Introduction to R

Marlene Müller

November 18, 2008

Fraunhofer Institut Techno- und Wirtschaftsmathematik
ITWM

Contents

1 What is this R? 2

2 How do I start? 3

3 How to get help? 6

4 Some calculations to start with 10

5 Data & files 20

6 Wonderful world of graphics 27

7 Some statistics 41

8 "Advanced" mathematics 53

9 Basics in programming 59

1 What is this R?

Programming Language S = developed at Bell Labs for statistics, simulation, graphics (Becker and Chambers; 1984)

→ S-PLUS: commerical implementation

→ R: implementation under GPL (GNU General Public License), open source

- + interpreted program code, object orientation
- + easily extensible by self-written routines, packages, DLLs
- + many types of graphics (mainly static)
- + standardized, simple-to-used data format (`data.frame`)
- + well developed format fo fitting (regression) models
- + active developers team, helpful mailing list
- – (up to now) no "standard" GUI
- – available routines/packages sometimes difficult to find
- – books on R appearing slowly on the market (S books partly useable)

2 How do I start?

R is command-line oriented, so start simply by typing expressions like

```
> 1+1
[1] 2

> 1+2*3^4
[1] 163

> x <- 1; y <- 2
> x+y
[1] 3

> x <- seq(-pi,pi,by=0.1)
> plot(x,sin(x),type="l",col="red",main="Sinuskurve")
```

2.1 Working with R under Unix/Linux

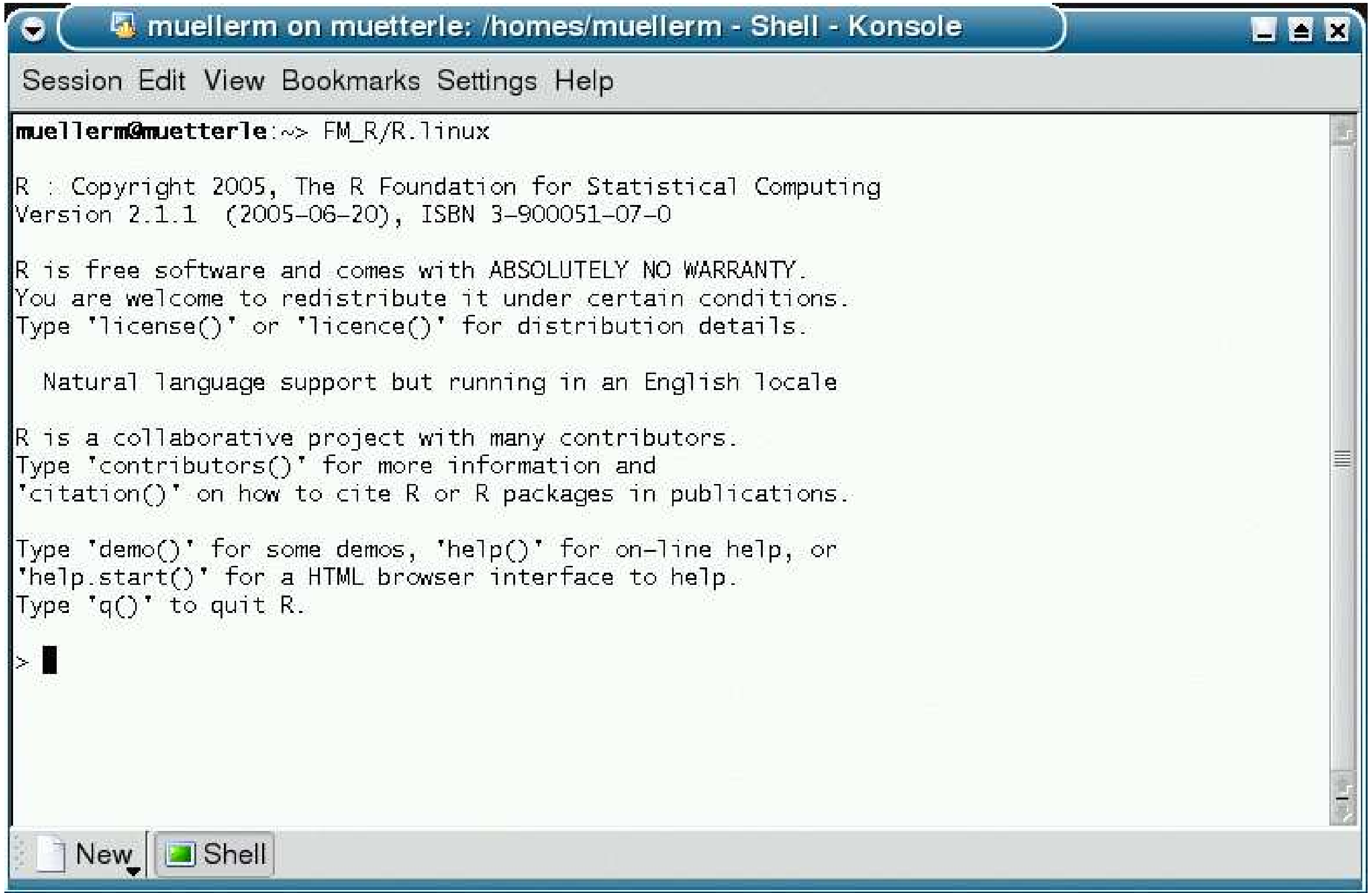


Fig. 1: R in a Unix/Linux shell

2.2 Working with R under Windows

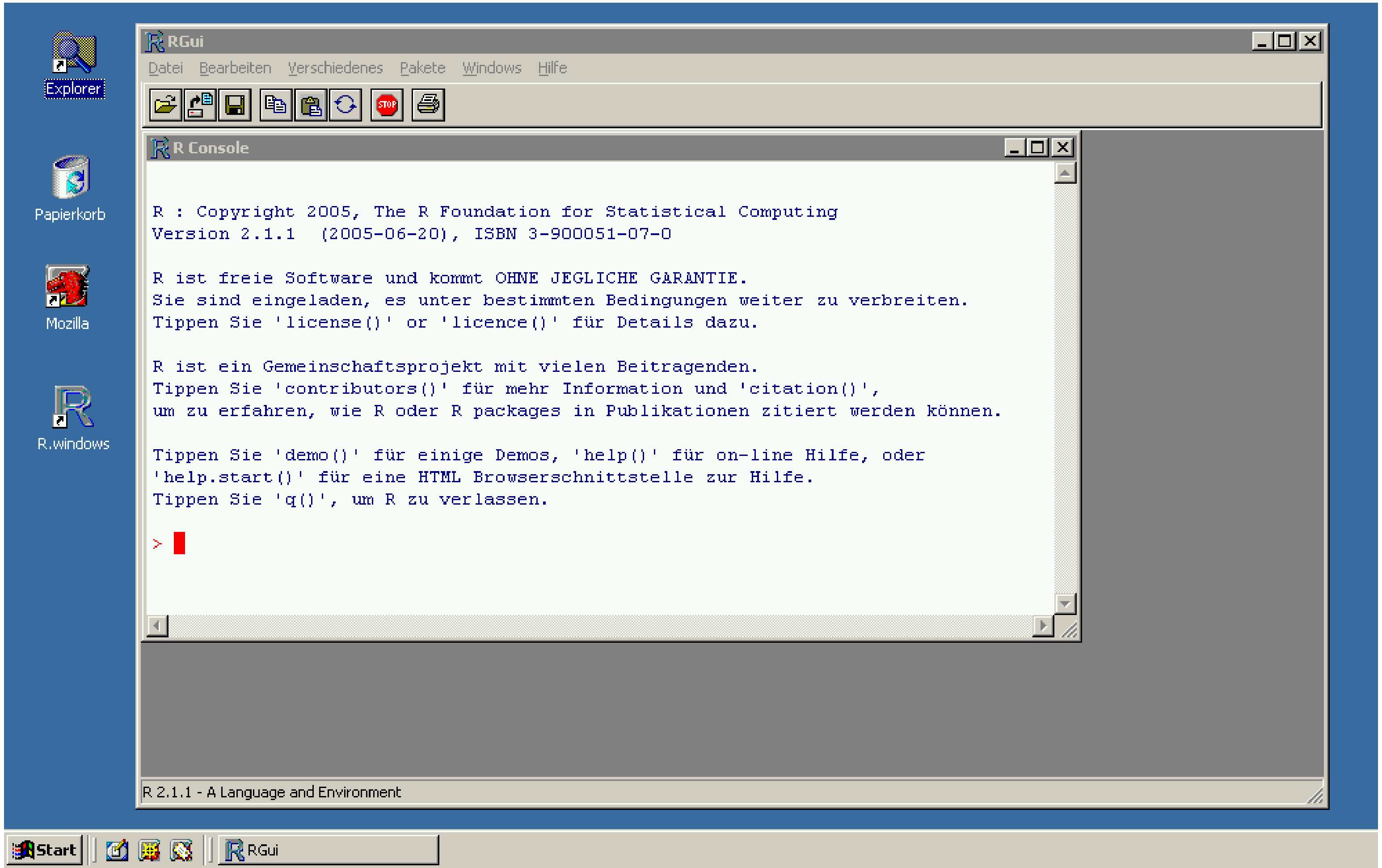


Fig. 2: R in a Windows desktop

3 How to get help?

3.1 Local help pages

- help for a function:
  `help(<function name>)` or `?<function name>`
- help for a package:
  `library(help=<Package>)`

