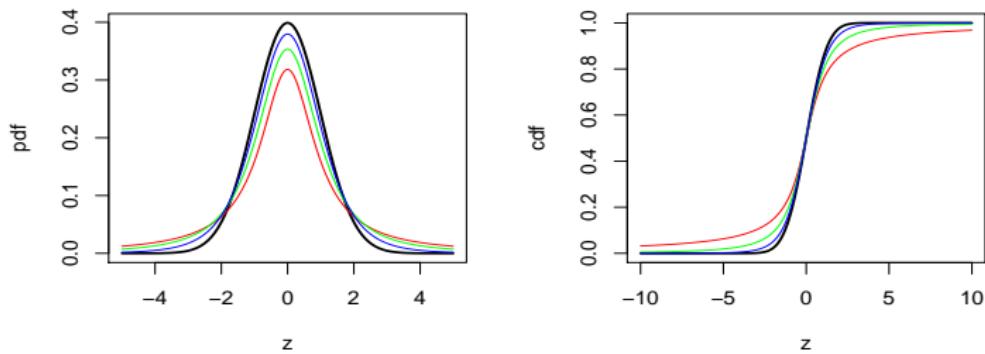


Figure 18: Pdf and cdf of Student's t -distribution for $n = 1$, $n = 2$ and $n = 5$ respectively. The normal distribution is depicted by the black line.

Student's *t*-distribution

Distribution properties

- ▶ $E[X] = 0$, $\text{Var}[X] = \frac{n}{n-2}$, for $n > 2$
- ▶ $M = 0$, $S = 0$ (for $n > 3$), $K = \frac{3(n-2)}{n-4}$ (for $n > 4$)
- ▶ Asymptotically normally distributed



F-distribution

- random variable Z has the Fisher-Snedecor distribution(F-distribution) with n and m degrees of freedom, if:

$$Z = \frac{\chi_n^2/n}{\chi_m^2/m} \sim F_{n,m}$$

- χ_n^2 and χ_m^2 independent rv.

Ronald Aylmer Fisher on BBI:



F-distribution

- Probability density function

$$d(z) = \left(\frac{n}{m}\right)^{\frac{n}{2}} \frac{\Gamma\left(\frac{n+m}{2}\right) z^{\left(\frac{n}{2}-1\right)}}{\Gamma\left(\frac{n}{2}\right) \Gamma\left(\frac{m}{2}\right) \left(1 + \frac{zn}{m}\right)^{\frac{n+m}{2}}}$$

- Related R functions

- ▶ `df(z, n, m)`
- ▶ `pf(z, n, m)`



F-distribution

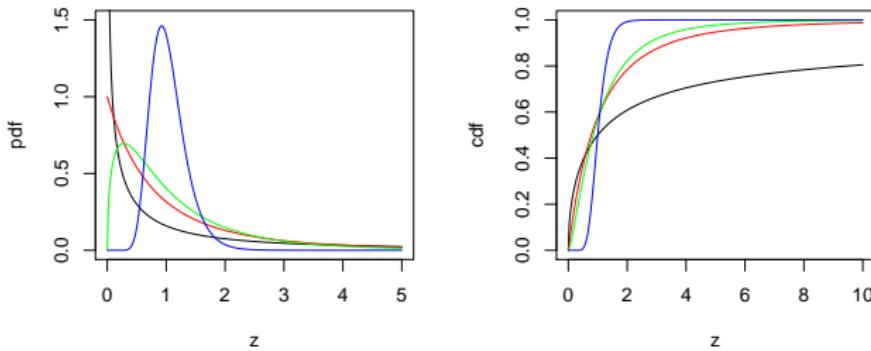


Figure 19: Density and cumulative distribution of F-distribution ($n = 1$, $m = 1$, $n = 2$, $m = 6$, $n = 3$, $m = 10$ and $n = 50$, $m = 50$ respectively)

F-distribution

□ Distribution properties

- ▶ $E[Z] = \frac{m}{m-2}$, for $m > 2$, $\text{Var}[Z] = \frac{2m^2(n+m-2)}{n(m-2)^2(m-4)}$, for $m > 4$
- ▶ $M = \frac{m(n-2)}{n(m+2)}$, for $n > 2$, $S = \frac{(2n+m-2)\sqrt{8(m-4)}}{(m-6)\sqrt{n(n+m-2)}}$ for $m > 6$
- ▶ Z is asymptotically $N\left(\frac{n-m}{2nm}, \frac{n+m}{2nm}\right)$



Univariate Distributions

Motivation

Continuous Distribution

The Normal Distribution

Distributions Related to the Normal Distribution

Other Univariate Distributions

Conclusions

Exponential Distribution

- Probability density function

$$f(z, \lambda) = \begin{cases} \lambda e^{-\lambda z}, & \text{for } z \geq 0 \\ 0, & \text{for } z < 0 \end{cases}$$

- Related R functions

- ▶ `dexp(z, lambda)`
- ▶ `pexp(z, lambda)`



Exponential Distribution

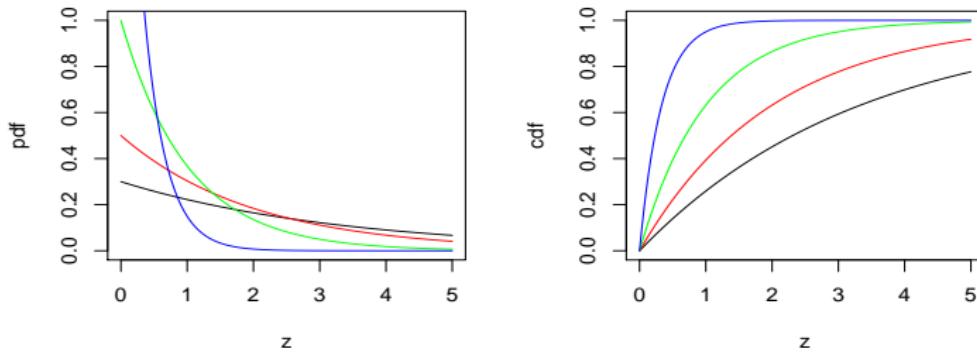


Figure 20: Pdf and cdf of the exponential distribution ($\lambda = 0.3$, $\lambda = 0.5$, $\lambda = 1$ and $\lambda = 3$)

BCS_ExpPdfCdf



Exponential Distribution

- Distribution properties

- ▶ $E[X] = \frac{1}{\lambda}$, $\text{Var}[Z] = \frac{1}{\lambda^2}$
- ▶ $M = 0$, $S = 2$ and $K = 6$
- ▶ memoryless: $P(x \leq t + q | x > t) = P(x \leq t)$



Stable Distributions

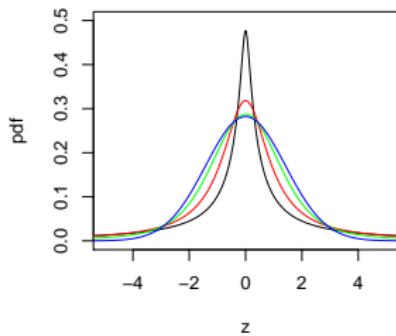
- Z is stable, if

$$aZ_1 + bZ_2 = cZ + d$$

- Z_1 and Z_2 are independent *rv* with same distribution as Z
- a and b are constants
- Z is strictly stable if this property holds for $d = 0$



Stable Distributions



$$\alpha \in (0.6, 1, 1.5, 2), \beta = 0$$

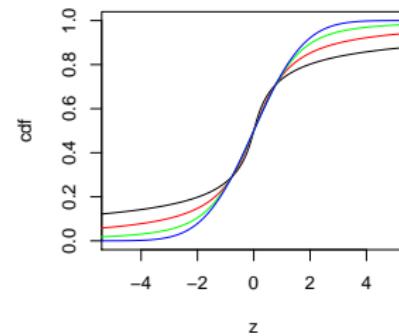
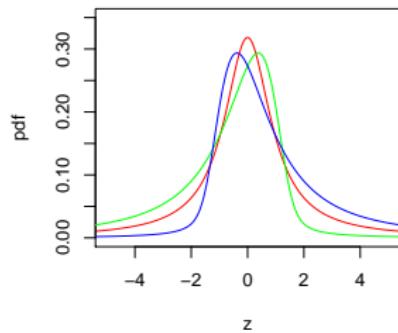


Figure 21: Stable distribution functions depending on tail parameter α . 

BCS_StablePdfCdf



Stable Distributions



$$\alpha = 1, \beta \in (0, -0.8, 0.8)$$

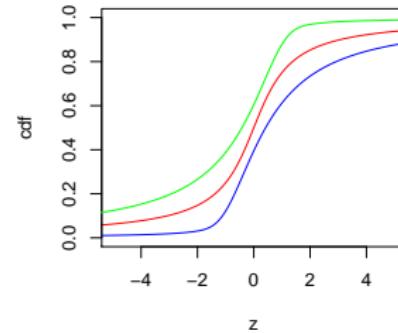


Figure 22: Stable distribution functions depending on skewness parameter

β

BCS_StablePdfCdf

BCS



Stable Distributions

- The characteristic function and its parameters given by Samorodnitsky and Taqqu, Weron:

$$\log \varphi_Z(t, \alpha, \beta, \sigma_S, \mu) = \begin{cases} -\sigma_S^\alpha |t|^\alpha (1 - i\beta \text{sign}(t) \tan \frac{\pi\alpha}{2}) + i\mu t, & \alpha \neq 1, \\ -\sigma_S |t| (1 + i\beta \text{sign}(t) \frac{2}{\pi} \log |t|) + i\mu t, & \alpha = 1. \end{cases}$$

- Related R functions
 - ▶ `dstable(z, alpha, beta, sigma, mu, pm)`
 - ▶ `pstable(z, alpha, beta, sigma, mu, pm)`
- `library fBasics` is required



Stable Distributions

Distribution properties

- ▶ All stable distributions are infinitely divisible
- ▶ Stable distributions are leptokurtic and heavy-tailed distributions (except normal distribution, when $\alpha = 2$)



Cauchy Distribution

- Probability density function

$$d(z; \mu, \sigma) = \frac{1}{\pi} \frac{\sigma}{(z - \mu)^2 + \sigma^2}$$

μ is a location parameter, σ is a scale parameter

- Related R functions

- ▶ `dcauchy(z, mu, lambda)`
- ▶ `pcauchy(z, mu, lambda)`



Cauchy Distribution

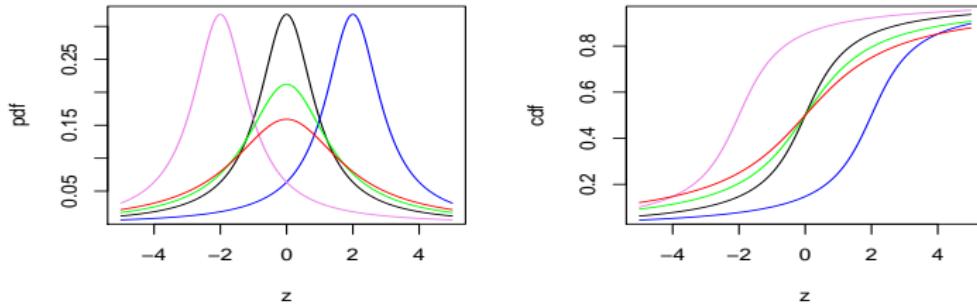


Figure 23: Cauchy distribution functions and respective density functions
($\mu = -2, \sigma = 1$; $\mu = 0, \sigma = 1$; $\mu = 2, \sigma = 1$; $\mu = 0, \sigma = 1.5$; $\mu = 0, \sigma = 2$)

BCS_CauchyPdfCdf



Cauchy Distribution

- Distribution properties

- ▶ Expected value, variance, skewness, kurtosis and other higher moments are undefined
- ▶ Mode and median equal to location parameter μ
- ▶ $U \sim N(0, 1)$ and $V \sim N(0, 1)$ are independent rv, then U/V has the standard Cauchy distribution, with $\mu = 0$ and $\sigma = 1$
- ▶ The standard Cauchy distribution is strictly stable



Univariate Distributions

Motivation

Continuous Distribution

The Normal Distribution

Distributions Related to the Normal Distribution

Other Univariate Distributions

Conclusions

Conclusions

- Distributions related to the normal one
 - ▶ χ^2 distribution
 - ▶ Student's t -distribution
 - ▶ F-distribution
- Other univariate distributions ✓
 - ▶ Exponential
 - ▶ Stable
 - ▶ Cauchy



The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

Univariate Statistical Analysis

Descriptive Statistics

Confidence Intervals and Hypotheses Testing

Linear Regression

Absolute and Relative Frequency

- Having realizations $\{x_i\}$, $i \in 1, \dots, n$ of a random variable X , with $\{a_j\}$, $j \in 1, \dots, k$, denoting all possible but different realizations of X in the sample, then:
- The **absolute frequency** of a_j , denoted by $n(a_j)$, is the number of occurrences of a_j in the sample $\{x_i\}$.
- The **relative frequency** of a_j , denoted by $h(a_j)$, is the ratio of the absolute frequencies of a_j and the sample size n :
$$h(a_j) = n(a_j)/n.$$
 Clearly $\sum_{j=1}^k h(a_j) = 1.$



Graphical Data Representation

For qualitative or discrete variables:

- Bar Plot
- Bar Diagram
- Pie Chart

For continuous (or quasi-continuous) variables:

- Histogram



Bar Plot and Bar Diagram

Bar Diagram:

```
1 > # with absolute frequencies  
2 > plot(table(chickwts$feed))  
3 > # with relative frequencies  
4 > plot(table(chickwts$feed)/length(chickwts$feed))
```

Bar Plot:

```
1 > # with absolute frequencies  
2 > barplot(table(chickwts$feed))  
3 > # with relative frequencies  
4 > barplot(table(chickwts$feed)/length(chickwts$feed))
```



Example: Bar Diagram and Bar Plot

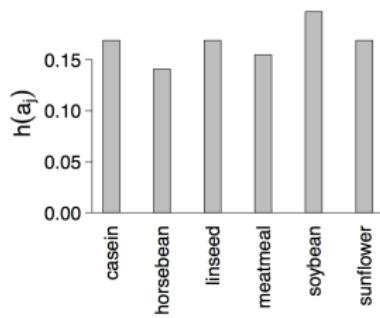
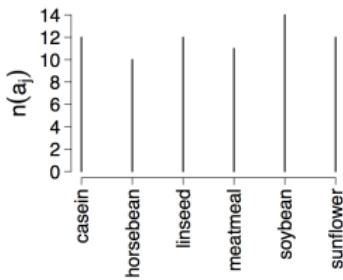


Figure 24: Bar Diagram of the Absolute Frequencies $n(a_j)$ (left) and Bar Plot of the Relative Frequencies $h(a_j)$ (right) of chickwts\$feed

Example: Pie Chart

```
1 > pie(table(chickwts$feed))
```

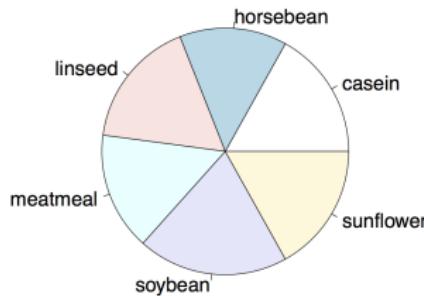


Figure 25: Pie Chart of the Data chickwts\$feed

Example: Histogram

Histogram:

```
1 > hist(nhtemp, freq = FALSE)
2 > hist(nhtemp, freq = FALSE, breaks = seq(47, 55, 0.5))
```



Example: Histogram

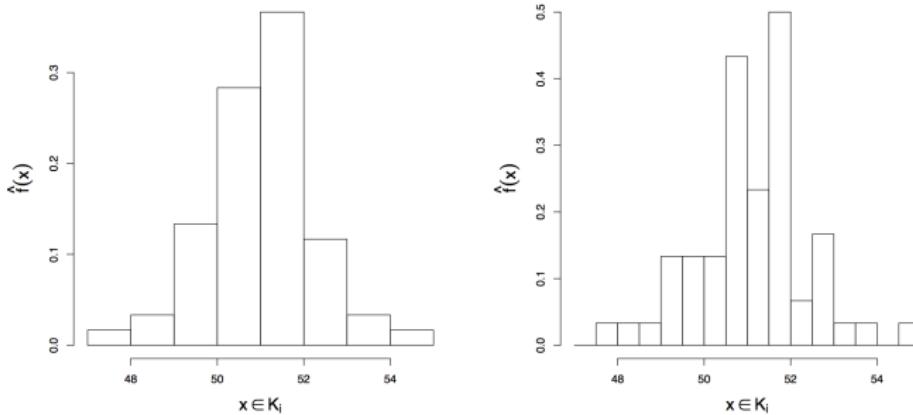


Figure 26: Histograms of the data nhtemp

BCS_hist1

BCS_hist2



Example: Histogram

Ecdf:

```
1 > ecdf(Formaldehyde$car)
2 Empirical CDF
3 Call: ecdf(Formaldehyde$car)
4 x[1:6] = 0.1, 0.3, 0.5, ..., 0.7, 0.9
5 > plot(ecdf(Formaldehyde$car))
```



Example: Ecdf

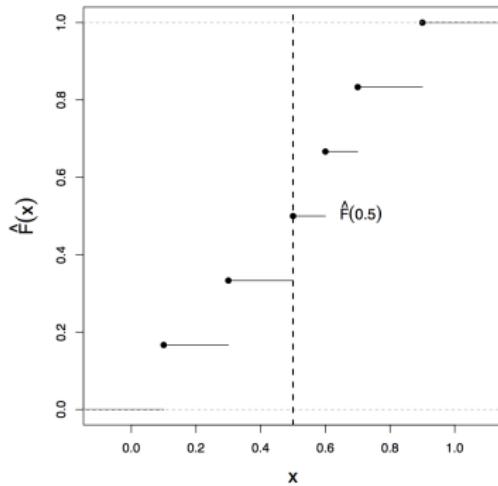


Figure 27: Ecdf of the data Formaldehyde

Location Parameters

Location parameters describe the center of a distribution through a numerical value. They can be quantified in different ways and visualised particularly well by boxplots.

- Arithmetic Mean
- α -trimmed Mean
- Quantiles
- Mode



Arithmetic Mean

Definition

The sample mean for a sample of n values, x_1, x_2, \dots, x_n is defined by

$$\bar{x} = n^{-1} \sum_{i=1}^n x_i.$$

- According to the *Law of Large Numbers*, if $\{X_i\}_{i=1}^n$ denotes n i.i.d. random variables with finite expected values $E(X_i) = \mu$, for all i , then $n^{-1} \sum_{i=1}^n X_i \xrightarrow{\text{a.s.}} \mu$ when $n \rightarrow \infty$.



α -trimmed Mean

Definition

The α -trimmed mean is the arithmetic mean computed after trimming the fraction α of the smallest and largest observations of X , given by

$$\bar{x}_\alpha = \frac{1}{n - 2\lfloor n\alpha \rfloor} \sum_{i=\lfloor n\alpha \rfloor + 1}^{n - \lfloor n\alpha \rfloor} x_{(i)}$$

with $\alpha \in [0, 0.5)$.

```
1 > mean(nhtemp, trim=0.2)
2 [1] 51.22222
3 > mean(nhtemp, trim=0.4)
4 [1] 51.225
```



Quantiles

Definition

The p -quantile \tilde{x}_p with $0 \leq p \leq 1$ is a value such that at most $100 \cdot p\%$ of the observations are less or equal to \tilde{x}_p and $100 \cdot (1 - p)\%$ are greater or equal to \tilde{x}_p . The number of observations, which are smaller than or equal to \tilde{x}_p is then equal to $\lceil np \rceil$.

$$\tilde{x}_p = \begin{cases} x_{(\lceil np \rceil)} & \text{if } np \notin \mathbb{Z} \\ \frac{1}{2} \{x_{(np)} + x_{(np+1)}\} & \text{if } np \in \mathbb{Z} \end{cases} \quad \text{for } p \in [0, 1].$$



Mode

- The mode is the most frequently occurring observation value in a data set (also called the most fashionable observation).
- Together with the mean and median, one can use it as an indicator of data skewness.
- In general, the mode is not equal to the mean or median and the difference depends on the skewness.

Definition

The mode is defined by

$$x_{mod} = a_j, \quad \text{with } n(x_{mod}) > n(a_j), \quad \forall j \in \{1, \dots, k\},$$

where $n(x)$ is the absolute frequency of x .



Dispersion Parameters

- Evaluating the measures of dispersion in addition to the measures of location provides a more complete description of the data.
 - ▶ Total Range
 - ▶ Interquartile Range
 - ▶ Variance
 - ▶ Standard Deviation
 - ▶ Median Absolute Deviation



Total Range

Definition

The sample total range is the difference between the sample maximum and the sample minimum, i.e.

$$\text{TotalRange} = x_{(n)} - x_{(1)}.$$

```
1 > range(nhtemp) # sample min and sample max
2 [1] 47.9 54.6
3 > # difference between maximum and minimum of data
4 > totalrange = diff(range(nhtemp))
5 > totalrange
6 [1] 6.7
```



Interquartile Range

Definition

The sample interquartile range (IQR) is the first difference between the upper quartile $\tilde{x}_{0.75}$ and the lower quartile $\tilde{x}_{0.25}$, i.e.

$$\text{IQR} = \tilde{x}_{0.75} - \tilde{x}_{0.25}.$$

```
1 > LUQ = quantile(nhtemp, probs = c(0.25,0.75)); LUQ
2 25%    75%
3 50.575 51.900
4 > IQR = diff(LUQ); IQR
5 75%
6 1.325
```



Variance

Definition

The **sample variance** for a sample of n values x_1, x_2, \dots, x_n is the averaged sum of the squared deviations from the sample mean \bar{x} , or

$$\tilde{s}^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2.$$

- The unbiased **variance estimator** of the population variance for a sample of n values x_1, x_2, \dots, x_n is

$$s^2 = \frac{n}{n-1} \tilde{s}^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2.$$



Standard Deviation

- As the variance is, due to the squares, not on the same scale as the original data, it is useful to introduce a normalised dispersion measure.

Definition

The sample standard deviation \tilde{s} and the estimator for the population standard deviation s based on the unbiased variance estimator are calculated as:

$$\tilde{s} = \sqrt{\tilde{s}^2}, \quad s = \sqrt{s^2}.$$



Median Absolute Deviation

Definition

The **median absolute deviation** (MAD) is the median of the absolute deviations from the median :

$$d_i = |x_i - \tilde{x}_{0.5}|,$$

$$MAD = \text{med}(d_i), \quad \forall i \in \{1, \dots, n\}.$$

- Besides the MAD, an alternative robust approach would be to measure the average absolute distance of each realisation from the median or the mean, i.e.

$$d_1 = \frac{1}{n} \sum_{i=1}^n |x_i - \tilde{x}_{0.5}| \quad \text{or} \quad d_2 = \frac{1}{n} \sum_{i=1}^n |x_i - \bar{x}|.$$



Box-Plot

- The boxplot (or box-whisker-plot) is a diagram which describes the distribution of a given data set.
- It summarises the location and dispersion measures discussed previously.
- The boxplot gives a quick glimpse of the observation values range and their empirical distribution.



Example: Box-Plot

```
1 > boxplot(nhtemp)
```

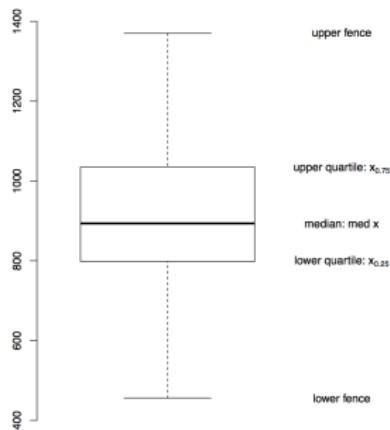


Figure 28: Box-Plot of the data nhtemp

Box-Plot Interpretation

Example 11

Approximately fifty percent of the observations are contained in the box. The upper edge is the 0.75-quantile and the lower edge is the 0.25-quantile. The distance between these two edges is the interquartile range (IQR). The median is indicated by the horizontal line between the two edges. If its distance from the upper edge is not equal to its distance from the lower edge, then the data are skewed.



Univariate Statistical Analysis

Descriptive Statistics

Confidence Intervals and Hypotheses Testing

Linear Regression

Introduction: Confidence Intervals

- When estimating a population parameter θ (e.g. population mean μ , variance σ^2), it is important to have some clue about the precision of the estimation.
- The precision in this context is the probability that the estimate $\hat{\theta}$ is wrong by less than plus minus a given amount, which can be calculated based on a random sample of size n drawn from a population.
- For most cases, like $\theta = \mu$, the sample size n must be large enough that $\hat{\theta}$ can be assumed to be normal distributed (CLT).



Confidence Intervals

Definition

The **confidence interval (CI)** for θ of a continuous random variable is a range of feasible values for an unknown θ together with a confidence coefficient $(1 - \alpha)$ conveying one's confidence that the true θ actually lies in the interval. Formally, it is written as

$$P(\theta \in \text{CI}) = 1 - \alpha.$$



Confidence Intervals for θ when σ is known

Example 12

Consider a normal population with an unknown mean μ and a known standard deviation σ . If we take n realizations x_1, x_2, \dots, x_n of a random variable from this population and calculate $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$, then

$\bar{X} \sim N(\mu, \sigma/\sqrt{n})$ and $\sqrt{n} \left(\frac{\bar{X} - \mu}{\sigma} \right) \sim N(0, 1)$.

Now, let $Z \sim N(0, 1)$ with such $z_{1-\frac{\alpha}{2}}$ that

$$P(-z_{1-\frac{\alpha}{2}} \leq Z \leq z_{1-\frac{\alpha}{2}}) = 1 - \alpha.$$

With $Z = \sqrt{n} \cdot \frac{\bar{X} - \mu}{\sigma}$, this implies

$$P \left(\bar{X} - z_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}} \leq \mu \leq \bar{X} + z_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}} \right) = 1 - \alpha.$$



Confidence Intervals for θ when σ is known

- Thus, for a fixed $\alpha \in [0, 1]$, the confidence coefficient is $(1 - \alpha)$ and the corresponding $100 \cdot (1 - \alpha)\%$ -confidence interval for the population mean μ when the population is normally distributed and σ is known is given by

$$\left[\bar{x} - z_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}}; \bar{x} + z_{1-\frac{\alpha}{2}} \cdot \frac{\sigma}{\sqrt{n}} \right].$$



Example: Confidence Intervals

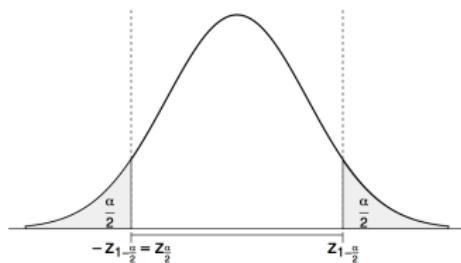


Figure 29: The $100 \cdot (1 - \alpha)$ -confidence interval is the interval between $z_{\frac{\alpha}{2}}$ and $z_{1-\frac{\alpha}{2}}$. The area under the curve within this interval is equal to $1 - \alpha$.

BCS_Conf2sided



Example: One-sided confidence intervals

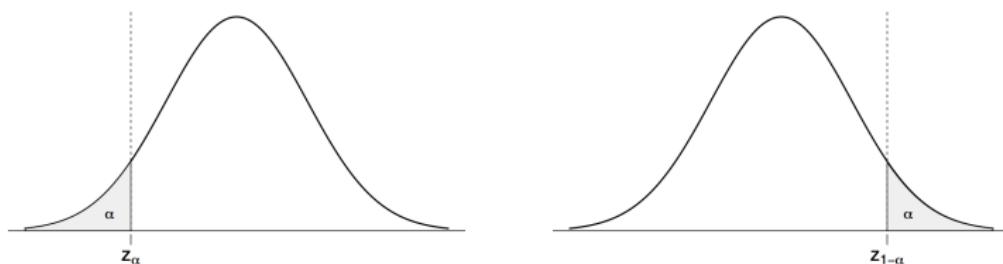


Figure 30: Two types of one-sided confidence intervals: upper tail (left) and lower tail (right) confidence interval

BCS_Conf1sidedleft

BCS_Conf1sidedright



Confidence Intervals for μ when σ is unknown

- Since the population's σ is generally unknown, the construction of confidence intervals for μ will usually be based on the Student's t -distribution.
- Define $V \sim t_v$ (V is t -distributed with v degrees of freedom) such that

$$P\left(-t_{1-\frac{\alpha}{2},v} \leq V \leq t_{1-\frac{\alpha}{2},v}\right) = 1 - \alpha.$$

- Now, assuming that the population is normally distributed, $V = \sqrt{n} \frac{\bar{X} - \mu}{S}$ and the sample size is n , it follows that

$$P\left(\bar{X} - t_{1-\frac{\alpha}{2},n-1} \cdot \frac{S}{\sqrt{n}} \leq \mu \leq \bar{X} + t_{1-\frac{\alpha}{2},n-1} \cdot \frac{S}{\sqrt{n}}\right) = 1 - \alpha.$$



Confidence Intervals for μ when σ is unknown

Definition

The $100 \cdot (1 - \alpha)\%$ -confidence interval for the population mean μ when the population is normal distributed and σ is **unknown** is defined by

$$\left[\bar{X} - t_{1-\frac{\alpha}{2}, n-1} \cdot \frac{S}{\sqrt{n}}; \bar{X} + t_{1-\frac{\alpha}{2}, n-1} \cdot \frac{S}{\sqrt{n}} \right].$$



Tests of Hypotheses

- A test of hypotheses, or *test of significance*, is a widely used tool in statistical analysis.
- Based on the information gained from a sample, one attempts to make a decision for a hypothesis.
- The hypotheses in this case are assumptions about the parameters of a distribution which a random variable follows in the population (e.g. mean μ , proportion p , variance σ^2 or standard deviation σ , etc.).



Tests of Hypotheses

- When conducting a test, two hypotheses are made, namely a null hypothesis H_0 and an alternative hypothesis H_1 (or often also H_a).
- Different hypotheses can for example be:

$$H_0 : \theta = \theta_0 \quad \text{vs} \quad H_1 : \theta \neq \theta_0.$$

or

$$H_0 : \theta \leq \theta_0 \quad \text{vs} \quad H_1 : \theta > \theta_0.$$



Two Types of Risk for Statistical Tests

Type I error is the error of rejecting a null hypothesis when it is in fact true.

Type II error is the error of failing to reject a null hypothesis when it is actually not true.



Critical Region

Definition

The critical region (or the region of rejection) is the set of values of \bar{X} that cause rejection of the null hypothesis H_0 . It can be determined with the distribution of the sample mean \bar{X} . The probability of a type I error (also called the **significance level** of a test), i.e. the probability of rejecting the null hypothesis H_0 , although it is true, should be at most α :

$$P(\bar{X} \in \text{critical region} \mid H_0) = \alpha.$$

The construction of the critical region depends on the type of test one conducts (two-tailed or one-tailed).



Two-tailed Tests

- Hypotheses:

$$H_0 : \mu = \mu_0 \quad \text{vs} \quad H_1 : \mu \neq \mu_0.$$

- Method:

1. Transform $(\bar{X} - \mu_0)$ into a standard normal deviate

$$Z = n^{1/2} \frac{\bar{X} - \mu_0}{\sigma} \sim N(0, 1).$$

2. Reject the null hypothesis $H_0 : \mu = \mu_0$ if z as a realization of Z fulfills

$$Z \in \text{critical region} \equiv (-\infty, -z_{1-\frac{\alpha}{2}}) \cup (z_{1-\frac{\alpha}{2}}, +\infty).$$



One-tailed Tests

- Test that μ is greater than an arbitrary value μ_0 .
 - ▶ Hypotheses: $H_0 : \mu \leq \mu_0$ vs $H_1 : \mu > \mu_0$.
 - ▶ Reject H_0 when $Z = \sqrt{n} \frac{\bar{x} - \mu_0}{\sigma} > z_{1-\alpha}$.
- Test that μ is smaller than an arbitrary value μ_0 .
 - ▶ Hypotheses: $H_0 : \mu \geq \mu_0$ vs $H_1 : \mu < \mu_0$.
 - ▶ Reject H_0 when $Z = \sqrt{n} \frac{\bar{x} - \mu_0}{\sigma} < z_\alpha$.



Example: Manual Hypothesis Testing in R

Example 13

For the following example, consider `nottem`, a sample of size $n = 240$ containing average monthly air temperatures at Nottingham Castle in degrees Fahrenheit for the period 1920–1939, which is then converted to Celsius. Supposing the standard deviation in the population $\sigma = 8.6$, one desires to test whether the average monthly temperatures in Nottingham is equal to $\mu = 60$ using the sample mean $\bar{x} = 49.04$. The hypotheses are written as follows:



Example: Manual Hypothesis Testing in R

- The hypotheses are written as follows:

$$H_0 : \mu = 60 \quad \text{vs} \quad H_1 : \mu \neq 60 \quad (\text{for two-sided test}),$$
$$H_0 : \mu \geq 60 \quad \text{vs} \quad H_1 : \mu < 60 \quad (\text{for left-sided test}),$$
$$H_0 : \mu \leq 60 \quad \text{vs} \quad H_1 : \mu > 60 \quad (\text{for right-sided test}).$$

```
1 > Z = ( mean ( nottem ) - 60 ) / ( 8.6 / sqrt ( length ( nottem ) ) )
2 > Z
3 [1] -19.74396
```



Test for the Mean of a Normal Population $(\sigma$ unknown)

- The decision rules for the hypothesis testing are similar to those of the test for the mean when σ is known, except that the standard normal distribution $N(0, 1)$ is replaced by the Student's t -distribution with $n - 1$ degrees of freedom, where n is the sample size.
- An indispensable assumption for small samples is that the random variable is approximately normally distributed.
- If this is the case, then the test statistic follows Student's t distribution.
- In large samples, this assumption is no longer obligatory since the sample means are normally distributed according to the CLT.



Test for the Mean of a Normal Population (σ unknown)

- The rule for **two-sided tests** is to reject the null hypothesis $H_0 : \mu = \mu_0$ if

$$T = \sqrt{n} \frac{\bar{x} - \mu}{s} \in \text{critical region} \equiv$$

$$(-\infty, -t_{n-1, 1-\frac{\alpha}{2}}) \cup (t_{n-1, 1-\frac{\alpha}{2}}, +\infty),$$

that is to say $|T| > t_{n-1, 1-\frac{\alpha}{2}}$.

- The value $t_{n-1, 1-\frac{\alpha}{2}}$ is determined such that

$$P(T \leq t_{n-1, 1-\frac{\alpha}{2}}) = 1 - \frac{\alpha}{2}, \quad \text{where } T \sim t_{n-1}.$$



Test for the Mean of a Normal Population (σ unknown)

- The rule for **one-sided tests** is to reject the null hypothesis

$$H_0 : \mu \leq (\geq) \mu_0 \quad \text{vs} \quad H_1 : \mu > (<) \mu_0,$$

if

$$T = \sqrt{n} \frac{\bar{x} - \mu_0}{s} > (<) t_{n-1, 1-\alpha}.$$



Testing σ^2 of a Normal Population

- How to construct confidence intervals for σ^2 from the estimator s^2 ?

- If X_1, \dots, X_n are i.i.d. random normal variables, then:

$$\frac{(n-1)S^2}{\sigma^2} \sim \chi_{n-1}^2, \quad \text{with } S^2 = \frac{1}{n-1} \sum_{i=1}^n (X_i - \bar{x})^2.$$



Confidence Intervals for σ^2 (Part I)

- Let $Y \sim \chi_{\nu}^2$. Now, choose $\chi_{\nu,1-\alpha}^2$ such that

$$P(Y \geq \chi_{\nu,1-\alpha}^2) = \alpha.$$

and $\chi_{\nu,\alpha}^2$ such that

$$P(Y \leq \chi_{\nu,\alpha}^2) = \alpha.$$



Confidence Intervals for σ^2 (Part II)

- Since $Y = \frac{v \cdot S^2}{\sigma^2}$, it is easy to show that

$$P\left(\chi_{\nu, \frac{\alpha}{2}}^2 < \frac{vS^2}{\sigma^2} < \chi_{\nu, 1-\frac{\alpha}{2}}^2\right) = 1 - \alpha,$$

which is equivalent to

$$P\left(\frac{vS^2}{\chi_{\nu, 1-\frac{\alpha}{2}}^2} < \sigma^2 < \frac{vS^2}{\chi_{\nu, \frac{\alpha}{2}}^2}\right) = 1 - \alpha.$$

- This is the general formula for a two-tailed $100 \cdot (1 - \alpha)\%$ -confidence limit for σ^2 .
- The degrees of freedom $\nu = n - 1$, if s^2 is computed from a sample of size n .



Testing a Null Hypothesis Value of σ^2

- This situation occurs, for example, when a theoretical value of σ^2 is to be tested or when the sample data are being compared to a population whose σ^2 is known.
- If the hypotheses are

$$H_0 : \sigma^2 \leq (\geq) \sigma_0^2 \quad \text{vs} \quad H_1 : \sigma^2 > (<) \sigma_0^2,$$

then reject H_0 if

$$Y = \frac{vs^2}{\sigma_0^2} = \frac{\sum_{i=1}^n x_i^2}{\sigma_0^2} > \chi_{\nu, \alpha}^2 (< \chi_{\nu, 1-\alpha}^2).$$



Test for Equal Means $\mu_1 = \mu_2$ of Two Independent Samples

- One of many aspects of comparative studies is comparing the means of two different populations.
- There are two cases to distinguish:
 1. both populations have an identical standard deviation ($\sigma_1 = \sigma_2$);
 2. both populations have different standard deviations ($\sigma_1 \neq \sigma_2$).



Test for Equal Means $\mu_1 = \mu_2$ of Two Independent Samples

- Before testing the hypotheses, it is important to acquire information about the variance of the difference of the sample means $\sigma_{\bar{x}_1 - \bar{x}_2}^2$.
- Under the assumption of independent random variables, the variance of the difference between the sample means is defined as:

$$\sigma_{\bar{x}_1 - \bar{x}_2}^2 = \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}.$$



Test for Equal Means $\mu_1 = \mu_2$ of Two Independent Samples

- When both populations have a common variance

$\sigma^2 = \sigma_1^2 = \sigma_2^2$, then σ^2 is estimated by the unbiased pooled estimator of the $\bar{x}_1 - \bar{x}_2$ variance s_{pooled}^2 :

$$s_{\text{pooled}}^2 = \frac{\text{pooled sum of squares}}{\text{pooled degrees of freedom}} = \frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}.$$

Thus, the sample estimate $s_{\bar{x}_1 - \bar{x}_2}^2$ of the population variance $\sigma_{\bar{x}_1 - \bar{x}_2}^2$ is

$$s_{\bar{x}_1 - \bar{x}_2}^2 = s_{\text{pooled}}^2 \left(\frac{1}{n_1} + \frac{1}{n_2} \right).$$



Test for Equal Means $\mu_1 = \mu_2$ of Two Independent Samples

2. When both populations have different variances $\sigma_1^2 \neq \sigma_2^2$, then $\sigma_{\bar{x}_1 - \bar{x}_2}^2$ is estimated by the following unbiased estimator:

$$s_{\bar{x}_1 - \bar{x}_2}^2 = \frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}.$$



Testing the Hypothesis $\mu_1 = \mu_2$ when $\sigma^2 = \sigma_1^2 = \sigma_2^2$

- Under this assumption the rejection rule for H_0 in the **two-tailed test** is

$$H_0 : \mu_1 = \mu_2 \quad \text{vs} \quad H_1 : \mu_1 \neq \mu_2,$$

$$\text{with } |T| = \left| \frac{\bar{x}_1 - \bar{x}_2}{s_{\bar{x}_1 - \bar{x}_2}} \right| > t_{n_1+n_2-2, 1-\frac{\alpha}{2}}.$$

- For a **one-tailed test** the hypotheses are

$$H_0 : \mu_1 \leq (\geq) \mu_2 \quad \text{vs} \quad H_1 : \mu_1 > (<) \mu_2,$$

with rejection of H_0 if

$$T = \frac{\bar{x}_1 - \bar{x}_2}{s_{\bar{x}_1 - \bar{x}_2}} > (<) t_{n_1+n_2-2, 1-\alpha}.$$



Testing the Hypothesis $\mu_1 = \mu_2$ when $\sigma_1^2 \neq \sigma_2^2$

- Under this assumption the rejection rule for H_0 in the two-tailed test is

$$H_0 : \mu_1 = \mu_2 \quad \text{vs} \quad H_1 : \mu_1 \neq \mu_2,$$

with $|T| = \left| \frac{\bar{x}_1 - \bar{x}_2}{s_{\bar{x}_1 - \bar{x}_2}} \right| > t_{\nu, 1 - \frac{\alpha}{2}}$.

- The degrees of freedom ν are computed as

$$\nu = \left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2} \right)^2 \left(\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1} \right)^{-1}.$$



Testing the Hypothesis $\mu_1 = \mu_2$ when $\sigma_1^2 \neq \sigma_2^2$

- For a **one-tailed test** the hypotheses are

$$H_0 : \mu_1 \leq (\geq) \mu_2 \quad \text{vs} \quad H_1 : \mu_1 > (<) \mu_2,$$

with rejection of H_0 if

$$T = \frac{\bar{x}_1 - \bar{x}_2}{s_{\bar{x}_1 - \bar{x}_2}} > (<) t_{v, 1-\alpha}.$$

- Again the degrees of freedom v are computed as

$$\nu = \left(\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2} \right)^2 \left(\frac{(s_1^2/n_1)^2}{n_1 - 1} + \frac{(s_2^2/n_2)^2}{n_2 - 1} \right)^{-1}.$$



Univariate Statistical Analysis

Descriptive Statistics

Confidence Intervals and Hypotheses Testing

Linear Regression

Linear Regression

- The linear regression is a method used to study a linear relationship between two quantitative variables, an explained variable y and an explanatory variable X .
- Therefore, it is of interest to study "how y varies with changes in X ", or precisely, the causal (*ceteris paribus*) relationship between both variables.
- The model, which is assumed to hold in the population, is written as follows:

$$y = \beta_0 + \beta_1 X + \varepsilon.$$



Estimation

- Before estimating the parameters, it is important to check for 4 OLS assumptions:
 1. H_1 : the vectors \mathbf{e} and \mathbf{x} are independent.
 2. H_2 : $E(e_i|\mathbf{x}) = 0, \quad \forall i$.
 3. H_3 : $Var(e_i|\mathbf{x}) = \sigma^2, \quad \forall i$ (Homoscedasticity).
 4. H_4 : $E(e_i \cdot e_j|\mathbf{x}) = 0, \quad \forall i, j, i \neq j$.



Estimation

Definition

The estimator of the ordinary least square (OLS) method is defined as a vector $\hat{\beta}_{OLS} = (\hat{\beta}_0, \hat{\beta}_1)^\top$. It contains the coefficients of the linear combination of 1_n and \mathbf{X} . $\hat{\beta}_{OLS}$ minimises the distance between \mathbf{y} and $\mathbf{X}\beta$ using Euclidean norm:

$\hat{\beta}_{OLS} = \arg \min_{\beta} \|\mathbf{y} - \mathbf{X}\beta\|^2$. With assumption 1, the OLS estimator exists, is unique and has the representation

$$\hat{\beta}_{OLS} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$



Gauss Markov Theorem

- Since the OLS estimator is a function of the observations, its statistical properties depend on the joint distribution of (X, Y) . The relation

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}.$$

restricts the conditional distribution of Y given X , see Assumptions 2-4.

- Under assumptions 1-4, the OLS estimator is the best linear unbiased estimator — BLUE by the *Gauss Markov Theorem*

$$\hat{\boldsymbol{\beta}}_{OLS} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$



Variance of the OLS Estimator

- The OLS estimator is unbiased because $E(\hat{\beta}_{OLS}) = \beta = (\beta_0, \beta_1)^\top$ and among all other unbiased estimators $\hat{\beta}$, OLS estimator $\hat{\beta}_{OLS}$ has the smallest variance:

$$\text{Var} \left(\mathbf{c}^\top \hat{\beta}_{OLS} \right) \leq \text{Var} \left(\mathbf{c}^\top \hat{\beta} \right), \quad \forall c \in \mathbb{R}.$$

- This result holds whether or not we regard X as stochastic.
- The expression of the OLS estimator variance given the regressor \mathbf{X} is given by

$$\text{Var} \left(\hat{\beta}_{OLS} | \mathbf{X} \right) = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1}.$$

- The non-conditional estimator variance is given by

$$\text{Var} \left(\hat{\beta}_{OLS} \right) = \sigma^2 E \left\{ (\mathbf{X}^\top \mathbf{X})^{-1} \right\}.$$



Second Order Parameter Estimation

- The error term variance is called a parameter of second order because it corresponds to the second moment of the variable Y given the regressor X .
- It is used to measure the goodness of fit and to generate test statistics.
- Under assumptions 1-4, it can be shown that the residual variance is an unbiased estimator of σ^2 .
- The degrees of freedom are $n - 2$ since there are two parameters in simple linear regression:

$$\hat{\sigma}^2 = \frac{\hat{\varepsilon}^\top \hat{\varepsilon}}{n - 2} = \frac{\sum_{i=1}^n \hat{\epsilon}_i^2}{n - 2}.$$



OLS Estimator under Normality of the Error Term

- Let us assume that the error term follows a normal distribution:

$$H_n : \varepsilon | \mathbf{X} \sim N(\mathbf{0}, \sigma^2 \mathbf{I}_n)$$

- With this assumption H_n , the joint distribution (conditional on \mathbf{X}) can be determined.
- There are three important results:
 - $\hat{\beta}_{OLS} | \mathbf{X} \sim N\{\beta, \sigma^2(\mathbf{X}^\top \mathbf{X})^{-1}\}$,
 - $(n - 2)\hat{\sigma}^2 / \sigma^2 | \mathbf{X} \sim \chi^2(n - 2)$,
 - $\hat{\beta}_{OLS}$ and $\hat{\sigma}^2$ are independent (*Cochran's Theorem*).