Usually, the texts in the local help pages correspond to those in the package documentation.

3.2 WWW

- **http://www.r-project.org**
  R home page, there are in particular FAQs as well as a Google site search, and additionally:
  - manuals (http://cran.r-project.org/manuals.html)
    introduction, language definition, "Writing R Extensions" (DLLs, packages), introduction written in different languages (German, French, etc.)
  - CRAN (http://cran.r-project.org)
    Comprehensive R Archive Network (→ R software for download)
  - mailing lists (→ Section 3.3)
  - books list (→ Section 3.4)
  - related projects

3.3 Mailing lists

- R-help
  main list for R user questions, take care to read
  http://www.r-project.org/posting-guide.html before!

  → also available as a (usenet-) news group gmane.comp.lang.r.general auf http://news.gmane.org

- R-announce, R-packages, R-devel
  announcements, package announcements, developers list (→ more for R specialists)

- R-sig-* (special interests groups)
  e.g. R-sig-finance = Special Interest Group for 'R in Finance'

For subscribing and archives see http://www.r-project.org/mail.html or http://news.gmane.org/index.php?prefix=gmane.comp.lang.r.

Helpful for search is http://www.rseek.org.

3.4 Books

- Dalgaard (2002): Introductory Statistics with R
- Murrell (2005): R Graphics
- Ligges (2005): Programmieren mit R
  (see also: http://www.statistik.uni-dortmund.de/∼ligges/PmitR/)
- Venables and Ripley (2002): Modern Applied Statistics with S
  (R complements: http://www.stats.ox.ac.uk/pub/MASS3)
- Venables and Ripley (2000): S Programming
  (see also: http://www.stats.ox.ac.uk/pub/MASS3/Sprog)

→ further: http://www.r-project.org/doc/bib/R-books.html

4 Some calculations to start with

Demos:

```
demo()
demo(graphics)   # nice graphics ;-)
demo(persp)      # nice 3D graphics ;-)
demo(image)      # more nice graphics ;-)
```

Assigning values:

```
x <- 1
x <- 0 -> y
x <- y <- z <- NA                  # missing
x <- 0/0                           # not a number (NaN)
x <- NULL                          # no value

x <- rnorm(100)                          # 100 N(0,1) random variables
hist(x, col="orange")                    # histogram
r <- hist(x, col="orange", freq=FALSE)   # same histogram?
g <- seq(-5,5,length=100)
ylim <- range(c(r$density,max(dnorm(g))))
hist(x, col="orange", freq=FALSE, ylim=ylim)   # again the same histogram?
lines(g, dnorm(g))                             # with N(0,1) pdf
```

Useful tools:

```
ls()                   ## list all R objects

x <- 1:3
x                      ## show object (here vector: x)

print(x)               ## show object (here vector: x), also within
                       ## R scripts and functions

fun <- function(x){ sin(x) }
fun                    ## show object (here function: fun)

median                 ## show internal object (here function: median)

memory.limit(1536)  ## ONLY Windows: increase memory limit to 1.5GB

rm(x)                  ## delete object x

save.image()           ## save workspace (.RData, .Rhistory)
load(".RData")         ## load workspace (.RData, .Rhistory)

date()                 ## date and time

q()                    ## quit R
```

4.1 Data types

Numeric:

```
x <- 1
y <- pi     # predefined pi = 3.1415926535898
```

Character:

```
x <- "a"
y <- "a text"
```

Logical:

```
x <- TRUE
y <- 1 > 2

> y
[1] FALSE
```

More complex data types can be constructed by combining these three simple types into vectors, matrices, arrays and lists.

4.2 Vectors, matrices, arrays, ...

Vectors:

```
x <- c(1,2,3)
x <- 1:3

y <- c(1,1,1)
y <- rep(2,10)

z <- as.character(1:3)
z <- c("a","b","c")

length(z)

names(x) <- z

x[2:3]
x["b"]
```

All elements of a vector are of the same type (numeric, character, logical)!

Matrices:

```
x <- 1:20
x <- matrix(x, 5,4)    # matrix(x, nrow=5,ncol=4)

x[2,3]
x[c(1,5),2:4]
x[,2:4]

dim(x)
nrow(x)
ncol(x)

length(x)
as.vector(x)

dimnames(x) <- list(paste("row",1:nrow(x), sep=""),c("a","b","c","d"))

x[,"b"]
x[,c("a","b")]
```

All elements of a matrix are of the same type (numeric, character, logical)!

Vectors from vectors:

```
x <- c(2,6,3)
y <- 1:3

c(x,y)                # concatenate two vectors
c(x,1:5,y,6)          # concatenate vectors and scalars
```

Matrices from vectors:

```
x <- c(2,6,3)
y <- 1:3

cbind(x,y)            # vertical concatenation
rbind(x,y)            # horizontal concatenation

cbind(x,y,rep(0,3))   # vertical concatenation
```

Arrays:

```
x <- 1:60
x <- array(x, c(5,4,3))

x[2,3,1]
x[1,2:4,3]
x[,,1]

dim(x)
nrow(x)
ncol(x)

length(x)
as.vector(x)

dimnames(x) <- list(paste("row",1:nrow(x), sep=""),c("a","b","c","d"),c("x","y","z"))
```

All elements of an array are of the same type (numeric, character, logical)!

Lists:

```
x <- list(One=11:15, Two=c("a","b","c"), Three=(1:4)>0)
y <- list(x=x, Four=1:3)

x$One
y$x$One

y$Four
y[[2]]

length(x)
length(y)

y$Five <- names(x)
```

Lists may contain objects of different type, these objects can be called with `$<name>` by name or with `[[<number>]]` by their number.

Data frames:

```
x <- data.frame(N=11:14, C=c("a","b","c","d"), L=(1:4)>0)

dim(x)
nrow(x)
ncol(x)

length(x)
as.vector(x)

names(x)

x[2,3]
x[,2:3]

x[,2]
x[,"C"]
x$C
```

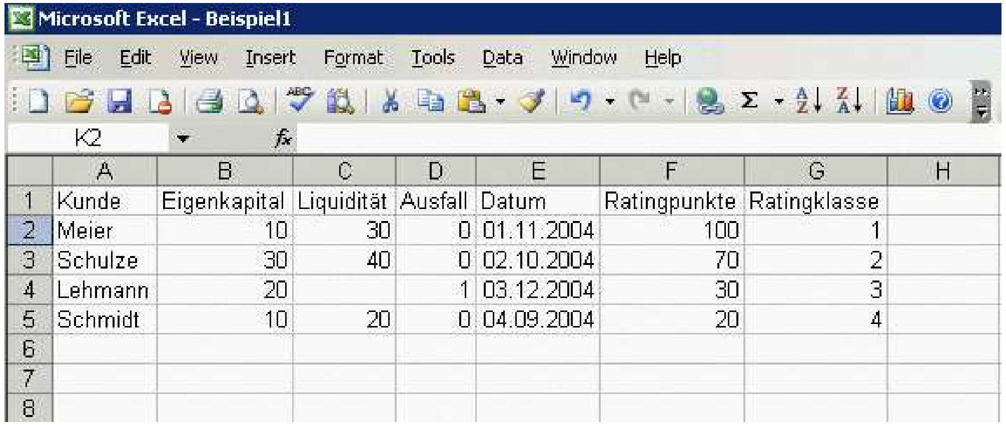
Data frames are lists, in which al columns have the same length. → Excel tables, save as .csv, are typically read as a data frame into R (`data.frame`).