Inferences about β (Part I)

- Under the hypothesis H_n one has the following result:

$$\frac{\hat{\beta}_k - \beta_k}{\hat{\sigma}_k^2} | X \sim t_{n-2}, \quad k = 1, 2.$$

- This result is crucial for hypothesis testing. If one considers the following test:

$$H_0 : \beta_k = \beta_k^0 \quad \text{vs} \quad H_1 : \beta_k \neq \beta_k^0.$$



Inferences about β (Part II)

- The t -statistic is given by

$$t = \frac{\hat{\beta}_k - \beta_k^0}{\hat{\sigma}_k}, \quad k = 1, 2.$$

with the rejection region

$$\begin{aligned} W &= \left\{ T \mid |T| > t_{n-2}(1 - \alpha/2) \right\} \\ &= (-\infty, -t_{n-1, 1-\frac{\alpha}{2}}) \cup (t_{n-1, 1-\frac{\alpha}{2}}, +\infty). \end{aligned}$$



Goodness of Fit

- ◻ The analysis of variance is based on the fact that the vector of residuals and the vector of fitted values are orthogonal.
- ◻ In a simple linear regression, the vector of fitted values is a linear combination of 1_n and \mathbf{X} .
- ◻ Therefore it can be shown that the sum of residuals is zero and $\bar{y} = \hat{y}$.
- ◻ Furthermore, we write

$$\mathbf{y} = \hat{\mathbf{y}} + \hat{\boldsymbol{\varepsilon}}$$

$$\mathbf{y} - \bar{\mathbf{y}} = \hat{\mathbf{y}} - \bar{\hat{\mathbf{y}}} + \hat{\boldsymbol{\varepsilon}}.$$



Goodness of Fit

- Using the orthogonality, it can be shown that the total sum of squares (SST) is the sum of the explained sum of squares (SSE) and the residual sum of squares (RSS).

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2 + \sum_{i=1}^n \varepsilon_i^2$$

$$SST = SSE + RSS$$

$$Var(y) = Var(\hat{y}) + Var(\varepsilon)$$

- Thus, the total variance is the sum of the explained variance and the residual variance. Normalizing the last expression, one obtains

$$1 = \frac{SSE}{SST} + \frac{RSS}{SST}.$$



Goodness of Fit

Definition

The measure of the goodness of fit R^2 is defined as

$$R^2 = \frac{SSE}{SST} = 1 - \frac{SSR}{SST} = \frac{\|\hat{\mathbf{y}} - \bar{\mathbf{y}}\mathbf{1}_n\|^2}{\|\mathbf{y} - \bar{\mathbf{y}}\mathbf{1}_n\|^2} = 1 - \frac{\|\hat{\boldsymbol{\varepsilon}}\|^2}{\|\mathbf{y} - \bar{\mathbf{y}}\mathbf{1}_n\|^2} \in [0, 1].$$



Multiple Linear Restriction

- A F -statistic is used that compares the difference between $SS(\text{reduced})$ and $SS(\text{full})$:

$$F = \frac{\frac{SS(\text{reduced}) - SS(\text{full})}{df(r) - df(f)}}{\frac{SS(\text{full})}{df(f)}}.$$

- Under the null hypothesis this statistic follows the distribution $F_{df(r)-df(f), df(r)}$, where $df(f)$ and $df(r)$ denote the degrees of freedom under the unrestricted model and the restricted model ($df(f) = n - d - 1$ and $df(r) = n - 1$).



Example: Multiple Linear Restriction

Example 14

Consider in this example `LifeCycleSavings`, a data frame with 50 observations on 5 variables: `sr` (aggregate personal savings), `pop15` (percentage of population under 15), `pop75` (percentage of population over 75), `dpi` (real per-capita disposable income) and `ddpi` (percentage growth rate of `dpi`). Suppose it is desired to examine whether real per-capita disposable income has a causal effect on aggregate personal savings. Thus, do a simple linear regression of the variable `sr` on `dpi`.

The model is written as

$$y_i = \beta_1 \cdot x_i + \varepsilon_i \quad (\text{model without intercept}),$$
$$y_i = \beta_0 + \beta_1 \cdot x_i + \varepsilon_i \quad (\text{model with intercept}).$$



Example: Multiple Linear Restriction

```
1 > # model without intercept
2 > lm(LifeCycleSavings$sr ~ LifeCycleSavings$dpi - 1)
3 Call:
4 lm(formula = LifeCycleSavings$sr ~ LifeCycleSavings$dpi -
   1)
5
6 Coefficients:
7 LifeCycleSavings$dpi
8 0.005332
```



Example: Multiple Linear Restriction

```
1 > # model with intercept
2 > lm(LifeCycleSavings$sr ~ LifeCycleSavings$dpi)
3 Call:
4 lm(formula = LifeCycleSavings$sr ~ LifeCycleSavings$dpi)
5
6 Coefficients:
7 (Intercept) LifeCycleSavings$dpi
8 8.5682306      0.0009964
```



Example: Multiple Linear Restriction

```
1 > summary(lm(LifeCycleSavings$sr ~ LifeCycleSavings$dpi))
2
3 Call:
4 lm(formula = LifeCycleSavings$sr ~ LifeCycleSavings$dpi)
5
6 Residuals:
7 Min      1Q  Median      3Q      Max
8 -9.1915 -3.6215  0.4418  2.8304 11.2790
```



Example: Multiple Linear Restriction

```
1 Coefficients:
2 Estimate Std. Error t value Pr(>|t|)
3 (Intercept)          8.5682306   0.9414690    9.101 5.04e-12
4 *** 
5 LifeCycleSavings$dpi 0.0009964   0.0006366    1.565    0.124
6 --- 
7 Residual standard error: 4.416 on 48 degrees of freedom
8 Multiple R-squared: 0.04856,      Adjusted R-squared:
9          0.02874
F-statistic: 2.45 on 1 and 48 DF,  p-value: 0.1241
```



Generalized Least Square Method

- The OLS method assumes that the error terms are i.i.d. with the variance equal $\sigma^2 \mathbf{I}_n$ (assumptions 3 and 4).
- This is an assumption that does not necessarily hold in all practical situations.
- Less strong assumptions are made by the GLS method.
- It assumes that the variances are not equal for each error term and that they can be correlated to each other.



Generalized Least Square Method

- The idea of GLS method is to weight the linear model with the variance-covariance matrix of the error term Ω .
- Thus, the model is written as

$$\Omega^{-1} \mathbf{Y} = \Omega^{-1} \mathbf{X} \boldsymbol{\beta} + \Omega^{-1} \boldsymbol{\varepsilon},$$

with the assumptions

1. $E(\boldsymbol{\varepsilon}|\mathbf{X}) = \mathbf{0}$;
2. $\text{Var}(\boldsymbol{\varepsilon}|\mathbf{X}) = \Omega$ invertible;
3. $\mathbf{X}^\top \mathbf{X}$ invertible.

- The GLS estimator $\hat{\boldsymbol{\beta}}_{GLS}$ is the OLS estimator of the weighed linear model, which is given by

$$\hat{\boldsymbol{\beta}}_{GLS} = (\mathbf{X}^\top \Omega^{-1} \mathbf{X})^{-1} \mathbf{X}^\top \Omega^{-1} \mathbf{y}.$$



The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

Basic Nonparametric Methods

Motivation

Nonparametric Tests

Nonparametric Density Estimation

Nonparametric Regression

- In standard approaches of statistics a parametric density is fitted to a sample
- Many datasets are not well approximated by parametric probability distributions
 - ▶ Inference about stationary samples with unknown distribution?



Basic Nonparametric Methods

Motivation

Nonparametric Tests

Nonparametric Density Estimation

Nonparametric Regression

Nonparametric Tests

Test	Samples	R Syntax	Null Hypothesis
Kolmogorov-Smirnov	≤ 2	<code>ks.test()</code>	$F = G$
Anderson-Darling	≤ 2	<code>ad.test()</code>	$F = G$
Cramér-von-Mises	≤ 2	<code>ad.test()</code>	$F = G$
Shapiro-Wilk	≤ 2	<code>shapiro.test()</code>	$X \sim N$
Wilcoxon signed rank	≤ 2	<code>wilcox.test()</code>	$\tilde{x}_{0.5} = c$
Mann-Whitney U	≤ 2	<code>wilcox.test()</code>	$F_1(x) = F_2(x)$
Kruskal-Wallis	any	<code>Kruskal.test()</code>	$F_1(x) = \dots = F_m(x)$

Table 4: Conducting nonparametric tests in R.



The Kolmogorov-Smirnov Test

- Let G and F be two cdf and x be on their support

$$\text{KS} = \sup_x |F(x) - G(x)| \quad (35)$$

- $H_0 : F = G$ $H_1 : F \neq G$



The Kolmogorov-Smirnov Test One Sample

```
1 > require(stats)
2 > dax      = EuStockMarkets[,1]    # DAX index
3 > r.dax    = diff(log(dax))        # log-returns
4 > r.dax_m = mean(r.dax); r.dax_sd = sd(r.dax)
5 > r.dax_st = (r.dax - r.dax_m)/r.dax_sd
6 > l = function(k, x){-sum(dt(x, df = k, log = T))}
7 > k_ML = optimize(f = l, interval = c(0, 30), x = r.dax_st
   )$minimum
8 > # test for t-dist.
9 > ks.test(x = r.dax_st, y = "pt", df = k_ML)
10    One-sample Kolmogorov-Smirnov test
11 data: r.dax_st
12 D = 0.063173, p-value = 7.194e-07
13 alternative hypothesis: two-sided
```



The Kolmogorov-Smirnov Test Two Samples

```
1 > ftse    = EuStockMarkets[,4]  # FTSE index
2 > r.ftse = diff(log(ftse)); r.ftse_m  = mean(r.ftse)
3 > r.ftse_sd = sd(r.ftse)
4 > r.ftse_st = (r.ftse - r.ftse_m)/r.ftse_sd
5 > ks.test(r.dax_st, r.ftse_st) # test with
6 >                                # standardised log-returns
7 Two-sample Kolmogorov-Smirnov test
8
9 data: r.dax_st and r.ftse_st
10 D = 0.034965, p-value = 0.2058
11 alternative hypothesis: two-sided
```



The Cramer-von-Mises Test

- Like KS but with squared loss function instead of supremum

$$\int_{-\infty}^{\infty} \{F(x) - G(x)\}^2 dG(x)$$

- L_2 -norm (square) or L_∞ -norm (supremum)
- Alternative: Anderson-Darling test

Richard E. v. Mises and Harald Cramer on BBI:



The Cramer-von-Mises Test

```
1 > require(goftest)
2 > cvm.test(r.dax_st, null = "pt", df = k_ML) # Cramer von
   Mises test
3
4     Cramer-von Mises test of goodness-of-fit
5     Null hypothesis: Student's t distribution
6     with parameter df = 11.5608814093413
7
8 data: r.dax_st
9 omega2 = 3.0274, p-value = 6.458e-08
```



Anderson Darling Test

- The Anderson-Darling test sets $w(x) = [G(x)\{1 - G(x)\}]^{-1}$, which leads to the test statistic

$$A^2 = -n - \frac{1}{n} \sum_{i=1}^n (2i-1) [\log\{G(z_{(i)})\} + \log\{1 - G(z_{(n+1-i)})\}] .$$



Andersson Darling Test

```
1 > require(goftest)
2 > # Andersson Darling test
3 > ad.test(r.dax_st, null = "pt", df = k_ML)
4
5   Anderson-Darling test of goodness-of-fit
6   Null hypothesis: Student's t distribution
7   with parameter df = 11.5608814093413
8
9   data: r.dax_st
10  An = 17.715, p-value = 3.228e-07
```



The Shapiro-Wilk Test

- ◻ X be a random variable with unknown probability density
- ◻ Let $\hat{\sigma}$ be the sample standard deviation

$$W = \frac{\sigma^2}{(n-1)\hat{\sigma}^2} \quad (36)$$

$$\sigma = \sqrt{\sum_{i=1}^n \frac{\mu^\top \Sigma^{-1} x_i}{\sqrt{\mu^\top \Sigma^{-1} \Sigma^{-1} \mu}}} \quad (37)$$

- ▶ σ is the theoretically expected variance if $X \sim N$



The Shapiro-Wilk Test

- The critical values of W are tabulated
- Alternatively they are obtained via Monte Carlo Simulation
- $H_0 : X \sim N$ $H_1 : X \not\sim N$



The Shapiro-Wilk Test

```
1 > shapiro.test(r.dax) # by default H_0: X ~ N(mu, sigma^2)
2
3   Shapiro-Wilk normality test
4
5 data: r.dax
6 W = 0.9538, p-value < 2.2e-16
```



Jarque-Bera Test

- An alternative test for normality is the Jarque-Bera test.
- The hypotheses are the same as for the Shapiro-Wilk test.
- The Jarque-Bera test considers the third and fourth moment of the distribution. Let S denote the sample skewness and K the sample kurtosis. Then

$$JB = \frac{n}{6} \left\{ S^2 + \frac{(K - 3)^2}{4} \right\}.$$



Wilcoxon Signed Rank Test

- The Wilcoxon signed rank test is an asymptotic test for the median $\tilde{x}_{0.5}$ of sample $\{x_1, \dots, x_n\}$.

$$H_0 : \tilde{x}_{0.5} = c \quad \text{vs.} \quad H_1 : \tilde{x}_{0.5} \neq c,$$

Frank Wilcoxon on BBI:



Wilcoxon Signed Rank Test

1. Randomly draw $n_s = \min(n_1, n_2)$ observations from the larger sample.
2. Calculate $s_i = \text{sign}(x_{1,i} - x_{2,i})$ and $d_i = |x_{1,i} - x_{2,i}|$ for the paired samples.
3. Compute the ranks R_i of d_i ascending from 1.
4. Then the test statistic is $W = |\sum_{i=1}^{n_s} s_i R_i|$.



Wilcoxon Signed Rank Test

```
1 > c = 50                      # maximum median under H_0
2 > y = rep(c, length(presidents))# vector of constants
3 > wilcox.test(presidents, y,    # test for presidents
4 +   alternative = c("greater"),# specifies H1
5 +   paired = T)              # signed rank test
6
7 Wilcoxon signed rank test with continuity correction
8
9 data: presidents and y
10 V = 4613.5, p-value = 3.298e-05
11 alternative hypothesis: true location shift is greater
than 0
```



The Kruskal-Wallis Test

- Test equality of m samples
- $H_0 : F_1(x) = \dots = F_m(x)$
 $H_1 : F_1(x) = \dots F_l(x + c) = \dots = F_m(x) \quad \forall x \in \mathbb{R}$
- Rank all observations by size, $\text{rank}_{i,j} = R_{i,j}$
- Let \bar{R}_l denote the average rank in sample l

$$K = (N - 1) \frac{\sum_{j=1}^m N_j (\bar{R}_l - \bar{R})}{\sum_{j=1}^m \sum_{l=1}^{N_i} (R_{j,l} - \bar{R})^2} \quad (38)$$



The Kruskal-Wallis Test

```
1 > decades = c(rep(1, length = 20), # group indicator for
   +           decades
2 +           rep(2:3, each = 40),
3 +           rep(4, length = 20))
4 > kruskal.test(presidents, decades) # Kruskal-Wallis test
5
6 Kruskal-Wallis rank sum test
7
8 data: presidents and decades
9 Kruskal-Wallis chi-squared = 12.2607, df = 3, p-value =
 0.006541
```



Basic Nonparametric Methods

Motivation

Nonparametric Tests

Nonparametric Density Estimation

Nonparametric Regression

Nonparametric Density Estimation

- A simple density estimator: The histogram
- Challenge: How fine should the histogram be?
 - ▶ More bins: More adaptive (better fit but less robust)
 - ▶ Less resolution: More smooth (robust but biased)



The Histogram and Binwidth

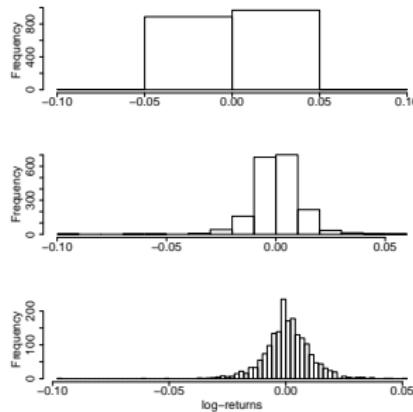


Figure 31: Histogram of DAX log-returns with different bin sizes.

BCS_HistBinSizes



The Integrated Mean Squared Error (MISE)

- For an estimator $\hat{\theta}$ the mean square error (MSE) is

$$E[(\hat{\theta} - \theta)^2] = \text{Var}(\hat{\theta}) + \{\text{BIAS}(\hat{\theta}, \theta)\}^2 \quad (39)$$

- The Integrated mean square error (MISE) is

$$\text{MISE} = \int \text{MSE} d\theta$$



The Mean Squared Error (MSE) and the Binwidth

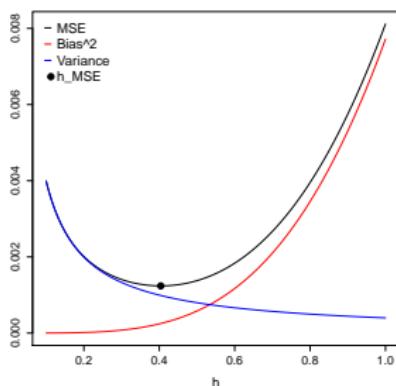


Figure 32: $\text{BIAS}\{\hat{f}_h(x)\}^2$ (red), $\text{Var}\{\hat{f}_h(x)\}$ (blue) and $MSE(x)$ (black) for a histogram with origin $x_0 = 0$ evaluated at $x = 0.001$ and $X \sim N(0, 1)$.

The Asymptotic MISE

- Let the bandwidth of the histogram be denoted by h
- $AMISE = \lim_{h \rightarrow 0} (MISE)$
- The optimal bandwidth with respect to the AMISE
 - ▶ $\frac{\partial AMISE}{\partial h} = 0$
 - ▶ $h_{opt}^{AMISE} = \left(\frac{6}{n ||f||_2^2} \right)^{1/5}$
- $||f||_2^2$ parametrically or via cross-validation



Kernel Density Estimation

- The histogram approximated the pdf as

$$\hat{p} \approx \frac{F(x+h) - F(x-h)}{2h}.$$

- Or equivalently:

$$\hat{p} = \frac{1}{2nh} \sum_{i=1}^n \mathbf{I}\{x+h \geq X_i > x-h\}.$$



The Kernel

- Allow for arbitrary weighting of $\mathbf{I}\{x + h \geq X_i > x - h\}$ with w
- The Kernel is defined as

$$K_h(x - x_i) = \frac{1}{2} \mathbf{I}\{x + h \geq X_i > x - h\} w(x_i)$$

- The pdf is estimated via

$$\hat{f}_h(x) = \frac{1}{n} \sum_{i=1}^n K_h(x - x_i).$$



Popular Kernels

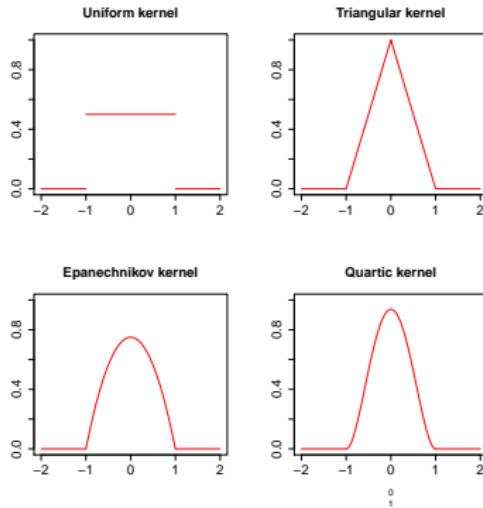


Figure 33: Popular kernel functions.

BCS_PopularKernels



Density Estimation of DAX log-returns

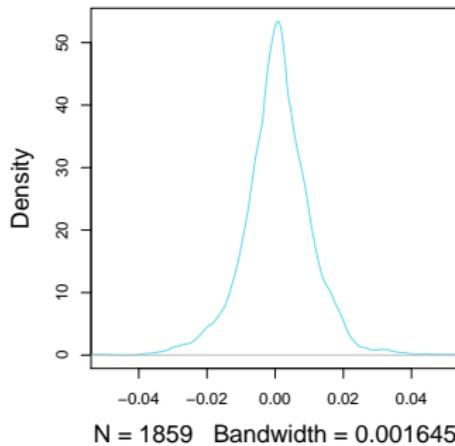


Figure 34: Nonparametric density estimation of the DAX log-returns.

Basic Nonparametric Methods

Motivation

Nonparametric Tests

Nonparametric Density Estimation

Nonparametric Regression

Kernel Regression

k Nearest Neighbors

Splines

LOESS

Application

Why Nonparametric Regression ?

For n observations

of $\{(X_i, Y_i)\}_{i=1}^n$ we pose:

$$y_i = g(x_i) + \varepsilon_i \quad , \quad i = 1, \dots, n \quad (40)$$

- X is our explanatory variable
- Y is the explained variable
- ε is the common modelling for the noise



Why Nonparametric Regression ?

Parametric regression: $g(X_i) = g(X_i, \theta)$

- estimate θ
- use $\hat{g}(X_i) = g(X_i, \hat{\theta})$



Analogy with the Histogram

Like the histogram, we can create bins of size h .

$$\hat{g}(x) = \begin{cases} \frac{\sum_{i=1}^n 1\{|x - x_i| < h/2\}y_i}{\sum_{i=1}^n 1\{|x - x_i| < h/2\}}, & \text{if } \exists i, |x - x_i| < h/2 \\ 0 & \text{else.} \end{cases}$$



General form with Kernel

$$\hat{g}(x) = \sum_{i=1}^n \frac{K(\frac{x-x_i}{h})y_i}{\sum_{i=1}^n K(\frac{x-x_i}{h})} \quad (41)$$

A Kernel follows:

- $\int_{-\infty}^{+\infty} K(u)du = 1$
- $K(u) = K(-u)$

If K is a Kernel, then $K^*(u) = \lambda K(\lambda u)$ is also a Kernel.

Many popular Kernels:

- Uniform Kernel: $K(x) = I(|x| \leq 1)/2$
- Gaussian Kernel: $K(x) = e^{-x^2}/\sqrt{2\pi}$
- Epanechnikov Kernel: $K(x) = 3(1 - u^2)I(|x| \leq 1)/4$



Financial Example

Data from EuStockMarkets, from 1991 to 1998.

$$\ln\left(\frac{DAX_{t+1}}{DAX_t}\right) = f\left(\ln\left(\frac{FTSE_{t+1}}{FTSE_t}\right)\right) + \varepsilon$$



Uniform Kernel

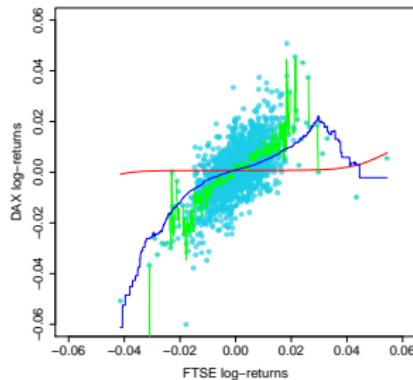


Figure 35: Nonparametric regression of daily DAX log-returns by daily FTSE log-returns, for $h = 0.1$ (red), $h = n^{-1}$ (green) and $h = n^{-1/2}$ (blue).

Gaussian Kernel

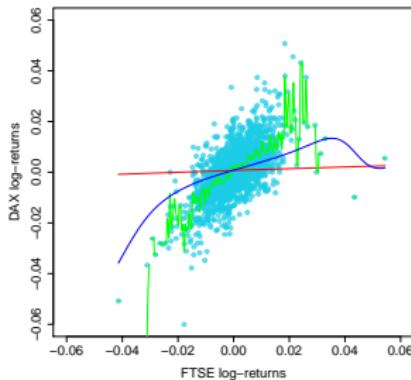


Figure 36: Nonparametric regression of daily DAX log-returns by daily FTSE log-returns, for $h = 0.1$ (red), $h = n^{-1}$ (green) and $h = n^{-1/2}$ (blue).

k Nearest Neighbors

$B_{h,i} = \{x, |x - x_i| < h/2\}$ for Kernel using $\mathbf{I}(|x| \leq h/2)$

k NN regression uses fixed-size (k) Bins: $B_{x,k}$

$\forall x_i \in B_{x,k}, \forall x^* \notin B_{x,k}$

$$|x - x_i| \leq |x^* - x_i| \tag{42}$$



k Nearest Neighbors

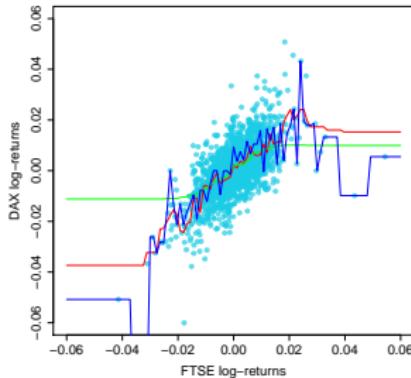


Figure 37: Nonparametric regression of daily DAX log-returns by daily FTSE log-returns, for $k = 1$ (red), $k = 10$ (green) and $k = 250$ (blue).

 BCS_kNN



Imposing some restriction on g

$$\min_{g, g \in \mathcal{C}^2} \sum_{i=1}^n \{Y_i - g(X_i)\}^2 + \lambda \int \left(\frac{d^2g}{dx^2}\right)^2 dx \quad (43)$$

Smoothing is controlled by λ , similar to the bandwidth h and the number of neighbours k .



Spline

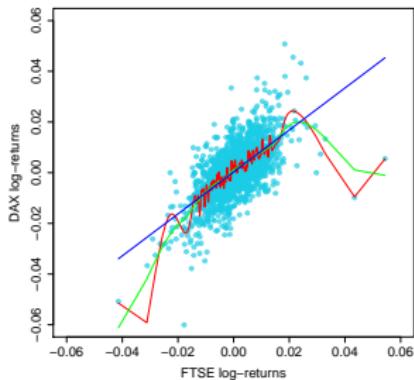


Figure 38: Nonparametric regression of daily DAX log-returns by daily FTSE log-returns, for $\lambda = 2$ (red), $\lambda = 1$ (blue) and $\lambda = 0.2$ (green).

BCS_Splines



Improvement of the kNN

Similar to k NN, with the following differences:

- adds a weighting to the sum of y_i ;
- uses local polynomials for the fitting

The usual weighting is made with the Tri-Cube function:

$$w(x) = (1 - |x|^3)^3 \times I(x > 0)$$



LOESS

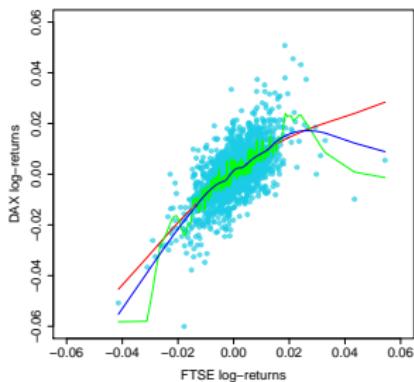


Figure 39: Nonparametric regression of daily DAX log-returns by daily FTSE log-returns, for $\alpha = 0.9$ (red), $\alpha = 0.3$ (blue) and $\alpha = 0.05$ (green)

BCS _ LOESS



How do we choose these 'smoothing parameters'?

Not many methods exist:

- rule-of-thumb
- cross-validation
- plug-in



Example of Nonlinear Relationship

$$Y = f(X) + \varepsilon$$

- $f(x) = \sin(x) - x^2$
- $X \sim N(0, 1)$
- $\varepsilon \sim N(0, 1)$
- $n = 50$



Example of Nonlinear Relationship

- rule-of-thumb: Gaussian Kernel
- cross-validation: Spline, LOESS, k NN

R has built-in packages for Kernel, Spline and LOESS but not for k NN regression (only k NN classification).



Example of Nonlinear Relationship

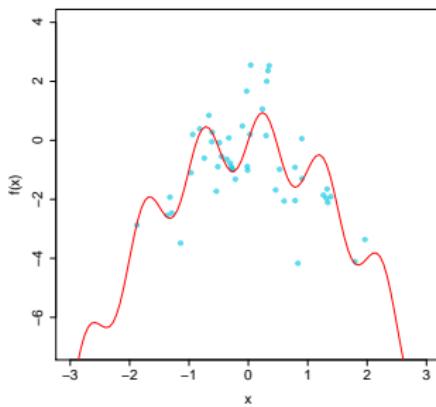


Figure 40: Simulations along the regression curve $f(x) = \sin(2\pi x) - x^2$ displayed by points. The true regression curve $y = \sin(2\pi x) - x^2$ is depicted in (red).

Cross-validation for k NN

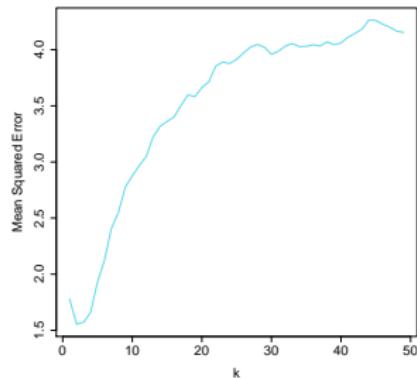


Figure 41: MSE for k -NN regression using the Leave-one-out cross validation method.

BCS_LeaveOneOut



Results

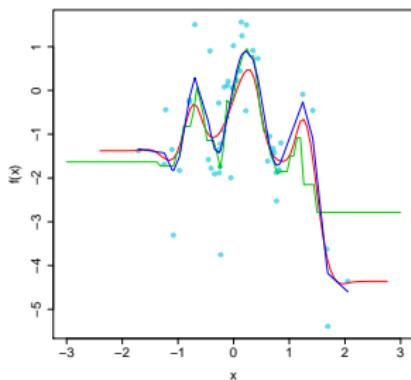


Figure 42: Nonparametric regressions for simulated data. The regression results for kernel (red), kNN (green) and spline (blue) are plotted.

 BCS_NonparametricRegressions



The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

Multivariate Distributions

Motivation

Distribution and density function

Multivariate normal distribution

Copula

Copula estimation

Multivariate distribution

This chapter gives an introduction to the basic probability and statistical tools for multivariate analysis, where the relationship between d random variables is considered.

- Distribution and density function
- Multivariate normal distribution
- Copula
- Copula estimation



Multivariate Distributions

Motivation

Distribution and density function

Multivariate normal distribution

Copula

Copula estimation

Multivariate distribution

- Here we set a random vector $X = (X_1, X_2, \dots, X_d)^\top$, then the cumulative distribution function (CDF) is

$$F(X) = P(X \leq x) = P(X_1 \leq x_1, X_2 \leq x_2, \dots, X_d \leq x_d).$$

- If X is discrete, then the joint probability distribution function $p(\cdot)$ is

$$p(x) = P(X = x) = P(X_1 = x_1, X_2 = x_2, \dots, X_d = x_d).$$



Density function

- Assuming X_1, X_2, \dots, X_d to be continuous, then the joint probability density function is

$$f(x) = \frac{\partial^d F(x)}{\partial x} = \frac{\partial^d F(x_1, \dots, x_d)}{\partial x_1 \dots \partial x_d},$$

$$\int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} f(u) d u_1 \dots d u_d = 1.$$

- If the joint probability density function is differentiable, then the following relationship holds,

$$F(x) = \int_{-\infty}^{x_1} \dots \int_{-\infty}^{x_d} f(u_1, \dots, u_d) d u_1 \dots d u_d.$$



Marginal function

Partition $(X_1, X_2, \dots, X_d)^\top$ as $X_k = (X_{i_1}, \dots, X_{i_k})^\top \in \mathbb{R}^k$ and $X_{-k} = (X_{i_{k+1}}, \dots, X_{i_d})^\top \in \mathbb{R}^{d-k}$, where i_1, \dots, i_d is a permutation of $1, \dots, d$.

- k-dimensional marginal cdf,

$$F_{X_k}(x_1) = P(X_{i_1} \leq x_1, \dots, X_{i_k} \leq x_k) = F(x_1, \dots, x_k, \infty, \dots, \infty).$$

- For continuous variables, the marginal pdf,

$$f_1(x_{i_1}, \dots, x_{i_k}) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} f(x_1, \dots, x_d) \, dx_{i_k+1} \cdots dx_{i_d}.$$

- For discrete X , the marginal probability is calculated by

$$p_1(x_{i_1}, \dots, x_{i_k}) = \sum_{x_{i_{k+1}}, \dots, x_{i_d}} p(x_1, \dots, x_d).$$



Conditional function

For simplicity, we restrict our example to the 2-dimensional case,
 $X = (X_1, X_2)^\top$.

- The conditional probability density function of X_2 is given

$$X_1 = x_1,$$

$$f(x_2|x_1) = \frac{f(x_1, x_2)}{f_{X_1}(x_1)}.$$

- X_1, X_2 are independent iff

$$f(x_1, x_2) = f_{X_1}(x_1)f_{X_2}(x_2).$$



Expectation

- The first order moment of X is often called expectation,
 $E(X) \in \mathbb{R}^d$,

$$E(X) = \begin{pmatrix} E(X_1) \\ \vdots \\ E(X_d) \end{pmatrix} = \int xf(x) d(x) = \begin{pmatrix} \int x_1 f(x) d(x) \\ \vdots \\ \int x_d f(x) d(x) \end{pmatrix} = \mu.$$

- The operation of expectations is linear,

$$E(\alpha X + \beta Y) = \alpha E X + \beta E Y.$$



Expectation

- ◻ If \mathcal{A} is a $(q \times d)$ matrix of real numbers, then we have,

$$\mathbb{E}(\mathcal{A}X) = \mathcal{A}\mathbb{E}X.$$

- ◻ If X and Y are independent then,

$$\begin{aligned}\mathbb{E}(XY^\top) &= \int xy^\top f(x, y) dx dy \\ &= \int xf(x) dx \int y^\top f(y) dy \\ &= \mathbb{E}X\mathbb{E}Y^\top.\end{aligned}$$



Estimator of expectation

- ◻ \bar{x} is an unbiased estimator of the expectation of $E X$, which is obtained by,

$$\bar{x} = \begin{pmatrix} \bar{x}_1 \\ \vdots \\ \bar{x}_d \end{pmatrix} = \frac{1}{n} \mathcal{X}^\top \mathbf{1}_n.$$

- ◻ R offers several functions to calculate the sample mean. The function `mean` is applicable to `vector`, `matrix` and `data.frame` types.



Example of mean computation

```
1 > mean(women)           # mean of data.frame  
2   height    weight  
3   65.0000 136.7333
```

```
1 > women.m=as.matrix(women) # convert to matrix  
2 > mean(women.m)           # mean of matrix  
3 [1] 100.8667
```

```
1 > rowMeans(women.m)       # averages by rows  
2 [1] 86.5 88.0 90.0 92.0 94.0 96.0 98.0  
3 [2] 100.0 102.5 104.5 107.0 109.5 112.0 115.0 118.0  
4 > colMeans(women.m)       # averages by columns  
5   height    weight  
6   65.0000 136.73333
```



Covariance matrix

- Definition of variance matrix Σ

$$\text{Var}(X) = \Sigma = E(X - \mu)(X - \mu)^T = E(XX^T) - \mu\mu^T,$$

where the random vector X has a distribution with the vector of expected values μ and covariance matrix Σ .

$$X \sim (\mu, \Sigma), \Sigma \geq 0, \text{ with elements } \Sigma = (\sigma_{X_i X_j}).$$

$$\text{Cov}(X_i, X_j) = \sigma_{X_i X_j} = E(X_i X_j) - \mu_i \mu_j.$$

$$\text{Var}(X_i) = \sigma_{X_i X_i} = E X_i^2 - \mu_i^2.$$



Properties of variance and covariance

$$\text{Var}(\mathcal{A}^\top X) = \mathcal{A}^\top \text{Var}(X) \mathcal{A} = \sum_{i,j} a_i a_j \sigma_{X_i X_j}, \quad i, j = 1, \dots, d,$$

$$\text{Var}(\mathcal{A}X + b) = \mathcal{A} \text{Var}(X) \mathcal{A}^\top,$$

$$\Sigma_{XY} = \text{Cov}(X, Y) = E(X - \mu_X)(Y - \mu_Y)^\top,$$

$$\text{Cov}(X + Y, Z) = \text{Cov}(X, Z) + \text{Cov}(Y, Z),$$

$$\text{Var}(X + Y) = \text{Var}(X) + \text{Cov}(X, Y) + \text{Cov}(Y, X) + \text{Var}(Y),$$

$$\text{Cov}(\mathcal{A}X, \mathcal{B}Y) = \mathcal{A} \text{Cov}(X, Y) \mathcal{B}^\top.$$



Sample covariance matrix

- The unbiased estimator of the second moment $\widehat{\Sigma}$

$$\widehat{\Sigma} = \frac{1}{n-1} \mathcal{X}^\top \mathcal{X} - \frac{n}{n-1} \bar{x} \bar{x}^\top. \quad (7.4)$$

Equation (7.4) can be written equivalently in scalar form or on based of the centering matrix $\mathcal{H} = \mathcal{I}_n - n^{-1} \mathbf{1}_n \mathbf{1}_n^\top$:

$$\begin{aligned}\widehat{\Sigma} &= (n-1)^{-1} (\mathcal{X}^\top \mathcal{X} - n^{-1} \mathcal{X}^\top \mathbf{1}_n \mathbf{1}_n^\top \mathcal{X}), \\ &= (n-1)^{-1} \sum_{i=1}^n (x_i - \bar{x})(x_i - \bar{x})^\top, \\ &= (n-1)^{-1} \mathcal{X}^\top \mathcal{H} \mathcal{X}. \quad (7.5)\end{aligned}$$



R code for calculation of $\hat{\Sigma}$

```
1 > n      = dim(women.m)[1]
2 > cov1  = (t(women.m) %*% women.m - n*mean.w %*%
3           t(mean.w)) / (n-1)          # using (7.4)
4 > H      = diag(1,n) - 1/n*rep(1,n) %*% t(rep(1,n))
5 > cov2  = t(women.m) %*% H %*% women.m / (n-1)
6                                         # using (7.5)
7 > cov3  = cov(women)                 # for data.frame
8 > cov4  = cov(women.m)               # for matrix
```

All matrices cov1, cov2, cov3 and cov4 return the same result.

	height	weight
height	20	69.0000
weight	69	240.2095



Correlation

- The correlation ρ_{X_i, X_j} between two random variables X_i and X_j ,

$$\rho_{X_i, X_j} = \frac{\text{Cov}(X_i, X_j)}{\sqrt{\text{Var } X_i \text{ Var } X_j}} = \frac{\sigma_{X_i, X_j}}{\sqrt{\sigma_{X_i} \sigma_{X_j}}}.$$

- Correlation matrix for random variables X and Y ,

$$\text{Cor}(X, Y) = \text{Var } X^{-1/2} \text{ Cov}(X, Y) \text{ Var } Y^{-1/2}.$$



R code for sample correlation

The unbiased sample correlation is given by:

$$\hat{\rho}_{X_i, X_j} = \frac{n \sum_{m=1}^n x_{im} x_{jm} - \sum_{m=1}^n x_{im} \sum_{m=1}^n x_{jm}}{\sqrt{n \sum_{m=1}^n x_{im}^2 - (\sum_{m=1}^n x_{im})^2} \sqrt{n \sum_{m=1}^n x_{jm}^2 - (\sum_{m=1}^n x_{jm})^2}}.$$

In R, the sample correlation is calculated by the automic function `cor` and the covariance matrix can be done by using `cov2cor` function.

```
1 > cor(women)
2 > cov2cor(cov(women))
3
4      height      weight
5 height  1.0000000  0.9954948
6 weight  0.9954948  1.0000000
```



Kendall's τ and Spearman's ρ_S

Let $(X_1, X_2), (X'_1, X'_2)$ be independent random pairs with distribution F ,

- Kendall's τ

$$\tau = P \left\{ (X_1 - X'_1)(X_2 - X'_2) > 0 \right\} - P \left\{ (X_1 - X'_1)(X_2 - X'_2) < 0 \right\}.$$

- Spearman's ρ_S

$$\rho_S = \frac{\text{Cov} \{ F_1(X_1), F_2(X_2) \}}{\sqrt{\text{Var} \{ F_1(X_1) \}, \text{Var} \{ F_2(X_2) \}}}.$$



Empirical $\hat{\tau}$ and $\hat{\rho}_S$

- The empirical $\hat{\tau}$ and $\hat{\rho}_S$ can be given as follows,

$$\hat{\tau} = \frac{4}{n(n-1)} P_n - 1,$$
$$\hat{\rho}_S = \frac{\sum_{i=1}^n (R_i - \bar{R})^2 (S_i - \bar{S})^2}{\sqrt{\sum_{i=1}^n (R_i - \bar{R})^2 \sum_{i=1}^n (S_i - \bar{S})^2}}.$$

where P_n is the number of concordant pairs, i.e. the amount of pairs (x_{1k}, x_{2k}) and (x_{1m}, x_{2m}) of points in the sample for which holds:

$$\{x_{1k} < x_{1m} \text{ and } x_{2k} < x_{2m}\} \text{ or } \{x_{1k} > x_{1m} \text{ and } x_{2k} > x_{2m}\}.$$



Pearson's ρ , Spearman's ρ_s and Kendall's τ

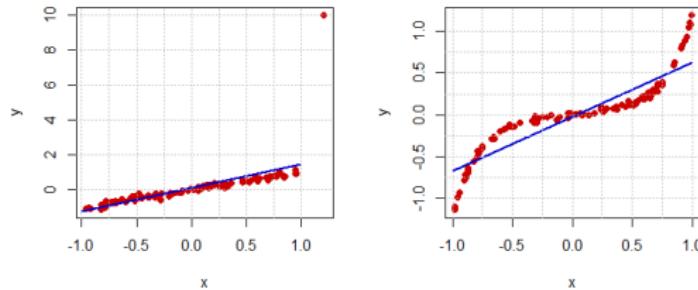


Figure 43: Linear fit for the linearly correlated data with an outlier (left) and most perfectly dependent monotone transformed data (right) (on the left panel $\rho = 0.66, \rho_s = 0.98, \tau = 0.88$ and on the right panel $\rho = 0.892, \rho_s = 0.996, \tau = 0.956$).



Multivariate Distributions

Motivation

Distribution and density function

Multivariate normal distribution

Copula

Copula estimation

Multivariate normal distribution

- ◻ X is normally distributed with μ and covariance $\Sigma > 0$, then the joint probability density function is

$$f(x) = |2\pi\Sigma|^{-1/2} \exp \left\{ -\frac{1}{2}(x - \mu)^\top \Sigma^{-1}(x - \mu) \right\}.$$

- ◻ The multivariate normal distribution does not have an explicit form of cdf

$$\Phi(x) = \int_{-\infty}^{x_1} \dots \int_{-\infty}^{x_d} f(u) \, d u.$$



Multivariate *t*-distribution

- The multivariate *t*-distribution is closely related to the multivariate normal distribution.
- For $Z \sim N_d(0, I_d)$ and $Y^2 \sim \chi_m^2$ as two independent random variables, a *t*-distributed random variable T with m degrees of freedom is defined as,

$$T = \sqrt{m} \frac{Z}{Y}$$



R packages for multivariate normal density

- ❑ `fmultivar`, `mvtnorm` and `mnormt` packages can be used.
we set $\sigma_{12} = \sigma_{21} = 0.7$, $\sigma_{11} = \sigma_{22} = 1$, $\mu = (0, 0)^\top$ at the point $x = (0.3, 0.4)^\top$.

```
1 > 1.sigma = matrix(c(1,0.7,0.7,1),ncol=2)
2 >     1.mu = c(0,0)
3 >     x = c(0.3,0.4)
4 > dmvnorm(x,mean=1.mu,sigma=1.sigma)
5                                     # from package mvtnorm
6 [1] 0.2056464
```



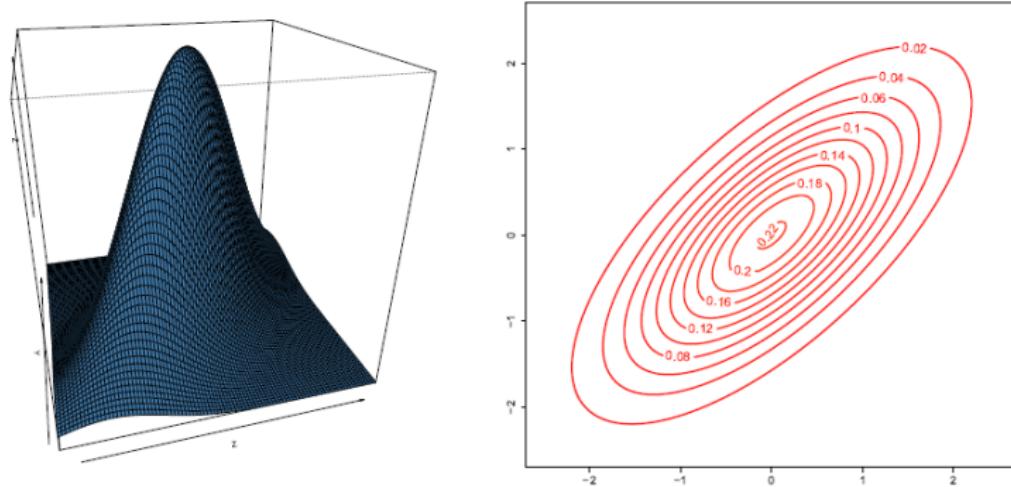


Figure 44: Density plot and contour plot of a 2-dimensional normal distribution with $\rho_{12} = 0.7$, $\mu = (0, 0)^\top$, $\sigma_{12} = \sigma_{21} = 0.7$, $\sigma_{11} = \sigma_{22} = 1$.