4.3 Operations (elementwise and/or vector-/matrixwise)

```
x <- matrix( 1:20, 5, 4)   # 5x4 Matrix

x+1; x-1; x*1; x/1         # elementwise operations
sin(x); exp(x)             # elementwise function calls

y <- 1:5
x * y                      # elementwise multiplication

z <- 1:4
x %*% z                    # matrix multiplication

min(x)                     # minimum of all elements of x
apply(x,1,min)             # row minima
apply(x,2,min)             # column minima

y <- c(TRUE, TRUE, FALSE, FALSE)
y & TRUE                   # elementwise logical operation (‘‘AND’’)
y | FALSE                  # elementwise logical operation (‘‘OR’’)
!y                         # elementwise logical operation (‘‘NOT’’)

y && TRUE                  # BUT: here only the first result holds! (‘‘AND’’)
y || FALSE                 # BUT: here only the first result holds! (‘‘OR’’)
```

5 Data & files

Example file in Excel:



	A	B	C	D	E	F	G	H
1	Kunde	Eigenkapital	Liquidität	Ausfall	Datum	Ratingpunkte	Ratingklasse	
2	Meier	10	30	0	01.11.2004	100	1	
3	Schulze	30	40	0	02.10.2004	70	2	
4	Lehmann	20		1	03.12.2004	30	3	
5	Schmidt	10	20	0	04.09.2004	20	4	
6								
7								
8								

→ save under Excel as CSV: Example1.csv

```
Kunde;Eigenkapital;Liquidität;Ausfall;Datum;Ratingpunkte;Ratingklasse
Meier;10;30;0;01.11.2004;100;1
Schulze;30;40;0;02.10.2004;70;2
Lehmann;20;;1;03.12.2004;30;3
Schmidt;10;20;0;04.09.2004;20;4
```

5.1 Reading and saving CSV files

Reading the file Example1.csv:

```
x <- read.csv("Example1.csv", sep=";")

dim(x)
names(x)
x
```

```
    Kunde Eigenkapital Liquidität Ausfall      Datum Ratingpunkte Ratingklasse
1   Meier           10         30       0 01.11.2004          100            1
2 Schulze           30         40       0 02.10.2004           70            2
3 Lehmann           20         NA       1 03.12.2004           30            3
4 Schmidt           10         20       0 04.09.2004           20            4
```

Saving the data to Example2.csv:

```
write.table(x,file="Example2.csv",sep=";",row.names=FALSE,quote=FALSE)
```

More functions for reading data:

- `read.table` (ASCII data)
- `scan` (scans any text file, further postprocessing necessary)

Functions to convert data:

- `as.numeric`, `as.character`, `as.factor`

Other possibilities to communicate data (not testet ;-)):

- `RODBC` (accessing data from databases)
- R-Excel-interface via DCOM server (http://cran.at.r-project.org/contrib/extra/dcom)

5.2 Random numbers and probability distributions

Examples for normal distribution:

`rnorm(n, mean=0, sd=1)`	pseudo-random numbers
`dnorm(x, mean=0, sd=1)`	density (pdf)
`pnorm(x, mean=0, sd=1)`	cumulative distribution function (cdf)
`qnorm(p, mean=0, sd=1)`	quantiles

In the same manner:

Uniform distribution	`{r|d|p|q}unif`	t distribution	`{r|d|p|q}t`
Lognormal distribution	`{r|d|p|q}lnorm`	Gamma distribution	`{r|d|p|q}gamma`
χ^2 distribution	`{r|d|p|q}chisq`	Beta distribution	`{r|d|p|q}beta`
Binomial distribution	`{r|d|p|q}binom`	Poisson distribution	`{r|d|p|q}pois`

...

$\longrightarrow$ it is possible to fix the seed by `set.seed`

5.3 R script files

Run a script with R code:

```
> source("MyProgram.R")
```

Saving R output to file:

```
sink("MyOutput.txt")   # from now all output goes to file
sink()                 # and now to the screen again
```

Normal- vs. t distribution:

```
x <- rnorm(100)
mean(x)
sd(x)

plot(rnorm(10000), rnorm(10000))

x <- seq(-5,5,by=0.1)
plot(x, dnorm(x), type="l", col="black", lwd=2)
lines(x, dt(x, df=1), col="blue")
lines(x, dt(x, df=5), col="orange")
lines(x, dt(x, df=20), col="red")

qnorm(0.95)
qnorm(0.975)
```

Multivariate normal distribution:

```
library(help=mvtnorm)
library(mvtnorm)

mu <- c(0,0)      # means
sigma <- c(1,1)   # std.dev.
rho <- 0.5        # correlation

S <- matrix(NA, 2,2)
diag(S) <- sigma^2
S[1,2] <- S[2,1] <- rho*prod(sigma)

x <- rmvnorm(n=10000, mean=mu, sigma=S)
plot(x)

x <- seq(-5*sigma[1]+mu[1], 5*sigma[1]+mu[1], length = 50)
y <- seq(-5*sigma[2]+mu[2], 5*sigma[2]+mu[2], length = 50)
f <- function(x,y) { dmvnorm(cbind(x,y), mean=mu, sigma=S) }
z <- outer(x, y, f)
persp(x, y, z, theta = 10, phi = 20, expand = 0.5, col = "lightblue", shade = 0.75)
```

6 Wonderful world of graphics

Credit scoring data:

```
file <- read.csv("kredit.csv",sep=";")
y <- 1-file$kredit        # default set to 1
prev    <- (file$moral >2)+0                                   # previous loans were OK
employ  <- (file$beszeit >1)+0                                 # employed (>=1 year)
dura    <- (file$laufzeit)                                     # duration
d9.12   <- ((file$laufzeit >9)&(file$laufzeit <=12)) +0 #  9 < duration <= 12
d12.18  <- ((file$laufzeit >12)&(file$laufzeit <=18))+0 # 12 < duration <= 18
d18.24  <- ((file$laufzeit >18)&(file$laufzeit <=24))+0 # 18 < duration <= 24
d24     <- (file$laufzeit >24)+0                               # 24 < duration
amount  <- file$hoehe                                          # amount of loan
age     <- file$alter                                          # age of applicant
savings <- (file$sparkont > 4)+0                               # savings >= 1000 DM
phone   <- (file$telef==1)+0                                   # applicant has telephone
foreign <- (file$gastarb==1)+0                                 # non-german citizen
purpose <- ((file$verw==1)|(file$verw==2))+0                   # loan is for a car
house   <- (file$verm==4)+0                                    # house owner
```

6.1 Barplots

→ graphical representation of the frequency distribution for discrete variables

```
table(dura)                                  ## frequency table

barplot(table(dura), col="cyan", main="Duration of Loan")
                                             ## absolute frequencies

barplot(table(dura)/length(dura), col="cyan", main="Duration of Loan")
                                             ## relative frequencies

par(mfrow=c(1,3))   # graphical display with 1 row, 3 columns
barplot(table(dura),    col="cyan",    main="Duration of Loan")
barplot(table(savings), col="orange",  main="Savings >1000 DM")
barplot(table(house),   col="magenta", main="House Owner")
par(mfrow=c(1,1))   # reset display to single plot
```

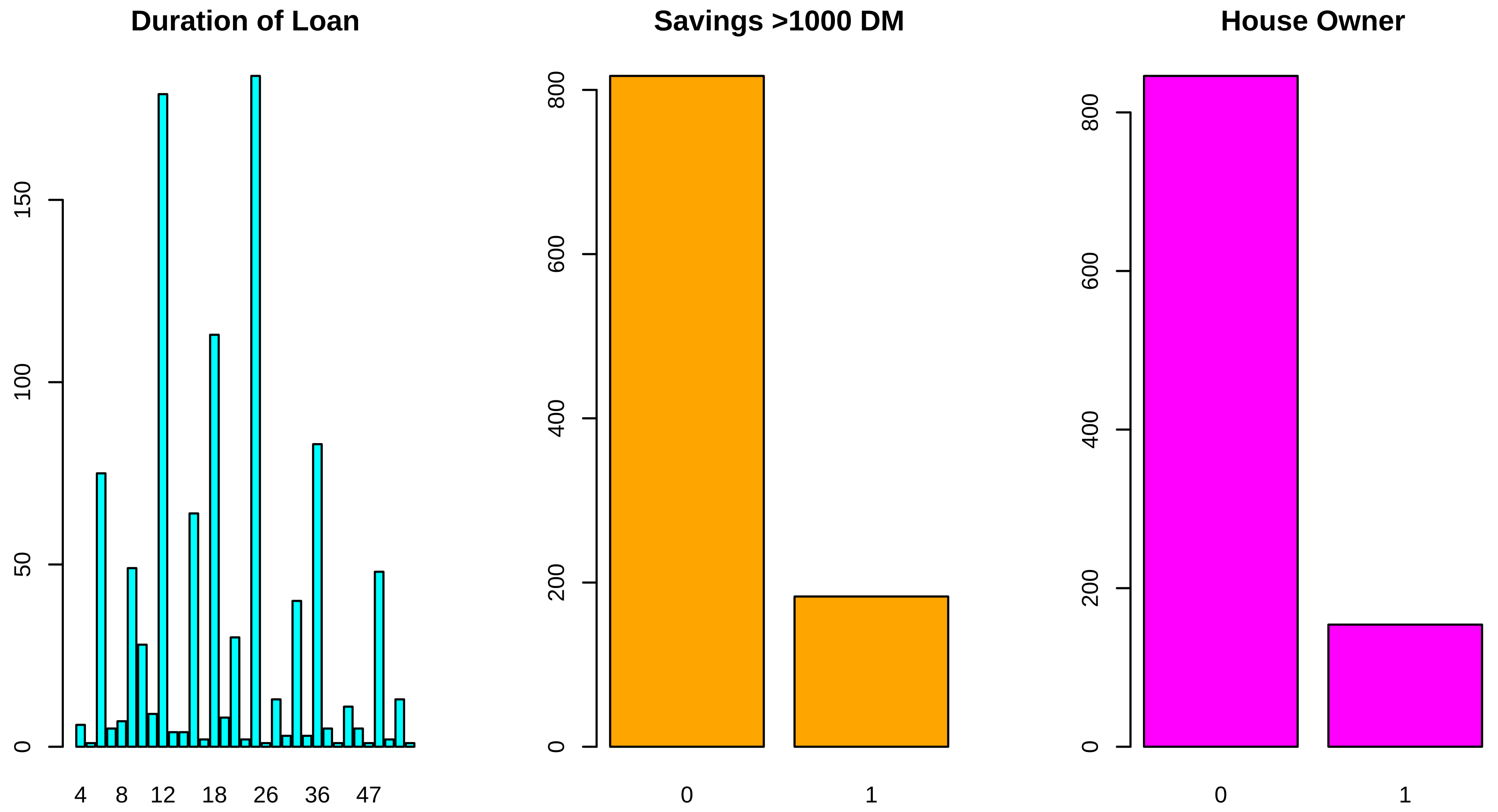


Fig. 3: Examples for bar plots: duration of loan (left), savings (center) and house-owner indicator (right)

6.2 Boxplots

→ graphical representation of outliers, minima/maxima, 25%-,50%-,75%-quantiles

```
boxplot(age)
boxplot(age, horizontal=TRUE)

boxplot(age, col="gray",horizontal=TRUE)

boxplot(age ~ y, col=c("gray","red"),horizontal=TRUE, main="Age vs. Y")
boxplot(amount ~ y, col=c("gray","red"),horizontal=TRUE, main="Amount vs. Y")
```

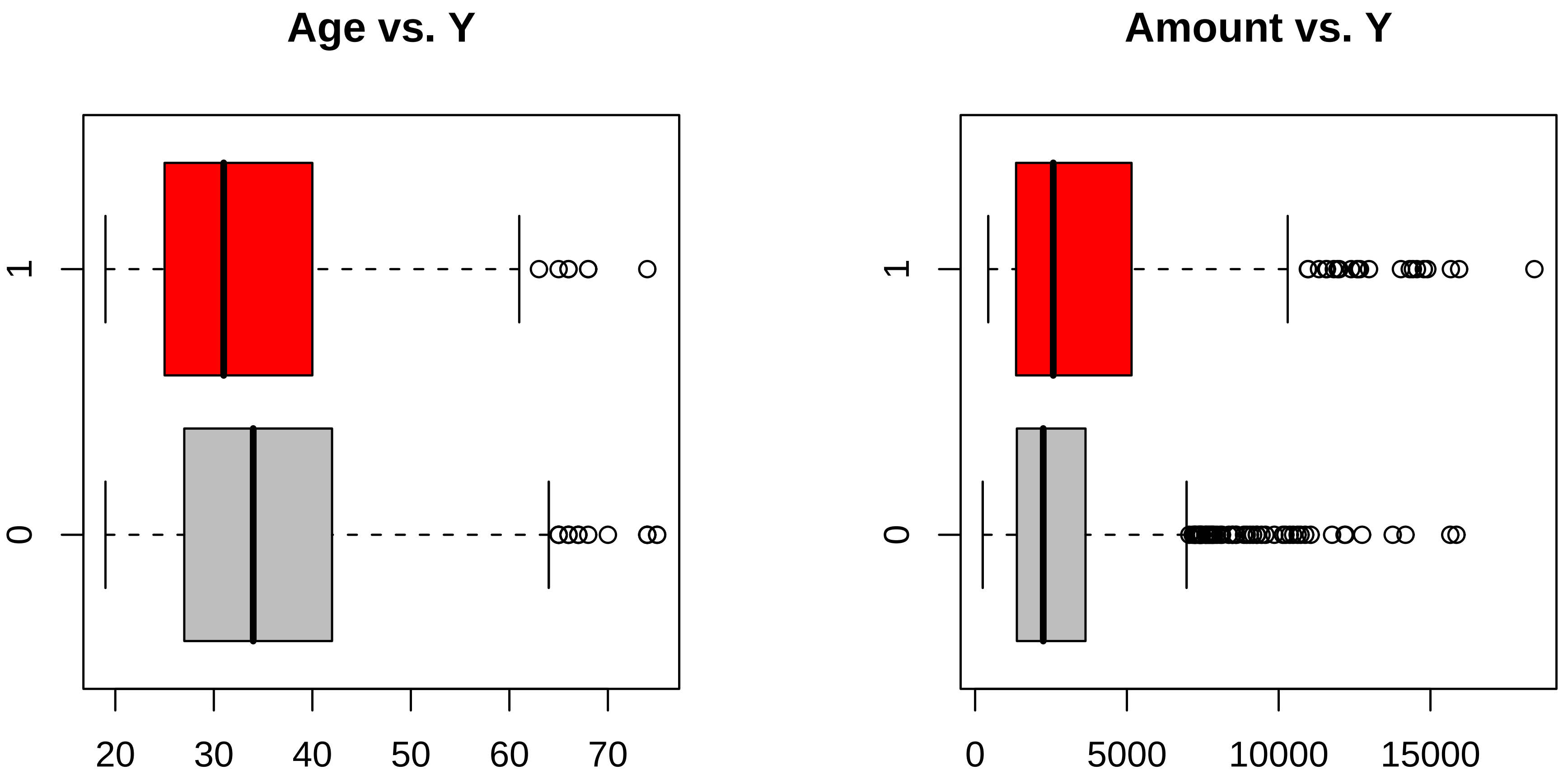


Fig. 4: Age of credit applicant (left) and amount of loan (right) vs. default indicator (1 = default, 0 = non-default)