BCS_BinormalDensity



Geometry of the $N_p(\mu, \Sigma)$ distribution

- Let the density of $N_p(\mu, \Sigma)$ be constant on ellipsoid with the form,

$$(x - \mu)^\top \Sigma^{-1} (x - \mu) = a^2.$$

- Theorem 7.1**

If $X \sim N_d(\mu, \Sigma)$, then the random variable

$U = (X - \mu)^\top \Sigma^{-1} (X - \mu)$ has a χ_d^2 distribution.



Geometry of the $N_p(\mu, \Sigma)$ distribution

□ Theorem 7.2 (Mahalanobis transformation)

Let $X \sim N_d(\mu, \Sigma)$ and $Y = \Sigma^{-1/2}(X - \mu)$, then
 $Y \sim N_d(0, I_d)$.

□ Theorem 7.3

Let $X \sim N_d(\mu, \Sigma)$, \mathcal{A} ($p \times p$) and $b \in \mathbb{R}^d$, where \mathcal{A} is nonsingular, then $Y = \mathcal{A}X + b$ is again a d -dimensional Normal: $Y \sim N_d(\mathcal{A}\mu + b, \mathcal{A}\Sigma\mathcal{A}^\top)$.



R packages for multivariate normal distribution

- The first algorithm is based on the randomized Quasi-Monte-Carlo procedure by Genz (1992) and Genz (1993) that is applicable to singular and non-singular covariance structures of dimension $d \leq 1000$.
- The second algorithm by Miwa *et al.* (2003) is only applicable for small dimension $d \leq 20$ and non-singular covariance matrix.



R packages for multivariate normal distribution

```
1 > pmvnorm(x, mean=1.mu, sigma=1.sigma)
2 [1] 0.2437731
3 > attr(, "error")
4 [1] 1e-15
5 > attr(, "msg")
6 'Normal Completion'
```



Multivariate normal distribution in R

	fMultivarz $(d = 2)$	mvtnorm $(d \geq 2)$	mnormt $(d \geq 2)$
cdf (probability)	pnorm2d	pmvnorm	pmnorm
pdf (density)	dnorm2d	dmvnorm	dmmnorm
simulation	rnorm2d	rmvnorm	rmvnorm
quantiles	n.a.	qmvnorm	n.a.

Table 5: Multivariate normal distribution in R



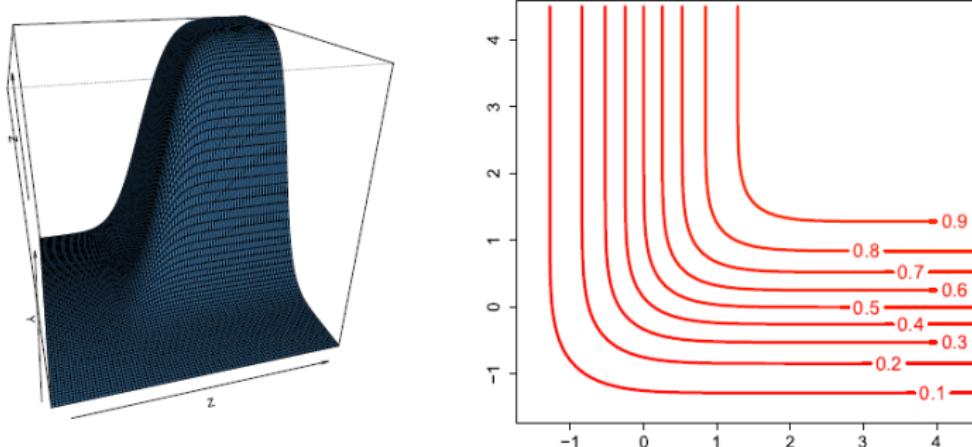


Figure 45: (left) The cdf of a 2-dimensional Gaussian distribution with $\rho_{12} = 0.2$ and (right) the corresponding contour.

 BCS_NormalProbability



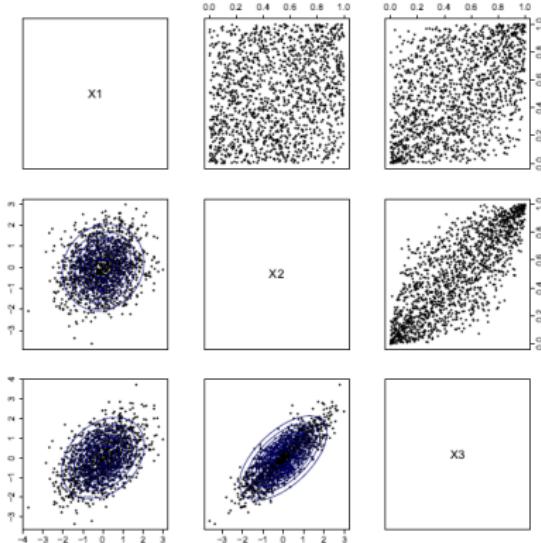


Figure 46: Sample \mathcal{X} from N_3 , with $\rho_{12} = 0.2$, $\rho_{13} = 0.5$, $\rho_{23} = 0.8$ and $n = 1500$.

BCS_NormalCopulaContour



Sampling distribution and limit theorem

□ Theorem 7.4

Let X_1, \dots, X_n be *i.i.d.* with $X_i \sim N_d(\mu, \Sigma)$, then

$$\bar{x} \sim N_d(\mu, n^{-1}\Sigma).$$

□ Theorem 7.5 (Central Limit Theorem-CLT)

Let X_1, \dots, X_n be *i.i.d.* with $X_i \sim (\mu, \Sigma)$, then the distribution of $\sqrt{n}(\bar{x} - \mu)$ is asymptotically $N_d(0, \Sigma)$ distributed, i.e.

$$\sqrt{n}(\bar{x} - \mu) \xrightarrow{\mathcal{L}} N_d(0, \Sigma) \quad \text{as } n \rightarrow \infty.$$



Sampling distribution and limit theorem

□ Corollary 7.1

If $\widehat{\Sigma}$ is a consistent estimate for Σ , then the CLT still holds, such that

$$\sqrt{n}\widehat{\Sigma}^{-1/2}(\bar{x} - \mu) \xrightarrow{\mathcal{L}} N_d(0, \mathcal{I}_d) \quad \text{as } n \rightarrow \infty.$$



Multivariate Distributions

Motivation

Distribution and density function

Multivariate normal distribution

Copula

Copula estimation

Copula

□ Theorem 7.6 (Sklar(1959))

Let F be a multivariate distribution function with margins F_1, \dots, F_d , then there exists the copula C such that,

$$F(x_1, \dots, x_d) = C \{F_1(X_1), \dots, F_d(X_d)\}, \quad x_1, \dots, x_d \in \mathbb{R}.$$

If F_i is continuous for $i = 1, \dots, d$, then C is unique.

Otherwise, C is uniquely determined on $F_1(\mathbb{R}) \times \dots \times F_d(\mathbb{R})$.



Copula

- We can determine the copula of an arbitrary continuous multivariate distribution from the transformation,

$$C(u_1, \dots, u_d) = F\{F_1^{-1}(u_1), \dots, F_d^{-1}(u_d)\}, \quad u_1, \dots, u_d \in [0, 1],$$

where F_i^{-1} is the i -th inverse marginal distribution function.

- The copula density and the density of the multivariate distribution with respect to the copula are,

$$c(u) = \frac{\partial^d C(u)}{\partial u_1, \dots, \partial u_d}, \quad u \in [0, 1]^d.$$

$$f(x_1, \dots, x_d) = c\{F_1(x_1), \dots, F_d(x_d)\} \prod_{i=1}^d f_i(x_i), \quad x_1, \dots, x_d \in \mathbb{R}.$$



R packages to estimate copula

- R packages: `copula`, `fCopulae`, `fgac`, `gumbel`, `HAC` and `sbgcop`.
- We concentrate on the two packages `copula` and `fCopulae`.
- Example: `mvdc` class in the `copula` package.
- The object describes a bivariate Gaussian copula with correlation parameter $\rho = 0.75$ with two $N(0, 2)$ margins.

```
1 > mvdc.gauss.n.e = mvdc(normalCopula(0.75), c("norm",  
+ "exp"), list(list(mean=0, var=2), list(rate=2)))
```



Fréchet-Hoeffding bounds and product copula

- Every copula C satisfies

$$W^d(u_1, \dots, u_d) \leq C(u_1, \dots, u_d) \leq M^d(u_1, \dots, u_d).$$

- Upper and lower bounds,

$$M^d(u_1, \dots, u_d) = \min(u_1, \dots, u_d),$$

$$W^d(u_1, \dots, u_d) = \max\left(\sum_{i=1}^d u_i - d + 1, 0\right).$$

- The functions M^d and Π^d are d -copulas for all $d \geq 2$, the function W^d is not a d -copula for any $d > 2$.

- Product copula $\Pi^d(u_1, \dots, u_d) = \prod_{j=1}^d u_j.$



Elliptical copula

- The Gaussian copula,

$$C_N(u_1, \dots, u_d, \Sigma) = \Phi_{\Sigma} \{ \Phi^{-1}(u_1), \dots, \Phi^{-1}(u_d) \}.$$

- The Gaussian copula density,

$$c_N(u_1, \dots, u_d, \Sigma) = |\Sigma|^{-\frac{1}{2}} \exp \left[-\frac{\{\Phi^{-1}(u_1), \dots, \Phi^{-1}(u_d)\}^T (\Sigma^{-1} - I_d) \{\Phi^{-1}(u_1), \dots, \Phi^{-1}(u_d)\}}{2} \right],$$

for all $u_1, \dots, u_d \in [0, 1]$.



Elliptical copula

- The t -copula,

$$\begin{aligned} C_t(u_1, u_2, \nu, \delta) &= \int_{-\infty}^{t_\nu^{-1}(u_1)} \int_{-\infty}^{t_\nu^{-1}(u_2)} \frac{\Gamma(\frac{\nu+2}{2})}{\Gamma(\frac{\nu}{2})\pi\nu\sqrt{(1-\delta^2)}} \\ &\quad \times \left\{ 1 + \frac{x_1^2 - 2\delta x_1 x_2 + x_2^2}{(1-\delta^2)\nu} \right\}^{-\frac{\nu}{2}-1} dx_1 dx_2. \end{aligned}$$

- The t -copula density,

$$c_t(u_1, u_2, \nu, \delta) = \frac{f_{\nu\delta}\{t_\nu^{-1}(u_1), t_\nu^{-1}(u_2)\}}{f_\nu\{t^{-1}(u_1)\}f_\nu\{t^{-1}(u_2)\}}, \quad u_1, u_2, \delta \in [0, 1].$$



R Packages for elliptical copula

As the `copula` package has the most advanced copula modeling, hence we pay special attention to it. We use the following functions: `normalCopula`, `tCopula` or `ellipCopula`

```

1 > norm.cop = normalCopula(param = c(0.5,0.6,0.7), dim = 3,
2   dispstr = "un")                                # 3D Gaussian copula
3 > t.cop = tCopula(param = c(0.5,0.3), dim = 3, df = 2,
4   dispstr = "toep")                             # 3D t-copula
5
6 > norm.cop = ellipCopula(family = "normal", param = c
7   (0.5,0.6,0.7),dim = 3,dispstr = "un")        # 3D Gaussian copula

```



R packages for elliptical copula

We use `rCopula`, `dCopula` or `pCopula` for simulation, calculating of the density and distribution function individually.

```
1 > rCopula(1000, norm.cop)    # simulate from a Gaussian  
2   copula  
3 > dCopula(c(0.2,0.5,0.1), norm.cop)  
4   [1] 1.103629  
5                               # evaluate the copula density  
6 > pCopula(c(0.2,0.5,0.1), t.cop)  
7   [1] 0.04190934  
8                               # evaluate the 3D t-copula
```



R packages for plotting

R functions: `plot`, `persp` and `contour` are used for plotting for a Gaussian copula with $d = 2$.

```
1 > norm.2d.cop = normalCopula(param = 0.7, dim = 2)
2                               # construct a 2D Gaussian copula
3 > plot(rCopula(1000, norm.2d.cop)) # scatterplot
4 > persp(norm.2d.cop, pCopula)      # 3D copula plot
5 > contour(norm.2d.cop, pCopula)    # copula contour curves
6 > persp(norm.2d.cop, dCopula)
7                               # 3D plot of the copula density
8 > contour(norm.2d.cop, dCopula)
9                               # contour curves of the density
```



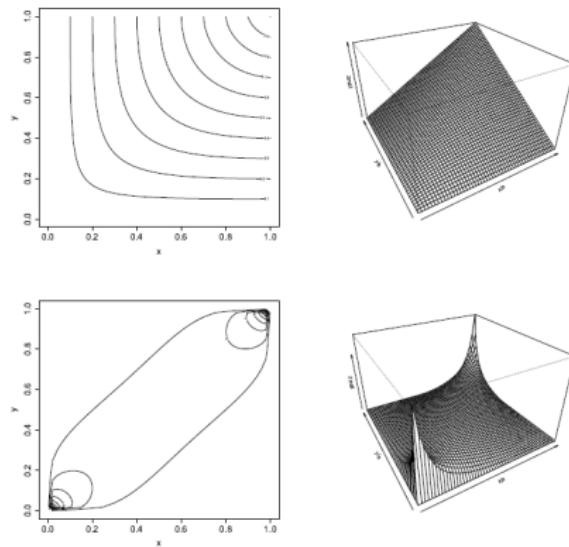


Figure 47: 2-dimensional Gaussian Copula $\rho = 0.7$ (from top to bottom: distribution and density function)

 BCS_NormalCopula



Archimedean copula

- d -dimension archimedean copula,

$$C(u_1, \dots, u_d) = \phi \left\{ \phi^{-1}(u_1) + \dots + \phi^{-1}(u_d) \right\}, \quad u_1, \dots, u_d \in [0, 1],$$

where $\phi \in \mathcal{L}$ is called the generator of the copula,

$$\mathcal{L} = \left\{ \phi : [0; \infty] \mid \phi(0) = 1, \phi(\infty) = 0; (-1)^j \phi^{(j)} \geq 0; j = 1, \dots, \infty \right\}$$



Frank copula

- The generator function and the copula function are given as follows,

$$\begin{aligned}\phi(x, \theta) &= \theta^{-1} \log \left\{ 1 - (1 - e^{-\theta}) e^{-x} \right\}, \\ -\infty < \theta < \infty, \quad \theta &\neq 0, \quad x \in [0, \infty),\end{aligned}$$

$$C_\theta(u_1, \dots, u_d) = -\frac{1}{\theta} \log \left[1 + \frac{\prod_{j=1}^d \{ \exp(-\theta u_j) - 1 \}}{\{ \exp(-\theta) - 1 \}^{d-1}} \right].$$

- The dependence is maximized when $\theta \rightarrow \infty$ and the independence is achieved when $\theta = 0$.



Gumbel copula

- The generator function and the copula function are given as follows:

$$\phi(x, \theta) = \exp \left\{ -x^{\frac{1}{\theta}} \right\}, \quad 1 \leq \theta < \infty, x \in [0, \infty),$$

$$C_\theta(u_1, \dots, u_d) = \exp \left[- \left\{ \sum_{j=1}^d (\log u_j)^\theta \right\}^{\theta^{-1}} \right].$$

- The Gumbel copula leads to asymmetric contour diagrams. For $\theta > 1$, this copula allows for the generation of dependence in the upper tail. For $\theta = 1$, the Gumbel copula reduces to the product copula and for $\theta \rightarrow \infty$, we obtain the Frèchet-Hoeffding upper bound.



Clayton copula

- The generator function and the copula function are given as follows:

$$\phi(x, \theta) = (\theta x + 1)^{-\frac{1}{\theta}},$$

$$-1/(d-1) \leq \theta < \infty, \theta \neq 0, x \in [0, \infty),$$

$$C_\theta(u_1, \dots, u_d) = \left\{ \left(\sum_{j=1}^d u_j^{-\theta} \right) - d + 1 \right\}^{-\theta^{-1}}.$$

- For $\theta \rightarrow \infty$, the dependence arrives the maximal and for $\theta \rightarrow 0$, we have the independence. For $\theta \rightarrow -1$ in the bi-variate case, the distribution tends to the lower Fréchet bound.



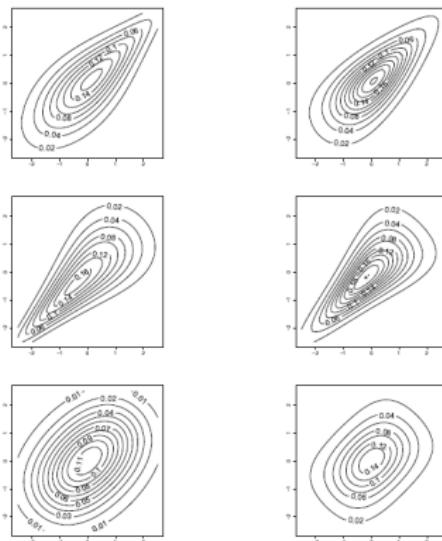


Figure 48: Contour plots for Gumbel, Clayton and Frank copulas (from top to bottom) with Normal (left column) and t_6 distributed (right column) margins.

 BCS_ArchimedeanContour



Hierarchical archimedean copula

- The fully nested hierarchical archimedean copula is as follows,

$$\begin{aligned}
 C(u_1, \dots, u_d) &= \phi_{d-1}^{-1} \circ \phi_{d-2}(\dots [\phi_2^{-1} \circ \phi_1 \\
 &\quad \{\phi_1^{-1}(u_1) + \phi_1^{-1}(u_2)\} \\
 &\quad + \phi_2^{-1}(u_3)] + \dots + \phi_{d-2}^{-1}(u_{d-1}) + \phi_{d-1}^{-1}(u_d)\} \\
 &= \phi_{d-1}[\phi_{d-1}^{-1} \circ C(\{\phi_1, \dots, \phi_{d-2}\})(u_1, \dots, u_{d-1}) \\
 &\quad + \phi_{d-1}^{-1}(u_d)].
 \end{aligned}$$

For $\phi_{d-i}^{-1} \circ \phi_{d-j} \in \mathcal{L}^*$, $i < j$, where

$$\begin{aligned}
 \mathcal{L}^* = \{ \omega : [0; \infty) \rightarrow [0, \infty) \mid \omega(0) &= 0, \\
 \omega(\infty) &= \infty; (-1)^{j-1} \omega^{(j)} \geq 0; j = 1, \dots, \infty \}.
 \end{aligned}$$



Illustration of HAC

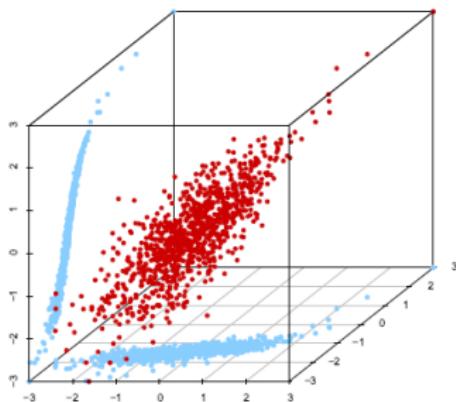


Figure 49: Scatterplot of a 3-dimensional HAC. $F(x_1, x_2, x_3) = C_{Gumbel}[C_{Gumbel}\{\Phi(x_1), t_2(x_2); \theta_1 = 2\}, \Phi(x_3); \theta_2 = 10]$.

BCS_HAC



Multivariate Distributions

Motivation

Distribution and density function

Multivariate normal distribution

Copula

Copula estimation

Copula estimation

□ Assumption:

1. Let X be a d -dimensional random variable with density f .
2. Let X be the univariate marginal distribution
 $F_j(x_j; \alpha_j)$, $j = 1, \dots, d$.
3. Let a copula belong to a parametric family $\mathcal{C} = \{C_\theta, \theta \in \Theta\}$.
4. For a sample of observations $\{x_i\}_{i=1}^n, x_i = (x_{1i}, \dots, x_{di})^\top$.
5. The likelihood function:

$$L(\alpha; x_1, \dots, x_n) = \prod_{i=1}^n f(x_{1i}, \dots, x_{di}; \alpha_1, \dots, \alpha_d, \theta).$$



Full maximum likelihood estimation

- The log-likelihood function:

$$\begin{aligned}\ell(\alpha; x_1, \dots, x_n) = & \sum_{i=1}^n \log c\{F_1(x_{1i}; \alpha_1), \dots, F_d(x_{di}; \alpha_d); \theta\} \\ & + \sum_{i=1}^n \sum_{j=1}^d \log f_j(x_{ji}; \alpha_j).\end{aligned}$$

- The efficient and asymptotically normal estimator:

$$\hat{\alpha}_{FML} = (\hat{\alpha}_1, \dots, \hat{\alpha}_d, \hat{\theta})^\top = \arg \min_{\alpha} \ell(\alpha).$$



IFM (inference for margins) method

- Estimate the parameter α_j from the marginal distributions in the first step.
- Estimate the dependence parameter θ in the second step.
- Maximize the pseudo log-likelihood function over θ to get the dependence parameter estimate $\hat{\theta}$,

$$\ell(\theta, \hat{\alpha}_1, \dots, \hat{\alpha}_d) = \sum_{i=1}^n \log c\{F_1(x_{1i}; \hat{\alpha}_1), \dots, F_d(x_{di}; \hat{\alpha}_d); \theta\}.$$



CML (canonical maximum likelihood) method

- Normalize the empirical cdf not by n but by $n + 1$

$$\hat{F}_j(x) = \frac{1}{n+1} \sum_{i=1}^n I(x_{ji} \leq x).$$

- The copula parameter estimator $\hat{\theta}_{CML}$ is given by:

$$\hat{\theta}_{CML} = \operatorname{argmax}_{\theta} \sum_{i=1}^n \log c\{\hat{F}_1(x_{1i}), \dots, \hat{F}_d(x_{di}); \theta\}.$$



R packages for copula estimation

The package `copula` has contained comprehensive methods for copula parameter estimation. An example for Gumbel copula estimation is given as follows,

```
1 > gumbel.cop = gumbelCopula(3, dim=2)
2 > n = 200
3 > x = rCopula(gumbel.cop, n)    # true observations
4 > u = apply(x, 2, rank) / (n+1) # pseudo-observations
5 > fitCopula(gumbel.cop, u, method='itau')@estimate
   [1] 3.150728          # inverting Kentall's tau
7 > fitCopula(gumbel.cop, u, method='irho')@estimate
   [1] 3.136526          # inverting Spearman's rho
9 > fitCopula(gumbel.cop, u, method='mpl')@estimate
   [1] 3.142805          # maximum pseudo-likelihood
```



The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion

Motivation

- How to model dependence between one variable and the group of variables?
- How to describe object with many characteristics in a simple way?
- How to divide objects into groups in the most natural way?
- How to predict the group of the object knowing other characteristics?