6.3 Histograms

→ graphical representation of the distribution (pdf) of continuous variables

```
hist(age)
hist(age, freq=FALSE)
hist(age, freq=FALSE, col="gray")

hist(amount, freq=FALSE, col="gray",    main="Amount")
xx <- seq(min(amount),max(amount), length=100)
lines(xx, dnorm(xx, mean(amount), sd(amount)), col="red")
lines(xx, dlnorm(xx, mean(log(amount)), sd(log(amount))), col="green", lwd=2)

## smaller intervals and better vertcal scale
b  <- seq(0,20000,by=1500)                                ## new intervals
h  <- hist(amount, freq=FALSE, breaks=b, plot=FALSE)   ## histogram without display
xx <- seq(min(amount),max(amount), length=100)
d1  <- dnorm(xx, mean(amount), sd(amount))             ## normal pdf
d2  <- dlnorm(xx, mean(log(amount)), sd(log(amount)))  ## lognormal pdf
ylim <- range( c(h$density, d1, d2) )

hist(amount, freq=FALSE, breaks=b, col="gray", main="Amount", ylim=ylim)
lines(xx, d1, col="red")
lines(xx, d2, col="green", lwd=2)
```

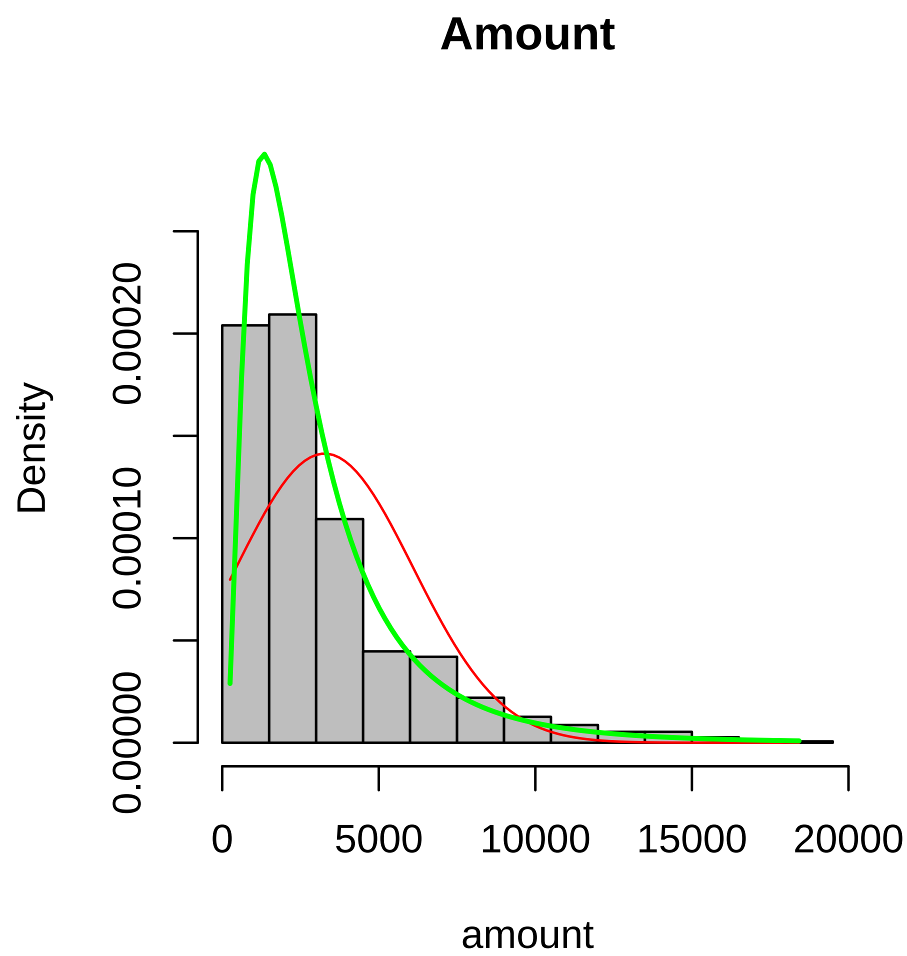


Fig. 5: Probability distribution of the amount of the loan, histogram in comparison with normal and lognormal pdfs

6.4 Scatterplots and curves

→ point clouds ...

```
plot(age, amount)

color <- 1*(y==1) + 2*(y==0)
plot(age, amount, col=color)

color <- rep("", length(age))
color[y==1] <- "red"
color[y==0] <- "blue"
plot(age, amount, col=color)

plot(1:20,1:20,col=1:20, pch=1:20)
text(1:20,1:20,labels=as.character(1:20), pos=4)

symbol <- 8*(y==1) + 1*(y==0)
plot(age, amount, col=color, pch=symbol)
```

→ ... or curves or both of them

```
x <- seq(-pi,pi,length=100)
plot(x, sin(x), type="l")
lines(x, cos(x), col="red")

logit <- glm(y ~ age, family=binomial(link = "logit"))

plot(age, logit$fitted.values)

plot(age, logit$fitted.values, type="l")         ## not this way ...

o <- order(age)
plot(age[o], logit$fitted.values[o], type="l")   ## ... but that way! (sorted data)

plot(age[o], logit$fitted.values[o], type="l", lwd=2, ylim=c(0,1))
title("PDs")
points(age, y, col="red", pch=3, cex=0.5)
```

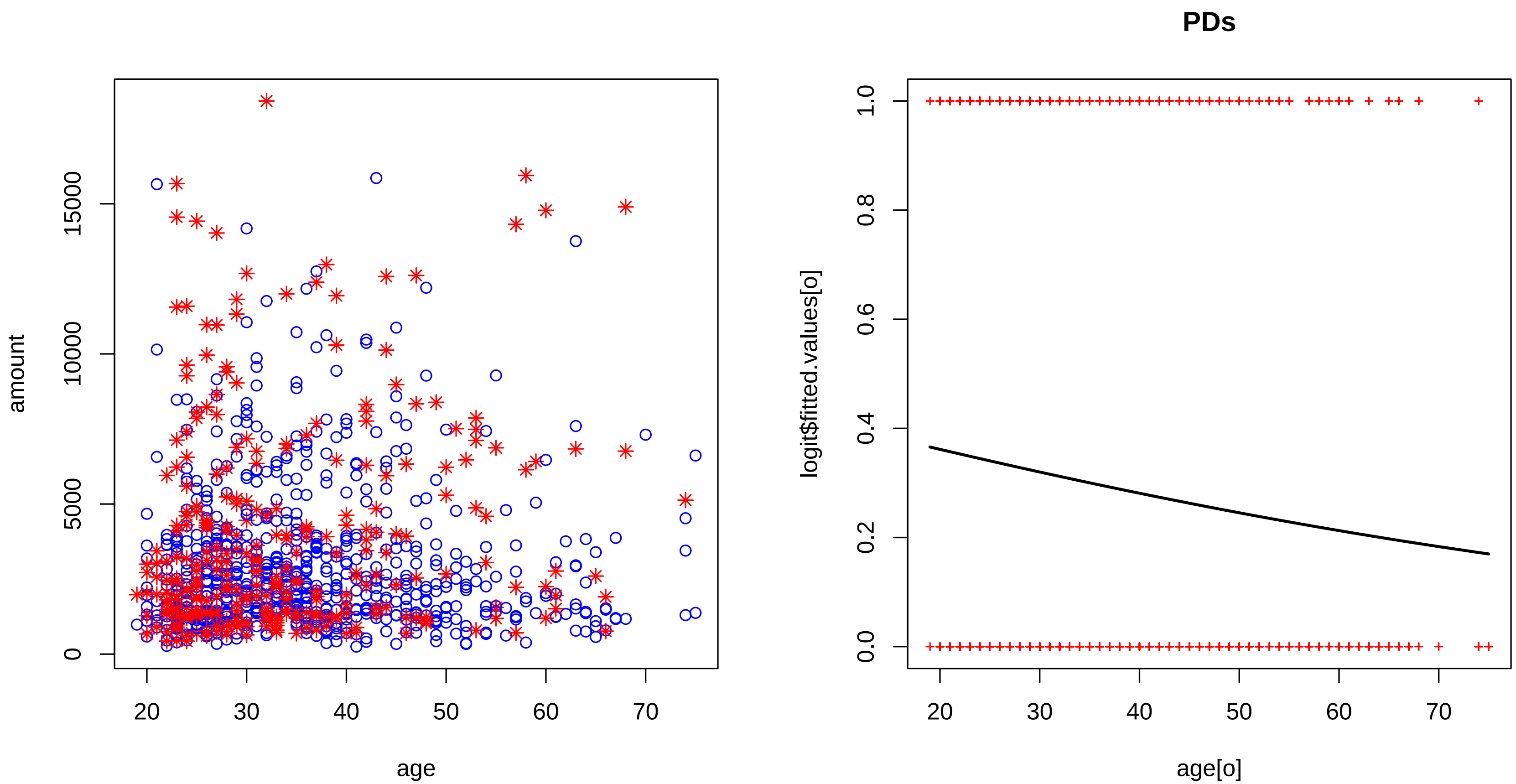


Fig. 6: Scatterplot of age vs. amount (left), logit PDs (right)