Why use R?

- Used by the majority of academic statisticians
- Platform independent
- Unrivaled help resources
- The best graphics
- The command-line interface
- Flexibility

Why not?

- Bugs



Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion

Multiple linear regression model

Application Describing dependence of one variable from several explanatory variables. Prediction.

Model

$$y = \mathcal{X}\beta + \varepsilon$$

Estimation

$$\hat{\beta} = \arg \min_{\beta \in \Theta} (y - \mathcal{X}\beta)^\top (y - \mathcal{X}\beta) = (\mathcal{X}^\top \mathcal{X})^{-1} \mathcal{X}^\top y$$

Testing

$$H_0 : \beta_j = 0 \text{ versus } H_1 : \text{no constraints}$$

Test statistics

$$t = \frac{\hat{\beta}_j}{SE(\hat{\beta}_j)} \stackrel{H_0}{\sim} t_{1-\alpha/2, n-(p+1)}$$



Example : US cereal

Example 15

The data come from the 1993 ASA Statistical Graphics Exposition and contain information about calories, fat, sugars and carbohydrates. Number of observations is 65.

The aim is to find dependence between calories and other variables.

Use the R package MASS to load the data UScereal and its function lm to perform analysis.



Estimating the model

```
1 > fit = lm( calories ~ ., data = UScereal_edited )
2 > summary(fit)
3 Call:
4 lm(formula = calories ~ protein + fat + carbo + sugars,
5 data = UScereal)
6
7 Coefficients:
8             Estimate Std. Error t value Pr(>|t|)
9 (Intercept) -18.7698    3.5127  -5.343 1.49e-06 ***
10 protein      4.0506    0.5438   7.449 4.28e-10 ***
11 fat          8.8589    0.7973  11.111 3.41e-16 ***
12 carbo        4.9247    0.1587  31.040 < 2e-16 ***
13 sugars       4.2107    0.2116  19.898 < 2e-16 ***
14 ---
15
16 Residual standard error: 8.862 on 60 degrees of freedom
17 Multiple R-squared: 0.9811,      Adjusted R-squared: 0.9798
```



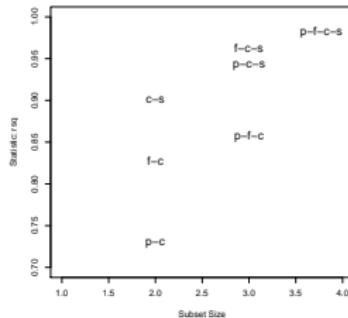


Figure 50: All subsets regression (c - carbohydrates, p - proteins, f - fat, s - sugars)

 BCS _ MLRleaps



Choosing the model

```
1 > stepAIC(fit, direction = "both")
2
3 Start: AIC=288.43
4 calories ~ protein + fat + carbo + sugars
5 Coefficients:
6 (Intercept)      protein        fat        carbo        sugars
7      -18.770       4.051      8.8589     4.9247      4.2107
```

```
1 > library ( leaps )
2 > leaps <- regsubsets ( calories ~ ., data = UScereal _ 
   edited )
3 > plot (leaps , scale = "r2")
```



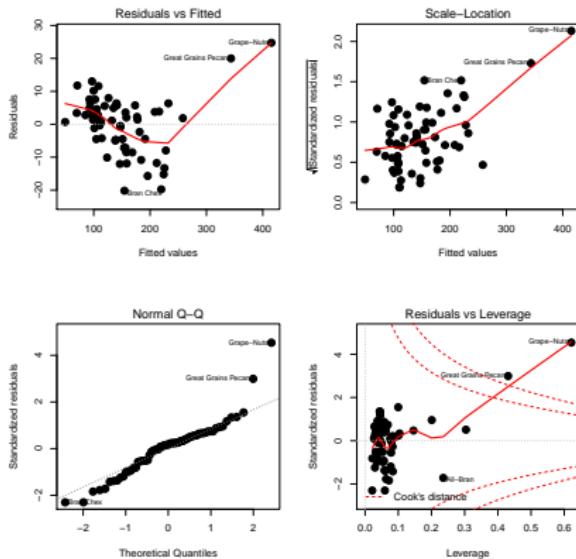


Figure 51: Diagnostic plots for multiple linear regression

BCS_MLRdiagnostic



Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion

Principal Component Analysis

Application Reduce the dimension of the data. Make the data appropriate for another multidimensional techniques.

Model

$$\max_{\{\delta: \|\delta\|=1\}} \delta^\top \text{Var}(X) \delta$$

Estimation

$$Y = \Gamma^\top (X - \mu)$$

Selecting the number of components

- Take eigenvalues which are greater than average
- Use scree diagram or log-eigenvalue diagram

Interpret the result

- Look at correlations between original variables and principal components



Example : Swiss banknotes

The swiss banknotes data set contains 7 variables and 200 observations. The first 6 variables are certain measures of explanatory variables of swiss banknotes (e.g. length, width etc.)

The aim is to reduce the number of variables.

Use the R package ncomplete to load the data Banknotes and function princomp (package stats) to perform analysis.



Estimating the model

```
1 > data(Banknotes)
2 > mydata<-as.data.frame(Banknotes[,1:ncol(Banknotes)-1])
3 > fit <- princomp(covmat=(n-1)/n*cov(mydata))
4 > summary(fit)
5 Importance of components:
6
7
8
9
```

	Comp.1	Comp.2	Comp.3	Comp.4	Comp.
Standard deviation	1.728	0.965	0.4921	0.4401	.5 Comp.6
	0.2912	0.1880			
Proportion of Variance	0.668	0.208	0.0542	0.0433	
	0.0190	0.0079			
Cumulative Proportion	0.668	0.876	0.9298	0.9731	
	0.9921	1.0000			



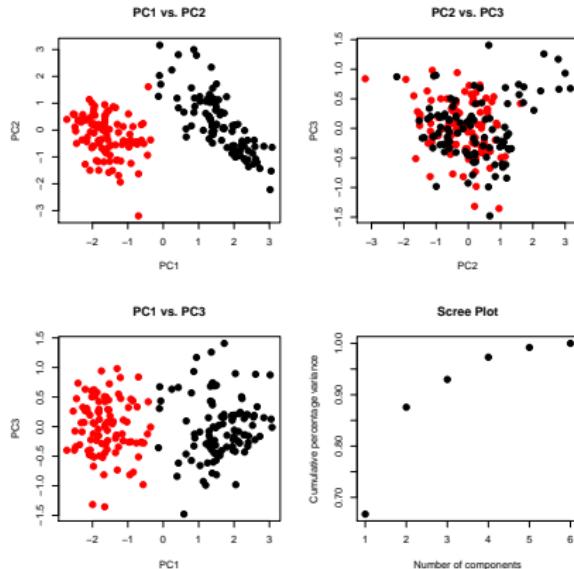


Figure 52: Principal components and scree plot for Swiss Banknote dataset

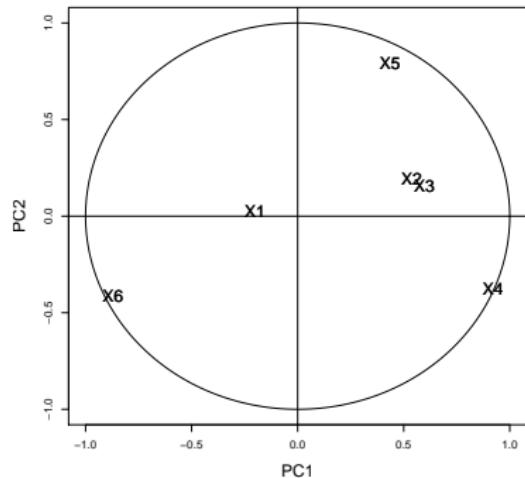
 BCS _ PCAvar



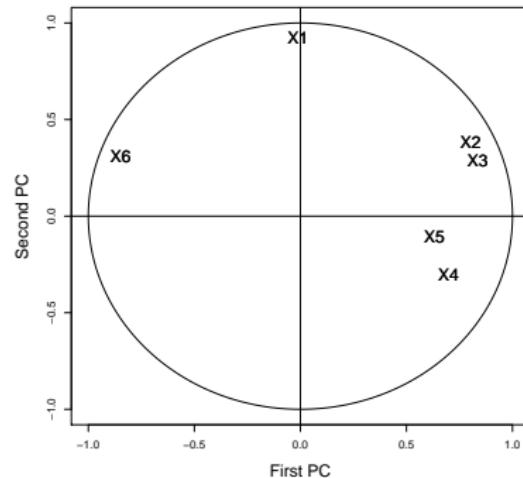
Interpretation

```
1 > cor(fit$scores,mydata)
2          V1      V2      V3      V4      V5      V6
3 Comp.1 -0.2014  0.538   0.597   0.9212  0.43526 -0.87023
4 Comp.2  0.0275  0.191   0.159  -0.3770  0.79422 -0.41011
5 Comp.3 -0.4275 -0.354  -0.421  -0.0745 -0.34205 -0.25338
6 Comp.4  0.6581  0.557   0.453  -0.0568 -0.24765 -0.09896
7 Comp.5  0.5834 -0.280  -0.386   0.0202  0.03705 -0.02140
8 Comp.6 -0.0491  0.400  -0.295   0.0029  0.00818  0.00747
9 > ucircle=cbind(cos((0:360)/180*pi),sin((0:360)/180*pi))
10 > plot(ucircle,type="l",lty="solid",xlab="First PC",ylab="Second PC")
11 > abline(h=0.0,v=0.0)
12 > label=c("X1","X2","X3","X4","X5","X6")
13 > text(cor(mydata,fit$scores),label,cex=0.8)
14 > text(cor(mydata,fit$scores),label,cex=0.8)
```





BCS_PCAbiplot



BCS_NPCAabiplot

Figure 53: The correlation of the original variable with the PCs (left) and normalized PCs (right)



Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion

Factor Analysis

Application Reduce the dimension of the data. Find unobservable factors.

Model

$$\mathcal{X} = \mathcal{Q} \cdot \mathcal{F} + \mathcal{U} + \mu$$

Estimation

$$S = \hat{\mathcal{Q}}\hat{\mathcal{Q}}^\top + \hat{\psi}$$

- Maximum likelihood, principal components, principal factors

Interpret the result

- Look at correlations between original variables and factors

Testing

- Only for ML method testing the hypothesis that k factors are sufficient is possible.



Example : Decathlon

A data frame with 41 rows and 13 columns: the first ten columns corresponds to the performance of the athletes for the 10 events of the decathlon. The columns 11 and 12 correspond respectively to the rank and the points obtained. The last column is a categorical variable corresponding to the sporting event (2004 Olympic Game or 2004 Decastar).

The aim is to reduce the number of variables.

Load data `decathlon` from package `FactoMineR`. Use the R package `stats` and function `factanal` to perform factor analysis.



Estimating the model

```
1 > require("stats")
2 > fit = factanal(mydata,          # fit the model
3 +                  factors = 3,
4 +                  rotation="none")
5 > fit                         # print the results
6
7 Call:
8 factanal(x = mydata, factors = 3, rotation = "none")
9
10 Uniquenesses:
11      100m      Long.jump   Shot.put   High.jump   400m
12      0.411      0.396      0.106      0.697      0.264
13     110m.hurdle   Discus     Pole.vault  Javeline   1500m
14      0.491      0.534      0.907      0.785      0.005
```



Testing

	Factor1	Factor2	Factor3
1 SS loadings	2.35	1.79	1.26
2 Proportion Var	0.24	0.18	0.13
3 Cumulative Var	0.24	0.41	0.54

- 1 Test of the hypothesis that 3 factors are sufficient .
- 2 The chi square statistic is 17.97 on 18 degrees of freedom
- 3 The p- value is 0.457



Interpreting the result

In the 1st factor the largest factor loadings are for 100 m, 400 m, 110 m hurdle run, and long jump. This factor can be interpreted as the "sprinting performance". The loadings on the 2nd factor

present a counter-intuitive throwing-jumping combination: the highest loadings are for throwing events (discus throwing and shot) and also for jumping events. On the 3rd factor the largest loading is

for 1500 m running. First and second factors can be straightforward interpreted as the "sprinting abilities" and "endurance".



Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion

Cluster Analysis

Application Search for groups in a priori unclassified multivariate data.

Model

- Data driven algorithm

Clustering algorithm

- Hierarchical (agglomerative, divisive), k-means clustering.
Depend on distance function (e.g. Euclidean) and linkage criterion (e.g. single linkage, Ward, etc.)



Example : Agriculture data

Gross National Product (GNP) per capita and percentage of the population working in agriculture for each country belonging to the European Union in 1993.

Obtain such clusters that objects within the clusters are very similar and different from objects in another clusters.

Use the R package `cluster` and function `hclust` for analysis and its dataset `agriculture'`.



Example : Finding the clusters

```
1 > mydata <- agriculture
2 > mydata <- na.omit(mydata)
3 > mydata <- scale(mydata)
4 > d <- dist(mydata, method = "euclidean")
5 > print(d[1:2, 1:2], digits = 2)
       DK    D
DK  1.02
D   0.40  0.63
9 > fit <- hclust(d, method = "ward")
10 > plot(fit, xlab = NULL)
11 > groups <- cutree(fit, k=5)
12 > rect.hclust(fit, k=5, border = "red")
```



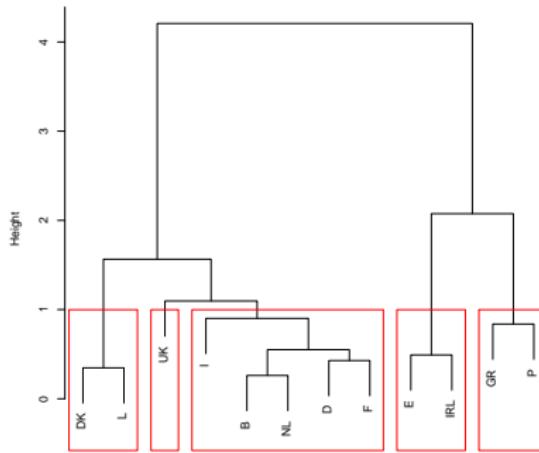


Figure 54: Agglomerative hierarchical clustering for agriculture data

BCS CAComplete



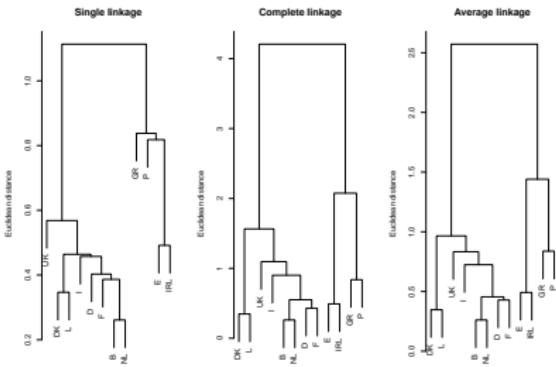


Figure 55: Comparison of different clustering algorithms

 BCS_CAmethods

Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion

Metric Multidimensional Scaling

Application Visualization of proximities in a low dimensional space.

Model

$$\min_{\mathcal{Y}} \sum_{i=1}^n \sum_{j=1}^n (d_{ij}^{\mathcal{X}} - d_{ij}^{\mathcal{Y}})^2$$

Estimation

$$\mathcal{Y} = \Lambda^{1/2} \Gamma^\top$$

- The solution for MDS is identical to PCA as far as Euclidean distance is concerned
- $P = \frac{\sum_{i=1}^p |\lambda_i|}{\sum_{i=1}^n |\lambda_i|} \geq 0.8$ or/and $P' = \frac{\sum_{i=1}^p \lambda_i^2}{\sum_{i=1}^n \lambda_i^2} \geq 0.8$ suggests a reasonable fit.



Example : Cars93

Information about manufacturer, model, type, minimal, maximal and average price, city mpg, highway mpg, air bags standard, drive train type, number of cylinders, engine size, horsepower etc.

Compare cars with rear drive train and with less than 18 mpg in the city.

Use the R function cmdscale from package stats. Load the data Cars93 from package MASS.



Example : Finding the clusters

```
1 > mydata = Cars93 [ which ( Cars93 $ DriveTrain == "Rear "
2   &
3 > Cars93 $MPG .city <=18) ,]
4 > mydata = mydata [,c(5, 7, 8, 11:15 , 17:19 , 20:25)]
5 > mydata <-na.omit (mydata )
6 > d = dist (mydata , method = "euclidean ")
```

```
1 > fit = cmdscale (d)
```

```
1 > plot (fit [,1], fit [,2], xlab = "", ylab = "", type = "n")
2 > segments ( -1000 , -0, 1500 , 0, lty="dotted ")
3 > segments (0, -500, 0, 500 , lty="dotted ")
4 > text ( fit [,1], fit [,2], rownames ( mydata ), cex
      =0.8)
```



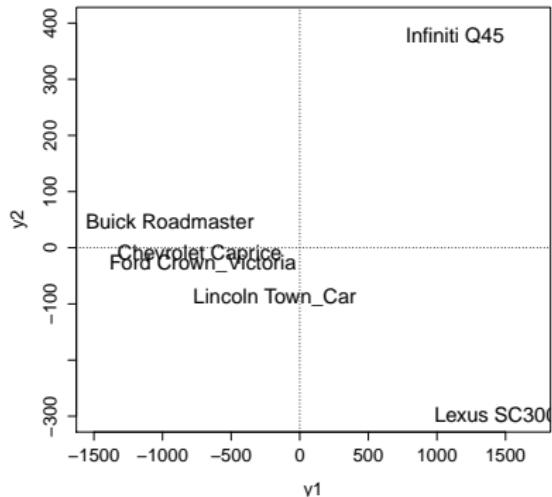


Figure 56: Configuration of the MDS of the American car subsample

BCS_MDS



Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion

Example : voting (nonmetric multidimensional scaling)

Voting results of 15 congressmen from New Jersey on 19 environmental bills.

Compare results of the voting between Democratic congressmen and Republicans.

Use the R function `isoMDS` (package MASS). The dataset `voting` is from package HSAUR2.



Example : Finding the clusters

```
1 > fit = isoMDS ( voting )
2 > plot (fit $ points [,1], fit $ points [,2], xlab ="",
3   ylab ="",
4   type = "n",main = "")
5 > segments (-10, -0, 10, 0, lty = " dotted ")
6 > segments (0, -10, 0, 10, lty = " dotted ")
7 > text (fit $ points [,1], fit $ points [,2], rownames (
  voting ),
  cex =0.8)
```



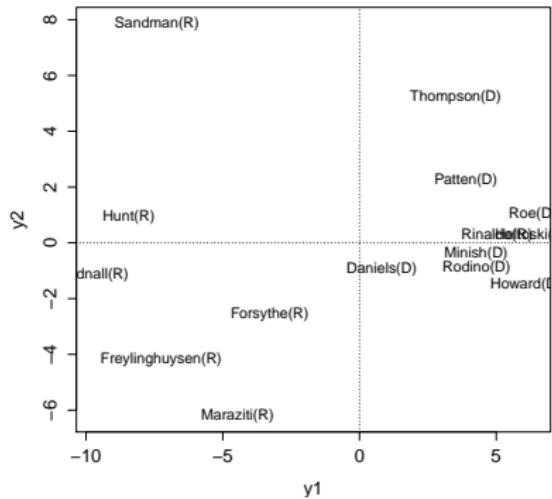


Figure 57: Configuration of the MDS of voting data

 BCS_isoMDS

Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion

Linear Discriminant Analysis

Application Allocating observations to one or another set of a priori defined classes in some optimal way.

Model

$$P(G = k \mid X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^K f_l(x)\pi_l}$$

Estimation

$$\log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^\top \Sigma^{-1}(\mu_k - \mu_l) + x^\top \Sigma^{-1}(\mu_k - \mu_l)$$



Example : Spain authors

Information about relative frequencies of 120 most frequent tag trigrams in 15 texts contributed by 3 Spanish authors (Cela, Mendoza and Vargas Llosa).

The aim of the analysis is to construct classification rule which makes able to assign text with unknown author to Cela, Mendoza or Vargas Llosa.

Use the R library MASS, function lda. The dataset spanish is from the package languageR.



```
1 > mydata = t( spanish )
2 > pca = prcomp (mydata , center = TRUE , scale = TRUE )
3 > datalda = (pca $x)
4 > datalda = datalda [ order ( rownames ( datalda )), ]
5 > data ( spanishMeta , package = " languageR ")
6 > mydata = cbind ( datalda [ ,1:2] , spanishMeta $ Author
    )
7 > colnames ( mydata ) = c("PC1", "PC2", "Author")
8 > mydata = as. data . frame ( mydata )
9 > mydata $ Author = as. factor ( mydata $ Author )
10 > fit = lda ( Author ~ PC1 + PC2 , data = mydata )
11 > pred . class = predict (fit , mydata )$ class
12 > pred . table = table ( mydata $Author , pred . class )
13 > pred . correct = diag ( prop . table ( pred . table , 1))
14 > 1 - sum ( diag ( prop . table ( pred . table )))
15 [1] 0.2
16 > partimat ( Author ~ PC1 + PC2 , data = mydata , method =
    "lda")
```



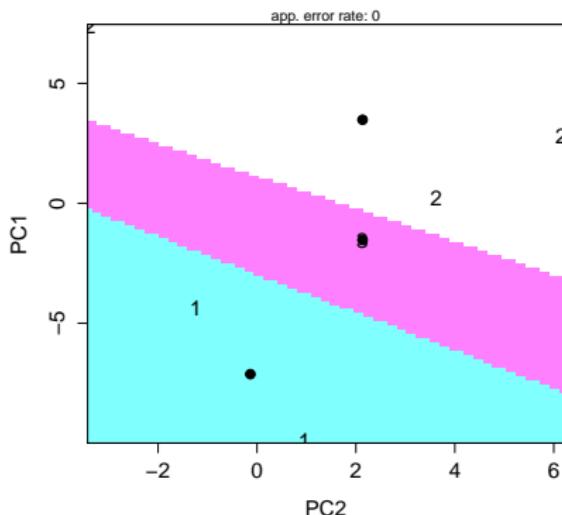


Figure 58: LDA for the Spanish author data

BCS_LDA



Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion

Outlook

- Relative importance of variables in multiple regression (bootstrap)
- Testing further hypothesis in multiple regression
- Detecting number of clusters in cluster analysis
- Testing assumptions of LDA
- Kernel discriminant analysis



Multivariate Statistical Analysis

Introduction

Multiple Linear Regression

Principal Component Analysis

Factor Analysis

Cluster Analysis

Metric Multidimensional Scaling

Non Metric Multidimensional Scaling

Linear Discriminant Analysis

Outlook

Discussion



The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

Random Numbers

Outline

Need of random numbers

Properties of random numbers

Gathering of true random numbers

Generating random numbers

Random numbers for a specific distribution

Quality of a random sequence of numbers

Outline

1. Need of random numbers
2. Properties of random numbers
3. Gathering of true random numbers
4. Generating pseudo random numbers
 - ▶ Linear Congruential Generator
 - ▶ Lagged Fibonacci Generator
 - ▶ Mersenne Twister
5. Pseudo random numbers for specific distributions
6. Quality of random numbers



Random Numbers

Outline

Need of random numbers

Properties of random numbers

Gathering of true random numbers

Generating random numbers

Random numbers for a specific distribution

Quality of a random sequence of numbers

Practical applications for random numbers

- statistical sampling
- computer simulation
- completely randomized design
- cryptography



Random Numbers

Outline

Need of random numbers

Properties of random numbers

Gathering of true random numbers

Generating random numbers

Random numbers for a specific distribution

Quality of a random sequence of numbers

Desired properties of random numbers

- Two important statistical properties:
 1. Uniformity
 2. Independence
- Random Number, R_i , must be independently drawn from a uniform distribution with pdf:

$$f(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$



Random Numbers

Outline

Need of random numbers

Properties of random numbers

Gathering of true random numbers

Generating random numbers

Random numbers for a specific distribution

Quality of a random sequence of numbers

How to get random numbers?

1. step: Measure a physical phenomenon that is expected to be random
2. step: Apply some bias correction

This would generate true random numbers, if the underlying effect is truly random.



Random Events

- Lavalamp bubbles
 - ▶ **lavarand generator** by Silicon Graphics (no longer available)



Random Events



Figure 59: Lava lamps



Random Events

- Lavalamp bubbles
- Radioactive decay



Random Events



Figure 60: Radioactive decay



Random Events

- Lavalamp bubbles
- Radioactive decay
- User input (via mixed strategy games)



Random Events



Figure 61: The standard pc user



How to get true random numbers in R?

Use package "random"!

- Harsh restrictions: max. $n = 10.000$
- In comparison to pseudo random numbers relatively slow



R package "random"

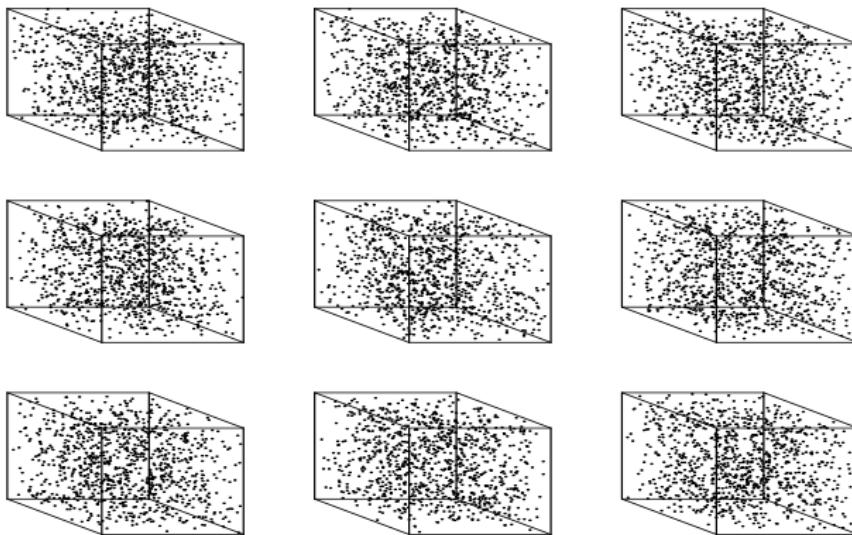


Figure 62: 3D plot of U_1 vs. U_2 vs. U_3 of true random numbers from the
BCS random package



(Dis)advantages of true random numbers

- Pro:

1. They are truly random

- Contra:

1. Often difficult to implement
2. In comparison to pseudo random numbers slow to gather
3. Results are not repeatable (bad for simulation)



Random Numbers

Outline

Need of random numbers

Properties of random numbers

Gathering of true random numbers

Generating random numbers

Random numbers for a specific distribution

Quality of a random sequence of numbers

Generating random numbers

Gimme some
random numbers.

First give me a
random number!



Figure 63: Computer are always deterministic!



What does this mean?

1. Computer are input-output systems
2. They always need a starting value called "seed"
3. Results are always determined by this value, i.e. we get always the same sequence for a specific seed



What is a random number generator?

"A random number generator (RNG) is a computational or physical device designed to generate a sequence of numbers or symbols that lack any pattern, i.e. appear random." Wikipedia



The magic function

- The magic comes from a simple function, known as **modulo**.
- The modulo operation finds the remainder of division of one number by another, for example:
 - ▶ $7 \bmod 3 = 1$
 - ▶ $9 \bmod 3 = 0$



The linear congruential generator - Remarks

- The random integers are being generated in $[0, m - 1]$, so they have to be converted via:

$$U_i = \frac{X_i}{m}$$

with $i=0,1,2,\dots$



The linear congruential generator - Remarks

- Selection of values for a, c, m and X_0 drastically affects statistical properties and cycle length
 - ▶ Full cycle length is m if and only if
 1. c and m are relatively prime, i.e. their greatest common divisor is 1
 2. $a - 1$ is divisible by all prime factors of m
 - ▶ Points plotted in n -dimensional space will lie on, at most, $m^{1/n}$ hyperplanes (Marsaglia's Theorem)
- Famous example for some bad choices is RANDU



The official IBM generator RANDU

$$X_i = (2^{16} + 3)X_{i-1} \bmod 2^{31}$$

with $0 \leq X_i \leq m$ and $i = 0, 1, 2, \dots$

Implementation in R:



The official IBM generator RANDU

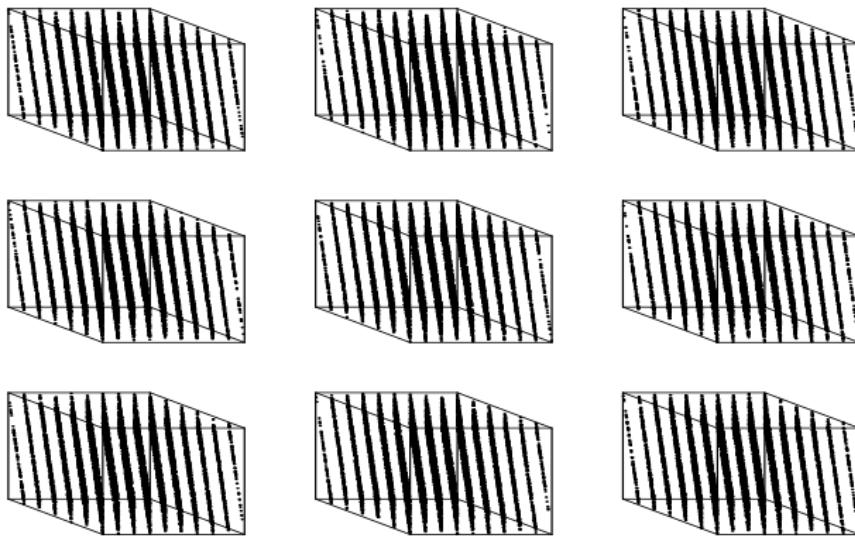


Figure 64: 3D plot of U_i vs. U_{i-1} vs. U_{i-2} from RANDU



Lagged Fibonacci generator - Method

$$X_i = (X_{i-j} + X_{i-k}) \bmod m$$

with $0 \leq X_i \leq m$ and $i, j = 0, 1, 2, \dots$ and $j, k < i$



Lagged Fibonacci generator - Remarks

- Seed is no longer a single value rather than a sequence
 - ▶ Statistical properties rely heavily on this seed!
 - ▶ The ratio of consecutive random numbers converges to the golden ratio $\frac{1+\sqrt{5}}{2}$ (≈ 1.61)
 - ▶ See "*The Art of Computer Programming*" for popular choices of j and k



Lagged Fibonacci generator - Remarks

- Maximum cycle length is $m^k - 1$ if and only if
 1. m is a prime
 2. $k < j$
- LFG is often used to generate the seed values for more complex RNGs, like the Mersenne Twister



The Mersenne Twister

- First developed in 1997 by Makoto Matsumoto and Takuji Nishimura
- Period length is chosen to be a Mersenne prime ($2^{19937} - 1$).
- Needs a seed of 624 numbers, as random as possible
- Bad choice of these numbers lead to a longer run of the algorithm to create high quality random numbers
- It passes even the high standard Diehard battery tests
- Implemented in R, Matlab, Stata, ...



The Mersenne Twister

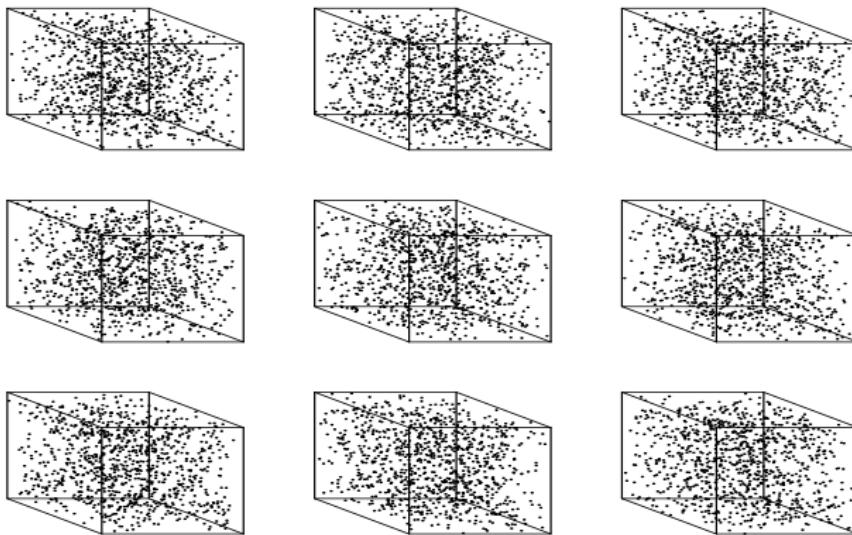


Figure 65: 3D plot of U_1 vs. U_2 vs. U_3 from the Mersenne Twister



Random Numbers

Outline

Need of random numbers

Properties of random numbers

Gathering of true random numbers

Generating random numbers

Random numbers for a specific distribution

Quality of a random sequence of numbers

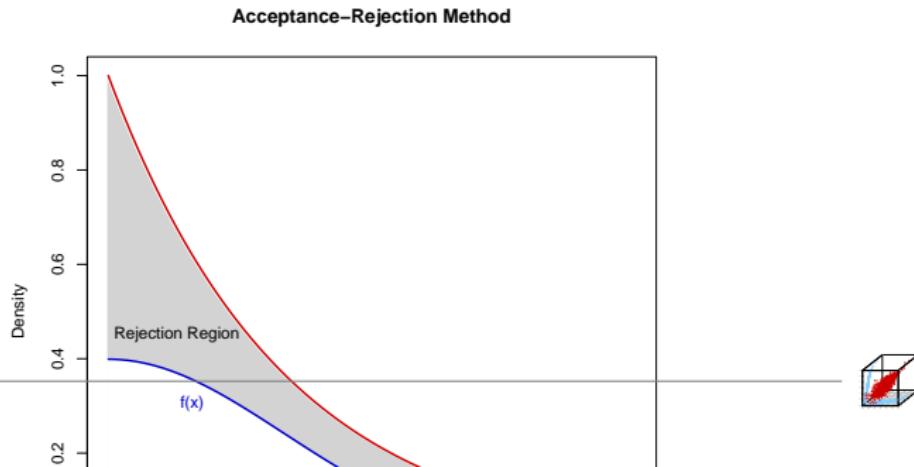
Random numbers for a specific distribution

- Several methods for generating random numbers for a specific distribution are known
 - ▶ Box/Muller method for normally distributed random numbers
 - ▶ Cheng/Feast Algorithm for generating gamma random variates
- Uniform distributed random numbers can be transformed into all other distributions
 - ▶ Acceptance/rejection methods



Acceptance/rejection method

1. Generate $U_1 \sim U_{[0,1]}$
2. Generate $U_2 \sim U_{[0,0.5]}$
3. If $U_2 \leq y$ safe tuple (U_1, U_2) , otherwise reject
4. Redo



Random Numbers

Outline

Need of random numbers

Properties of random numbers

Gathering of true random numbers

Generating random numbers

Random numbers for a specific distribution

Quality of a random sequence of numbers

Quality measurements

- Frequency tests: checking to make sure that there were roughly the same number of 0s, 1s, 2s, 3s, etc.
 - ▶ The **serial test**, did the same thing but for sequences of two digits at a time (00, 01, 02, etc.)
 - ▶ The **poker test**, tested for certain sequences of five numbers at a time (aaaaa, aaaab, aaabb, etc.) based on hands in the game poker.



Quality Measurements

- The **gap test**, looked at the distances between zeroes
- The **run test**, looked at the "runs" of certain sequences with special abilities. For example how long are the sequences of monoton de/increasing numbers.
- All this tests are included in the Diehard Battery, which is somehow the ultimate test suite since years



Thanks for paying attention!



© Alamy



The Basics of R

Numerical Techniques

Combinatorics and Discrete Distributions

Univariate Distributions

Univariate Statistical Analysis

Basic Nonparametric Methods

Multivariate Distributions

Multivariate Statistical Analysis

Random Numbers

Advanced Graphical Techniques

Advanced Graphical Techniques

Motivation

Package lattice

Package lattice

The Panel Argument and Appearance Settings

Package rgl

Package rpanel

Advanced Graphic Package in R

Data visualization is an important part of data analysis with R. However, basic R provide no capabilities for interactive plotting and dynamic graphics

- Package lattice
- Package rgl
- Package rpanel



Advanced Graphical Techniques

Motivation

Package lattice

Getting Started With lattice Function

Package lattice

The Panel Argument and Appearance Settings

Package rgl

Package rpanel

Lattice Function

Lattice Functions	Default display	Traditional Functions
barchart()	Barplot	barplot()
histogram()	Histogram	hist()
densityplot()	Conditional kernel density plot	-
dotplot()	Dotplot	dotchart()
bwplot()	Comparative Box-Whisker Plot	boxplot()
stripplot()	Stripchart	-
qqmath()	Theoretical Quantile Plot	qqplot()
qq()	Two-sample Quantile Plot	qqplot()

Table 6: High-level Functions in lattice



Lattice Function

Lattice Functions	Default display	Traditional Functions
dotplot()	Cleveland Dot Plot	dotchart()
xyplot()	Scatter Plot	plot()
contourplot()	Contour Plot of Surfaces	contour()
cloud()	3D Scatter Plot	-
levelplot()	Level Plot of Surfaces	image()
parallel()	Parallel Coordinates Plot	parcoord()
splom()	Scatter Plot Matrix	pairs()
wireframe()	3D Perspective Plot of Surfaces	persp()

Table 7: High-level Functions in lattice



Advanced Graphical Techniques

Motivation

Package lattice

Package lattice

The formula argument

The Panel Argument and Appearance Settings

Package rgl

Package rpanel

Formula Argument

- The syntax of the formula in lattice differs from the formula used in the lm() linear model function (see Chapter5)
- The variable on the left side of \sim is a dependent variable, while the independent variables are placed on the right side
- In order to define multiple dependent or independent variables the sign $+$ is placed between them
- To produce conditional plots, the conditioning variable should be also specified in the formula argument, standing after the $|$ symbol



Formula Argument

Notation	Explanation	Example of the function
$\sim x$	plots a single variable x	<code>bwplot()</code> , <code>histogram()</code>
$y \sim x$	plots variable y against variable x	<code>xyplot()</code> , <code>qq()</code>
$y \sim x * z$	plots three variables level	<code>plot()</code> , <code>cloud()</code> , <code>wireframe()</code>
$y \sim x z$	plots y against x for each level of z	<code>xyplot()</code>

Table 8: Trellis Formula Notations



Advanced Graphical Techniques

Motivation

Package lattice

Package lattice

The Panel Argument and Appearance Settings

Package rgl

Package rpanel

Panel Argument and Appearance Settings

- The panel function is a function that uses the subset of arguments to create a display
- the function call, e.g. `histogram()` sets up the external components of the display, such as scale rectangle, axis labels, etc
- the panel function creates everything placed into the plotting region of the graph, such as plotting symbols



Example

```
1 histogram(~x, data = dataset)
2 histogram(~x, data = dataset,
            panel = panel.histogram)
```

```
1 xyplot(y ~ x, data = dataset,
        panel = function(x, y) {
            panel.xyplot(x, y, pch = 20)
        })
5 my.panel = function(x, y) {panel.xyplot(x, y, pch = 29)}
```

In order to temporarily change the default settings of, for instance, plotting symbols, one can rewrite the new value into the panel function inside the general function call.



Conditional and Grouped Plot

```
1 xyplot(Sepal.Length + Sepal.Width ~ Petal.Width +
2   Petal.Length | Species, data = iris,
3   auto.key = list(columns = 2, lines = F,
4   points = T), par.strip.text = list(cex = 0.75)
5 )
6 xyplot(Sepal.Length + Sepal.Width ~ Petal.Width +
7   Petal.Length, groups = Species, data = iris,
8   auto.key = list(columns = 3, lines = F,
9   points = T), par.strip.text = list(cex = 0.75)
10 )
```



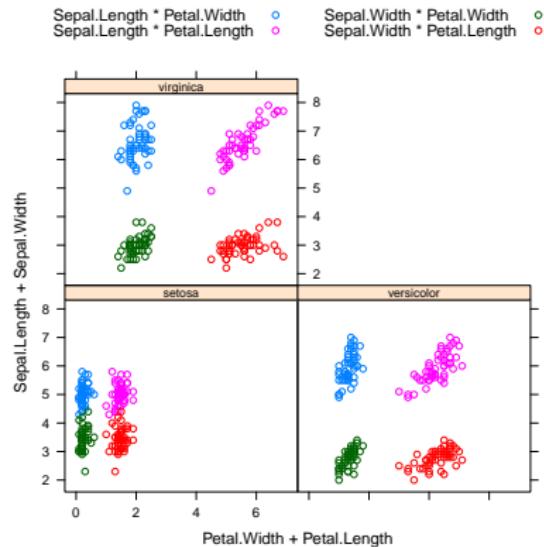
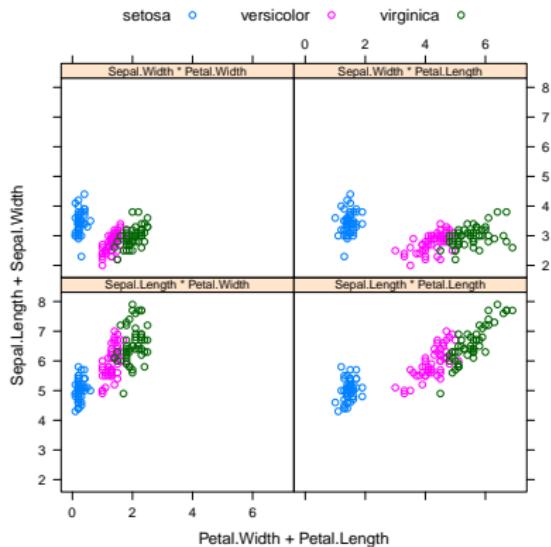


Figure 67: Conditional plots

Figure 68: Grouped plots
BCS_ConditionalGrouped

Conditional and Grouped Density Plot

```
1 densityplot(~weight | feed,
2             data = chickwts,
3             plot.points = FALSE)
```

```
1 densityplot(~weight, data = chickwts,
2             groups = feed,
3             plot.points = FALSE,
4             auto.key = list(columns = 3))
```