6.5 Three-dimensional graphics

→ surfaces, point clouds, contours

```
## bivariate normal pdf
library(mvtnorm)
x <- y <- seq(-5, 5, length = 50)
f <- function(x,y) { dmvnorm(cbind(x,y)) }
z <- outer(x, y, f)
persp(x, y, z, theta = 10, phi = 20, expand = 0.5, col = "lightblue")
persp(x, y, z, theta = 10, phi = 20, expand = 0.5, col = "lightblue", shade = 0.75)

## contours of the bivariate normal pdf
x <- y <- seq(-5, 5, length = 150)
z <- outer(x, y, f)
contour(x, y, z, nlevels=20)
contour(x, y, z, nlevels=20, col=rainbow(20))
contour(x, y, z, nlevels=20, col=rainbow(20), labels="")

## 3-dimensional normal data
library(scatterplot3d)
x <- matrix(rnorm(15000),ncol=3)
scatterplot3d(x)
scatterplot3d(x, angle=20)
```

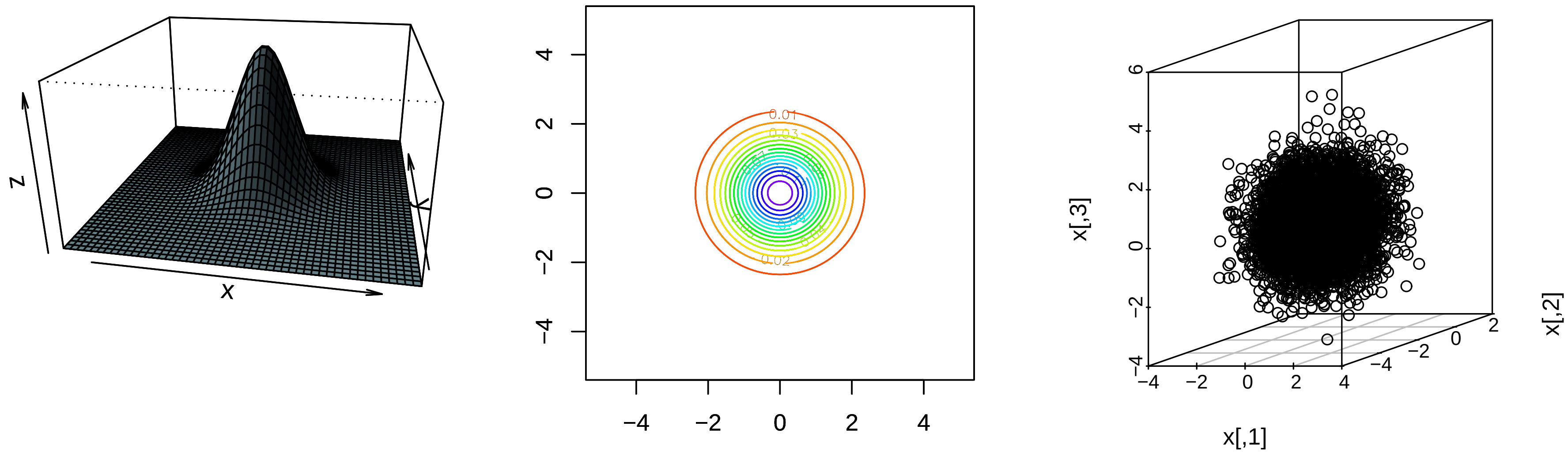


Fig. 7: Bivariate normal pdf: 3D plot of the pdf (left), contour curves (center); 3D scatterplot (right)

6.6 Arranging plots

Directly within the plot routines: → obtain help by `?par`

- set colors with `col=...` (generate colors by → `?rainbow`, `?rgb`, `?col2rgb`)
- set symbol style with `pch=...`, symbol size with `cex=...`
- set plot title with `main=...`, axes lables with `xlab=...`, `ylab=...`
- set plot drawing limits with `xlim=...`, `ylim=...`

After drawing a plot:

- add curves and points with `lines(...)` bzw. `points(...)`
- add labels (text) with `text(...)`
- add title with `title(...)`
- add legend with `legend(...)`

6.7 Save plots to files

- PostScript:

```
x <- matrix(rnorm(5000),ncol=2)
plot(x)
postscript(file = "MyPlot.ps", width = 5, height = 5.5, horizontal = FALSE)
plot(x)
dev.off()
```

- other are formats for example `pdf`, `pictex`, `xfig`, `png`, `jpeg`
  $\longrightarrow$ see `?Devices`

7 Some statistics

Credit scoring data:

```
file <- read.csv("kredit.csv",sep=";")
y <- 1-file$kredit        # default set to 1
prev    <- (file$moral >2)+0                             # previous loans were OK
employ  <- (file$beszeit >1)+0                           # employed (>=1 year)
dura    <- (file$laufzeit)                               # duration
d9.12   <- ((file$laufzeit >9)&(file$laufzeit <=12)) +0  #  9 < duration <= 12
d12.18  <- ((file$laufzeit >12)&(file$laufzeit <=18))+0  # 12 < duration <= 18
d18.24  <- ((file$laufzeit >18)&(file$laufzeit <=24))+0  # 18 < duration <= 24
d24     <- (file$laufzeit >24)+0                         # 24 < duration
amount  <- file$hoehe                                    # amount of loan
age     <- file$alter                                    # age of applicant
savings <- (file$sparkont > 4)+0                         # savings >= 1000 DM
phone   <- (file$telef==1)+0                             # applicant has telephone
foreign <- (file$gastarb==1)+0                           # non-german citizen
purpose <- ((file$verw==1)|(file$verw==2))+0             # loan is for a car
house   <- (file$verm==4)+0                              # house owner
```

7.1 Characteristics

```
kredit <- data.frame(y,age,amount,dura,prev,savings,house)

summary(kredit)

mean(kredit$age)
sd(kredit$age)
var(kredit$age)

cov(kredit[,1:3])
cor(kredit[,1:3])

median(kredit$age)
quantile(kredit$age,c(0.1,0.5,0.9))

library(help=e1071)
library(e1071)
skewness(kredit$age)
kurtosis(kredit$age)

skewness(rnorm(1000))
kurtosis(rnorm(1000))
```

7.2 Tables

```
length(kredit$age)
length(unique(kredit$age))

table(kredit$age)
table(kredit$dura)
table(kredit$savings)

table(kredit$y, kredit$savings)
table(kredit$y, kredit$savings)/nrow(kredit)

table(kredit$y, kredit$savings, kredit$house)

unique(kredit[,c("y","savings","house")])
```

7.3 Regression and time series analysis

7.3.1 Linear regression

```
plot(kredit$age, kredit$dura)

lm <- lm( dura ~ age, data=kredit)
summary(lm)                                  ## dependence on age
o <- order(kredit$age)
lines(kredit$age[o], lm$fitted.values[o], col="red", lwd=2)

lm2 <- lm( dura ~ age + amount, data=kredit)
summary(lm2)                                 ## dependence on age+amount

lm3 <- lm( dura ~ amount, data=kredit)
summary(lm3)                                 ## dependence on amount
plot(kredit$amount, kredit$dura)
o <- order(kredit$amount)
lines(kredit$amount[o], lm3$fitted.values[o], col="red", lwd=2)

lm4 <- lm( dura ~ amount + I(amount^2), data=kredit)
summary(lm4)                                 ## dependence on amount (also squared)
lines(kredit$amount[o], lm4$fitted.values[o], col="blue", lwd=2)
```

→ duration of loan is clearly a function of amount:

```
> summary(lm4)

Call:
lm(formula = dura ~ amount + I(amount^2), data = kredit)

Residuals:
     Min       1Q   Median       3Q      Max
-34.6115  -5.5761  -0.9547   5.0850  42.1110

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  8.410e+00  6.516e-01  12.906  < 2e-16 ***
amount       4.855e-03  2.961e-04  16.393  < 2e-16 ***
I(amount^2) -1.815e-07  2.309e-08  -7.863  9.7e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.144 on 997 degrees of freedom
Multiple R-Squared: 0.4262,     Adjusted R-squared: 0.425
F-statistic: 370.3 on 2 and 997 DF,  p-value: < 2.2e-16
```

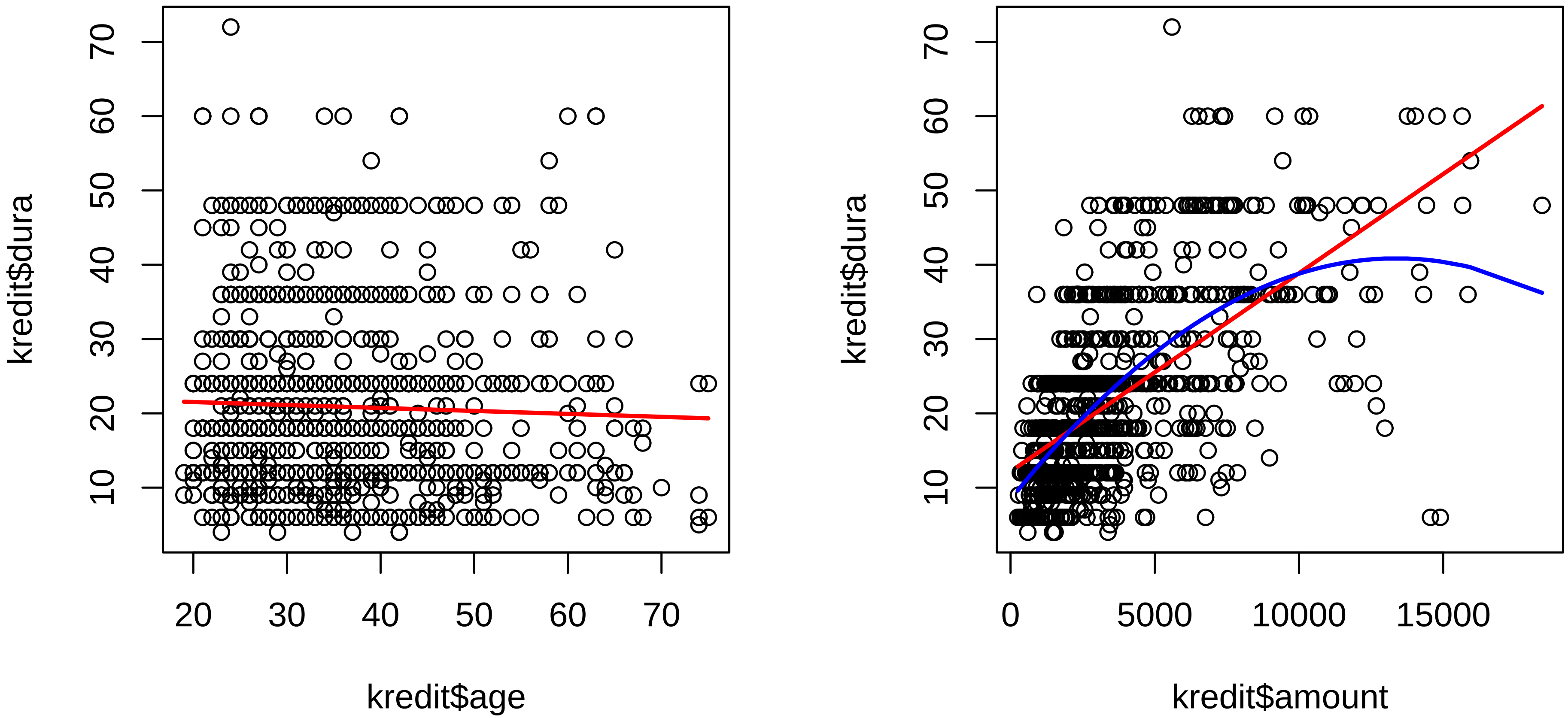


Fig. 8: Dependence of duration on age (left) and amount (right)

7.3.2 Generalized linear model (GLM)

→ estimation of default probabilities

```
logit <- glm(y ~ age + amount + dura + prev + savings + house,
             family=binomial(link = "logit"))
summary(logit)

logit2 <- glm(y ~ age + amount + I(amount^2) + dura + prev + savings + house,
              family=binomial(link = "logit"))
summary(logit2)
```

→ default probabilities are (among others) non-linearly dependent on amount:

```
> summary(logit2)

Call:
glm(formula = y ~ age + amount + I(amount^2) + dura + prev +
    savings + house, family = binomial(link = "logit"))

Deviance Residuals:
```

```
    Min       1Q   Median       3Q      Max
-2.1244  -0.8495  -0.6196   1.0935   2.2584

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -4.637e-01  3.035e-01  -1.528  0.12652
age         -1.748e-02  7.159e-03  -2.442  0.01460 *
amount      -2.070e-04  9.348e-05  -2.214  0.02679 *
I(amount^2)  1.870e-08  6.941e-09   2.694  0.00707 **
dura         3.992e-02  8.106e-03   4.925 8.46e-07 ***
prev        -7.589e-01  1.619e-01  -4.688 2.76e-06 ***
savings     -9.897e-01  2.232e-01  -4.435 9.22e-06 ***
house        6.277e-01  2.073e-01   3.027  0.00247 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1221.7  on 999  degrees of freedom
Residual deviance: 1102.1  on 992  degrees of freedom
AIC: 1118.1

Number of Fisher Scoring iterations: 4
```

7.3.3 Selected routines for regression and times series analysis

Model	Routines
linear	`lm`, `anova`
GLM	`glm`, `mgcv` / `gam` (additively nonparametric)
nonlinear	`nls`
nonparametric	`locpoly`, `locfit`
mixed models	`lmm`, `nlme`, `glmmML`, `glmmPQL`
time series	`ar`, `arma`, `arima`, `arima0`, `garch`
classification and regression trees	`tree`

Packages: `MASS`, `stats`, `KernSmooth`, `tseries`, `gam`, `gcv`

7.4 Hypothesis testing

7.4.1 Test for normality

```
library(KernSmooth)
f <- bkde(kredit$age)
plot(f, type="l", xlim=range(f$x), ylim=range(f$y))
title("Distribution of Age")         ## distribution is normal?

t <- shapiro.test(kredit$age)
t
t$p.value

library(tseries)
t <- jarque.bera.test(kredit$age)
t
t$p.value
```

7.4.2 Comparing distributions

```
library(KernSmooth)
f0 <- bkde(kredit$age[y==0])
f1 <- bkde(kredit$age[y==1])
plot(f0, type="l", col="blue", xlim=range(c(f0$x,f1$x)), ylim=range(c(f0$y,f1$y)))
lines(f1, col="red")
title("Age vs. Default")         ## same distribution?

t <- ks.test(kredit$age[y==1],kredit$age[y==0])
t
t$p.value

t <- wilcox.test(kredit$age[y==1],kredit$age[y==0])
t
t$p.value
```

7.4.3 Selected tests

Test	**Routines**
comparing means (t-Tests)	`t.test`
comaring variances (F-Tests)	`var.test`
binomial tests	`prop.test`, `binom.test`
correlation	`cor.test`
rank tests	`wilcox.test`
regression	`anova`
unit roots (mean reversion)	`adf.test`, `kpss.test`

Packages: `stats`, `tseries`, `exactRankTests`

8 "Advanced" mathematics

8.1 Optimizing functions

linear model

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \varepsilon_i$$

```
n <- 100; b <- c(-1,3)
x <- matrix(rnorm(n*length(b)),ncol=length(b))   ## regressors
e <- rnorm(n)/4

y <- 1 + x %*% b + e                             ## linear model

l <- lm( y~x ); summary(l)                       ## built-in linear model
```

$\longrightarrow$ to optimize

$$QS = \sum_i (y_i - x_i^\top \beta)^2$$

(agreed, this is not a very useful example for iterative optimization ;-))

Optimization (without intercept)

```
QS <- function(b, x, y){ sum( (y - x %*% b)^2 ) }  ## objective function

b0 <- c(0,0)
opt <- optim(b0, QS, method="BFGS", x=x, y=y)      ## optimization (without intercept!)
opt
sum( (x %*% opt$par - mean(y))^2 )/sum( (y-mean(y))^2 )    ## R^2
```