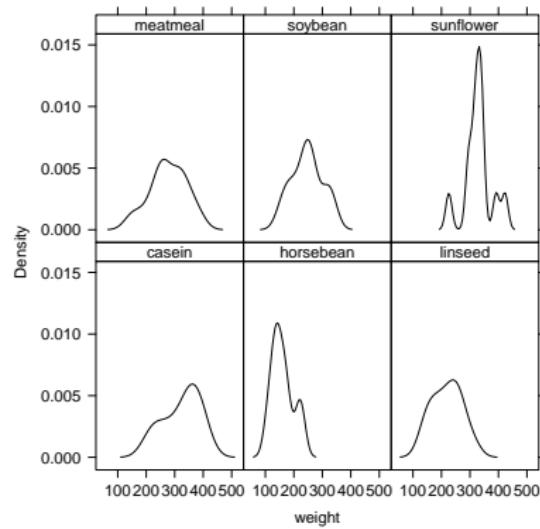


Figure 69: Conditional density plots

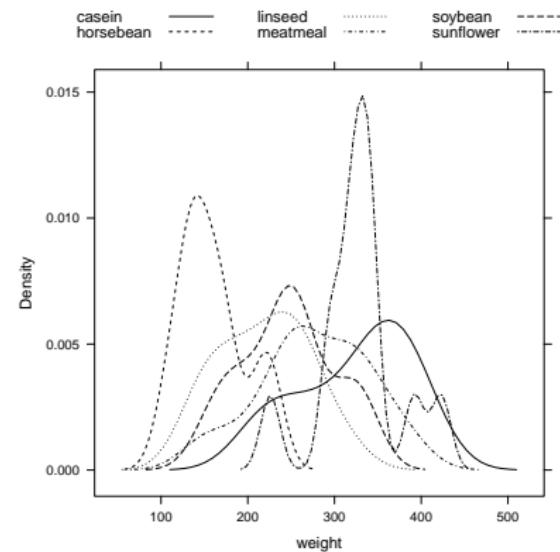


Figure 70: Grouped and overlaid density plots



BCS_ConditionalGroupedDensity



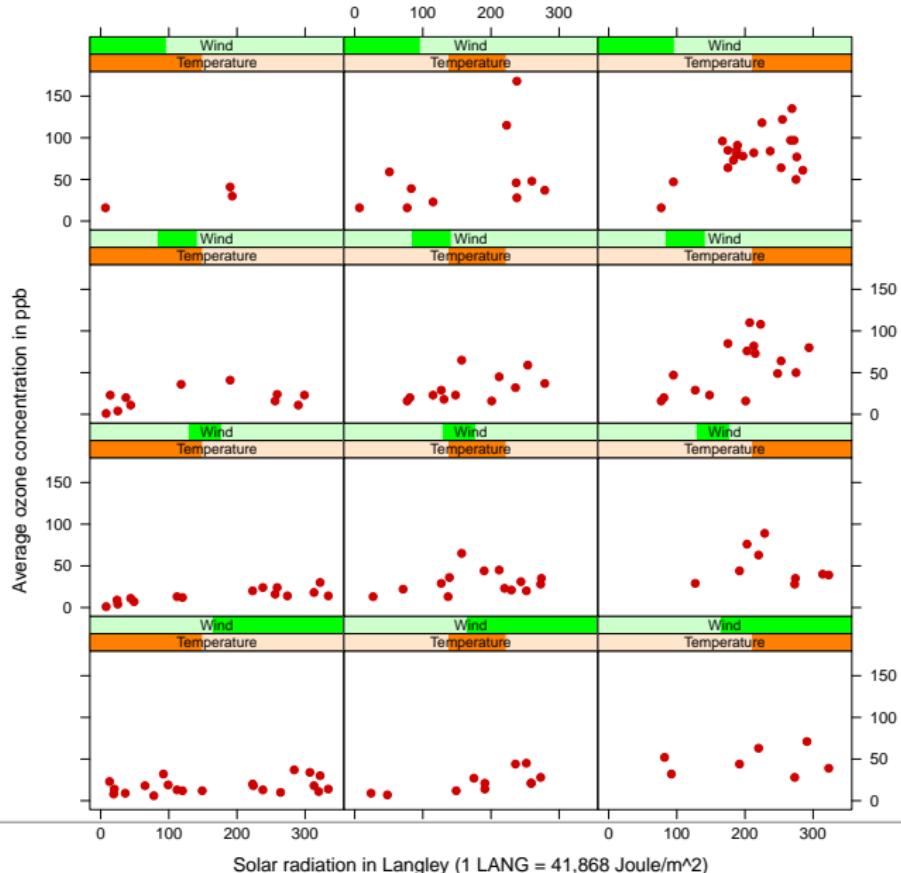
Shingle

lattice enables the usage of continuous (numeric) variables as conditioning, with the shingle concept

```
1 Temperature = equal.count(environmental$temperature,
2                               number = 3, overlap = 0)
3 Wind = equal.count(environmental $ wind,
4                      number = 4, overlap = 0)
5 xyplot(ozone ~ radiation | Temperature * Wind,
6        data = environmental, as.table = T)
```







Time Series Plot

```
1 > xyplot(Nile, aspect = "xy",
2           cut = list(number = 3,
3                     overlap = 0.1),
4           strip = strip.custom(bg = "yellow",
5                     fg = "lightblue"))
```



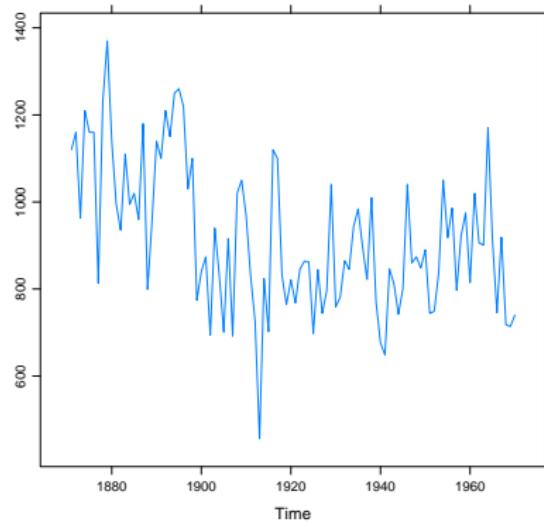


Figure 72: Default time series plot

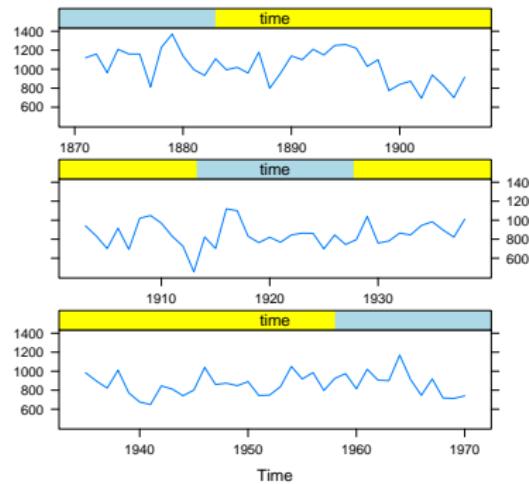


Figure 73: Cut-and-stack time series plot

BCS _ DefaultStackTimeSeries

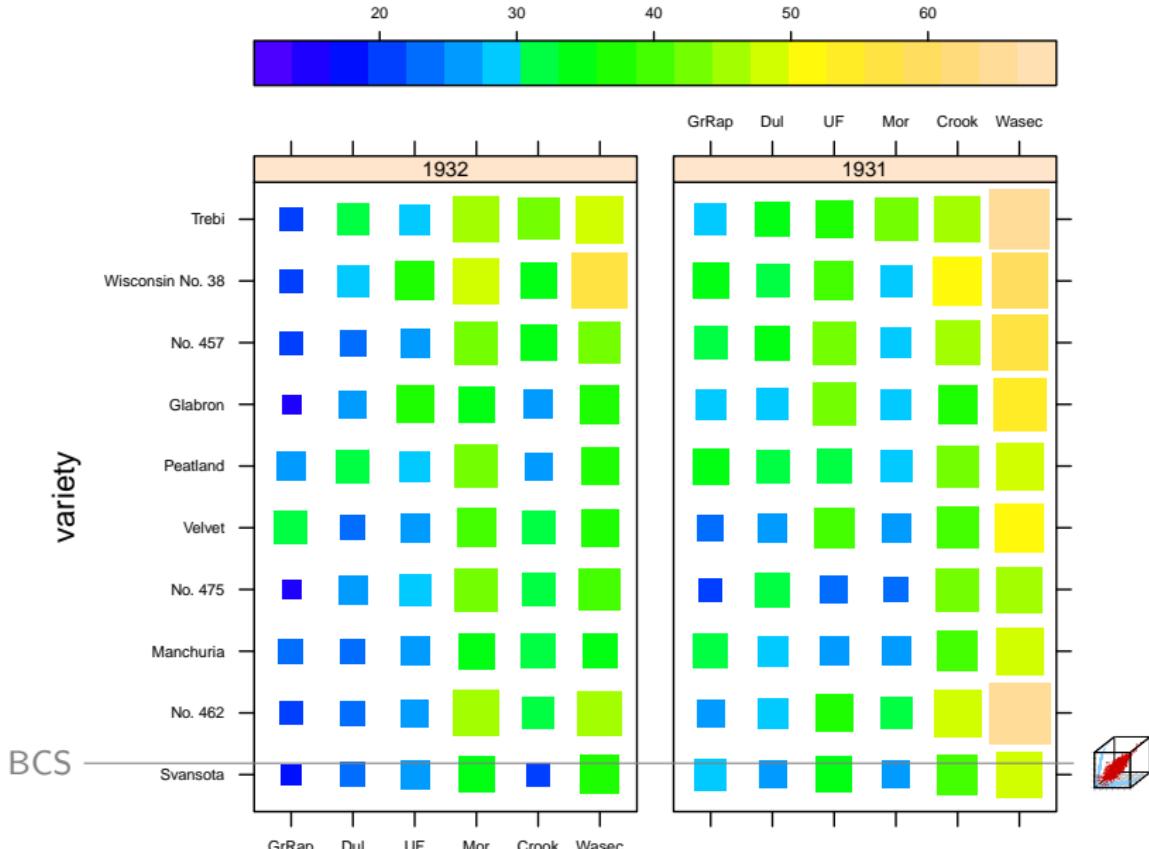


There- and Four-dimension Plot

```
1 levelplot(yield ~ site * variety | year,
2             data = barley,
3             scales = list(alternating = TRUE),
4             shrink = c(0.3, 1),
5             region = TRUE, cuts = 20,
6             col.regions = topo.colors (100),
7             par.settings = list(axis.text = list(cex = 0.5))
8             ,
9             par.strip.text = list(cex = 0.7),
10            between = list(x = 1),
11            aspect = "iso",
12            colorkey = list(space = "top"))
```







Advanced Graphical Techniques

Motivation

Package lattice

Package lattice

The Panel Argument and Appearance Settings

Package rgl

Package rpanel

Package rgl

- RGL is a library of functions that offers three-dimensional real-time visualization functionality with interactive viewpoint navigation to the R programming environment.
- The rgl package includes both low level rgl.*functions and a higher level rgl*3d functions
 - ▶ rgl.* calls set unspecified material properties to default values and permanently changes the material properties with each call
 - ▶ rgl*3d calls use current values as defaults and make temporary changes for the duration of the call



RGL Shape Function

- 3D points are drawn by the function `rgl.points(x,y,z,...)`.
- 3D lines can be depicted with the function `rgl.lines(x,y,z,...)`.
The nodes of the line are defined by the vectors x , y , z , each of length two.
- 3D linestrips are constructed with the function
`rgl.linestrips(x,y,z,...)`. The nodes of the linestrips are like in
`rgl.lines(x,y,z,...)` defined by the vectors x , y , z , each of length two. In the output, each next line strip starts in the point where the previous ends.



- 3D triangles are created with the function `rgl.triangles(x,y,z,...)`. The vectors x, y and z, each of length three, specify the coordinates of the triangle
- 3D quads can be drawn with the function `rgl.quads(x,y,z,...)`. Vectors x, y and z, each of length four, specify the coordinates of the quad.
- 3D spheres are not primitive, but they can be easily created with the function `rgl.spheres(x,y,z,r,...)`. This function plots spheres with centers defined by x, y, z and radius r.
- 3D surfaces can be drawn by the means of the generic `rgl.surface(x,...)` function. This is defined by a matrix specifying the height of the nodes and two vectors defining the coordinates.





Figure 75: Points



Figure 76: Lines



Figure 77: Linestrips



Figure 78: Triangles



Figure 79: Quads
BCS_Shapes

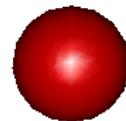


Figure 80: Spheres



3D Surface

```
1 x = z = seq(-9,9)
2 f = function(x,z) (x^2-z^2)/10
3 y = outer(x, z, f)
4 open3d()
5 surface3d(x, z, y,
6             back = "lines",
7             col = rainbow(1000),
8             alpha = 0.9)
9 bbox3d(back = "lines", front = "lines")
```



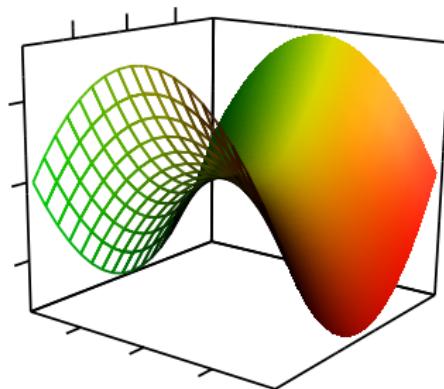


Figure 81: Surface shape

BCS_SurfaceShape



Lighted Plot

```
1 rgl.spheres(rnorm(3), rnorm(3), rnorm(3),
2             r = runif(5),
3             smooth = TRUE)
4 rgl.clear(type = "lights")
5 rgl.light(theta = -90, phi = 50,
6            ambient = "white",
7            diffuse = "#dddddd",
8            specular = "white")
9 rgl.light(theta = 45, phi = 30, ambient = "#dddddd",
10           diffuse = "#dddddd", specular = "white")
```



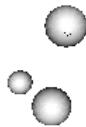


Figure 82: Default light source



Figure 83: Without light



Figure 84: One light source added



Figure 85: Two light sources added

 BCS_LightedPlots

Export Function

To save the screenshot, you can use function `rgl.postscript()`

```
1 rgl.postscript("filename.eps", fmt = "eps", drawText =  
    FALSE)  
2 #eps, tex, pdf, svg, pgf formats are also supported
```

Alternatively,

```
1 rgl.snapshot("filename.png", fmt = "png", top = TRUE)
```



Animation Function

To animate the 3D picture automatically, we can use play3d() and move3d(). Take the 3d surface for example (create it before animation)

```
1 M = par3d("userMatrix")
2 play3d(par3dinterp(userMatrix = list(M,
3 angle = pi,
4 x = 1, y = 1, z = 0)),
5 duration = 5)
```



Interactive Case

```
1 if (interactive()) {  
2   x = rnorm(5);  
3   y = z = x  
4   r = runif(5)  
5   open3d()  
6   spheres3d(x,y,z,r, col = "red3")  
7   k = select3d()  
8   keep = k(x,y,z)  
9   rgl.pop()  
10  spheres3d(x[keep],y[keep],z[keep],r[keep], col = "blue3")  
11  spheres3d(x[!keep],y[!keep],z[!keep],r[!keep], col = "red3"  
12    )  
}
```



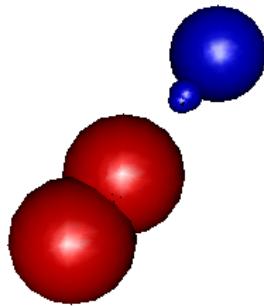


Figure 86: Select a part of the scene

BCS_SceneSelection



Advanced Graphical Techniques

Motivation

Package lattice

Package lattice

The Panel Argument and Appearance Settings

Package rgl

Package rpanel

Package rpanel

- enable the immediate communication with the graphical output and provides dynamic graphics
- enable to interactively choose between several types of plots applied for the same data set.
- offers the possibility to redraw the entire plot, interactively changing the values of the parameters set by the relevant controls



Control Function

Function	Action
rp.doublebutton()	adds a widget with "+" and "-" buttons
rp.checkbox()	adds checkbox to the panel
rp.control()	creates a control panel window
rp.listbox()	adds a listbox to the panel
rp.radiogroup()	adds a set of radiobuttons to the panel
rp.slider()	adds a slider to the panel
rp.tkrplot()	allows R graphics to be drawn in a panel

Table 9: Useful Control Function



Example

```
1 attach(trees)
2 r = diff(range(Height))
3 density.draw = function(panel){
4     plot(density(panel$y, panel$sp),
5           main = "Trees height density estimate")
6     panel }
7 density.panel = rp.control("density estimation",
8                             y = Height, sp = r/8)
9 rp.slider(density.panel, sp, from = r/40, to = r/2,
10           action = density.draw, "Bandwidth")
11 rp.doublebutton(density.panel, sp, 1.03, log = T,
12                  range = c(r/50, NA), title = "Bandwidth",
13                  action = density.draw)
```



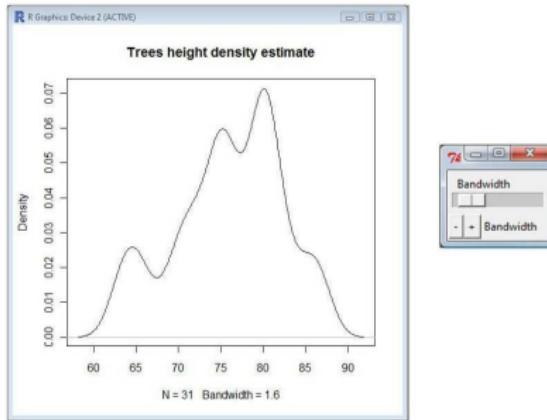


Figure 87: Slider and double button for the control of density estimate

 [BCS_ControlDensityEstimate](#)



Example

```
1 data.plotfn <- function(panel){  
2   if (panel$plot.type == "histogram")  
3     hist(panel$y)  
4   else if (panel$plot.type == "boxplot")  
5     boxplot(panel$y)  
6   panel  
7 }  
8 panel = rp.control(y = Height)  
9 rp.listbox(panel, plot.type,  
10             c("histogram", "boxplot"),  
11             action = data.plotfn,  
12             title = "Plot type")
```



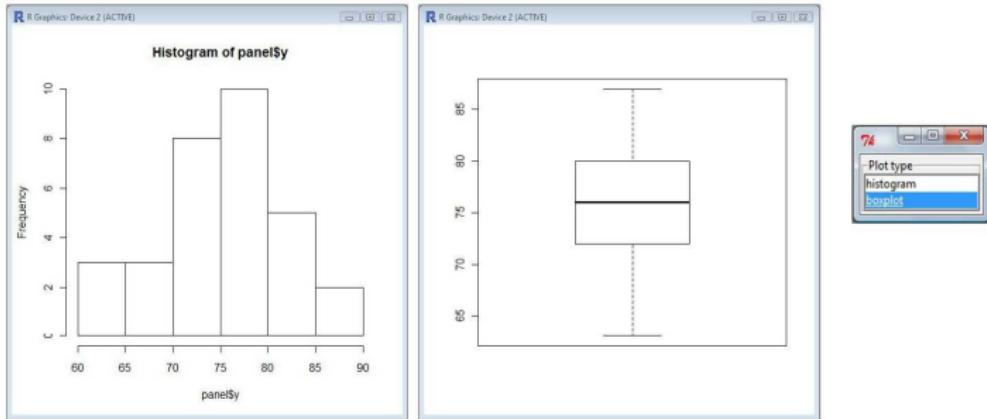


Figure 88: Listbox control function with histogram and boxplot as alternative plots



Application Function in rpanel

Function	Action
<code>rp.ancova()</code>	plots a response variable against a covariate
<code>rp.logistic()</code>	plots a binary response variable against a covariate
<code>rp.plot3d()</code>	plots a 3D scatterplot, using the <code>rgl</code> package
<code>rp.regression()</code>	plots a response variable against one or two covariates
<code>rp.normal()</code>	plots a histogram and adds normal and fitted distributions to it

Table 10: Application Functions



Application Function in rpanel

```
1 if (interactive()) {  
2   data(longley)  
3   attach(longley)  
4   rp.regression(GNP, Unemployed, line.showing = T,  
5                 panel.plot = F)}
```

```
1 if (interactive()) {  
2   data(longley)  
3   attach(longley)  
4   rp.regression(cbind(GNP, Armed.Forces), Unemployed)}
```

```
1 if (interactive()) {  
2   y = Height  
3   rp.normal(y, panel.plot = T)  
4 }
```



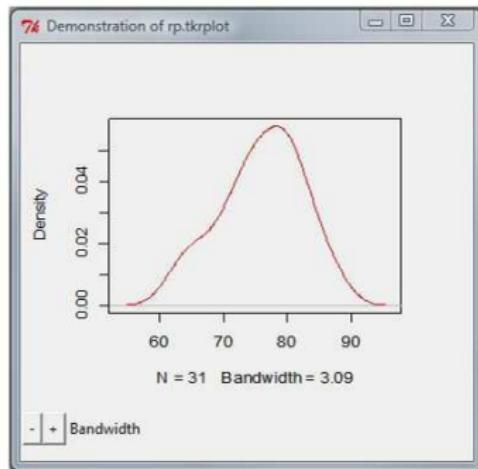


Figure 89: Density plot with `rp.tkrplot`

 BCS_rp.tkrplot



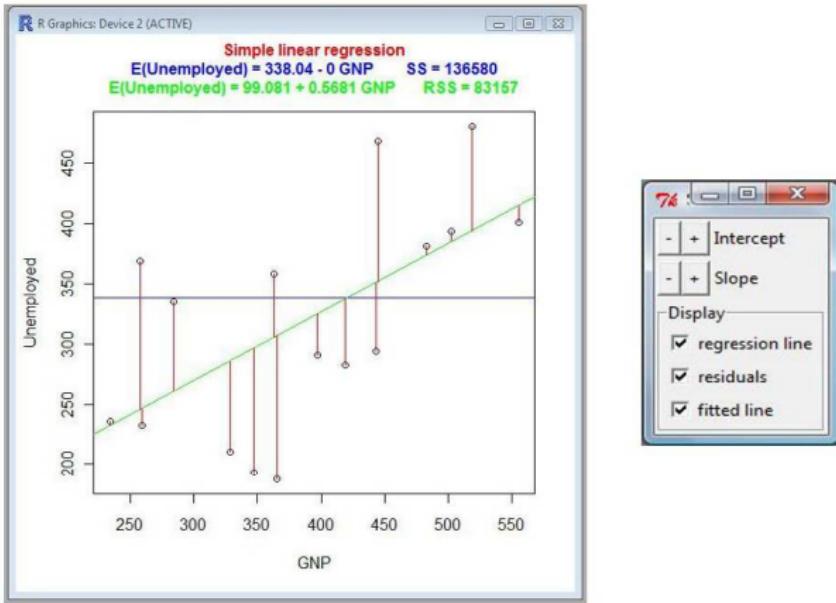


Figure 90: Regression with one covariate



BCS _UnivariateRegression



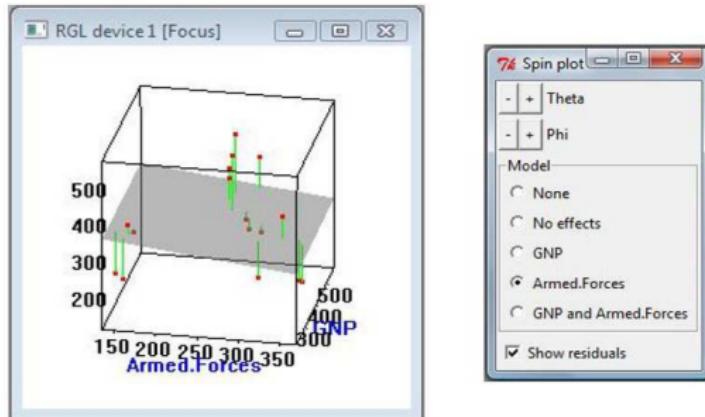


Figure 91: Regression with two covariates

BCS_BivariateRegression