→ coefficient of determination R^2 might be outside $[0, 1]$

Optimization (with intercept)

```
b1 <- c(0,0,0)
x1 <- cbind(rep(1,n),x)
opt1 <- optim(b1, QS, method="BFGS", x=x1, y=y)    ## optimization (without intercept!)
opt1
sum( (x1 %*% opt1$par - mean(y))^2 )/sum( (y-mean(y))^2 )  ## R^2
```

Optimization with gradient

$$QS = \sum_i (y_i - x_i^\top \beta)^2 = (y - \mathcal{X}\beta)^\top (y - \mathcal{X}\beta), \quad \frac{\partial QS}{\partial \beta} = -2\mathcal{X}^\top y + 2\mathcal{X}^\top \mathcal{X}\beta$$

```
D.QS <- function(b, x, y){ -2* t(x) %*% y + 2* t(x) %*% x %*% b }  ## gradient

opt2 <- optim(b1, QS, D.QS, method="BFGS", x=x1, y=y)
opt2
sum( (x1 %*% opt2$par - mean(y))^2 )/sum( (y-mean(y))^2 )  ## R^2
```

Optimization with box constraints (e.g. $\beta_j \geq 0$)

```
b2 <- c(0,0,0)
x1 <- cbind(rep(1,n),x)
opt3 <- optim(b1, QS, D.QS, method="BFGS", lower=0, x=x1, y=y)
opt3
sum( (x1 %*% opt3$par - mean(y))^2 )/sum( (y-mean(y))^2 )  ## R^2
```

Optimization with linear constraints (z.B. $\beta_0 \geq 0$, $\beta_1 + \beta_2 \leq 2$)

$$\mathcal{U}\beta - c = \begin{pmatrix} 0 & -1 & -1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{pmatrix} - \begin{pmatrix} -2 \\ 0 \end{pmatrix} \geq 0$$

```
u <- cbind( c(0,1), c(-1,0), c(-1,0) )
c <- c(-2,0)

applyDefaults <- function(fn, ...) {   ## for transferring further parameters
  function(x) fn(x, ...)               ## to QS and D.QS
}

b4 <- rep(0.5,3)
opt4 <- constrOptim(b4, applyDefaults(QS, x=x1, y=y),
                    applyDefaults(D.QS, x=x1, y=y), ui=u, ci=c)
opt4
sum( (x1 %*% opt4$par - mean(y))^2 )/sum( (y-mean(y))^2 )  ## R^2
```

8.2 Interpolation

→ `approx` for linear, `spline` and `interpSpline` for spline approximation

```
x <- seq(-5,5,by=1)
y <- sin(x)

xx <- seq(-5,5,by=0.1)
y.approx <- approx(x,y, xout=xx)$y
yy <- sin(xx)

plot(xx,yy, type="l", col="green")
lines(xx,y.approx, lwd=2)

library(splines)
sp <- interpSpline(x,y)
lines(predict(sp,xx), col="red")
```

8.3 Numerical integration

→ `integrate` for 1-dimensional, `adapt` for multidimensional integration

```
pnorm(0)

it <- integrate(dnorm, -Inf,0)
it

attributes(it)     ## result is object of class "integrate"

it$value

pmvnorm(c(0,0))
pmvnorm(c(0,0))[[1]]

library(adapt)
it <- adapt(2, c(-Inf,-Inf), c(0,0), functn=dmvnorm)

attributes(it)     ## result is object of class "integration"

it$value
```

9 Basics in programming

9.1 Functions

```
myfun <- function(x, a){
  r <- a*sin(x)
  return(r)
}
myfun(pi/2,2)

myfun1 <- function(x, a){ a*sin(x) }       ## same as myfun
myfun1(pi/2,2)

myfun2 <- function(x, a=1){                ## optional parameter with default value=1
  a*sin(x)
}
myfun2(pi/2,2)
myfun2(pi/2)

myfun3 <- function(x, a=NULL){             ## optional parameter without default value
  if (!is.null(a)){ a*sin(x) }else{ cos(x) }
}
myfun3(pi/2,2)
myfun3(pi/2)
```

```
myfun4 <- function(x, a=1){
  r1 <- a*sin(x); r2 <- a*cos(x)
  return(r1=r1,r2=r2)                 ## two results (depreciated!)
}
myfun4(pi/2)

myfun5 <- function(x, a=1){
  r1 <- a*sin(x); r2 <- a*cos(x)
  return(list(r1=r1,r2=r2))           ## one result (list of two!)
}
myfun5(pi/2)

myfun6 <- function(x, a=1, b=2){
  r1 <- a*sin(x); r2 <- b*cos(x)
  return(list(r1=r1,r2=r2))
}
myfun6(pi/2)                          ## a=1, b=2 (defaults)
myfun6(pi/2,1,2)                      ## a=1, b=2 (explicitely given)

myfun6(pi/2,2)                        ## a=2, b=2 (only a explicitely given)
myfun6(pi/2,a=2)                      ## a=2, b=2 (only a explicitely given)

myfun6(pi/2,b=3)                      ## a=1, b=3 (only b explicitely given)
```

→ input parameters may be omitted (if reasonable); multiple output parameters are in fact elements of a list

9.2 Conditional instructions, loops

- `if` & co.

```
x<- 1; if (x==2){ print("x=2") }
x<- 1; if (x==2){ print("x=2") }else{ print("x!=2") }
```

- `for` & `repeat`

```
for (i in 1:4){ print(i) }
for (i in letters[1:4]){ print(i) }
i <- 0; while(i<4){ i <- i+1; print(i)}
i <- 0; repeat{ i <- i+1; print(i); if (i==4) break }
```

- other: `ifelse`, `switch`

9.3 "Set theory"

```
a <- 1:3; b <- 2:6; a %in% b; b %in% a
a <- c("A","B"); b <- LETTERS[2:6]; a %in% b; b %in% a
```

9.4 Packages

- packages comprise (one or) more functions, are loaded with `library(<Package-Name>)`; available functions in a package can be queried with `library(help=<package-name>)`
- to create self-written packages, there exist two helpful functions:

  `package.skeleton(<package-name>)`
  generates the appropriate directory structure of the packages with templates for the necessary files

  `prompt(<Funktion>)`
  generates a template for the help text for a function
- collections of packages are called bundles
- packages or bundles may be installed with the according menu item under Windows; under Unix/Linux one uses `install.packages` or `R CMD INSTALL <Package-...>.tar.gz`

9.5 DLLs

C function:

```
#include <stdlib.h>
#include <math.h>

/* Compile into shared library: gcc -shared -O2 -o mydll.so mydll.c */

int mysum(double *dim, double *x, double *y, double *z)
{
    long i, n;
    n=dim[0];

    for (i=0; i<n; i++)   /* loop over obs  */
    {
        z[i] = x[i] + y[i];
    }
    printf ("mysum in C\n");
    return 0;
}
```

Call in R:

```
dyn.load("mydll.so")                                ## load DLL
is.loaded("mysum")                                  ## "mysum" is available?

d <- 3
x <- 1:3
y <- 4:6
z <- rep(0,3)

r <- .C("mysum", dim=d, x=x, y=y, z=z )             ## that doesn't work!
r$z

d <- as.double(3)
x <- as.double(1:3)
y <- as.double(4:6)
z <- rep(0.0,3)

r <- .C("mysum", dim=d, x=x, y=y, z=z )             ## this is the way to go ...
r$z
z                                                   ##    -> z is still =0

r <- .C("mysum", dim=d, x=x, y=y, z=z, DUP=FALSE)  ## another way (without copying)
r$z
z                                                   ##    -> z contains the result

dyn.unload("mydll.so")                              ## unload DLL
```

9.6 Tips & tricks

- syntax highlightening (and R in (X)Emacs integration):
  download ESS = “emacs speaks statistics” from http://ess.r-project.org/
  and add to `.emacs`
  `(load "<path to ESS>/ess-5.1.24/lisp/ess-site")`
- syntax highlightening for Windows is also available in WinEdt
  (http://cran.at.r-project.org/contrib/extra/winedt)
- rounding and formatting of numbers works with `round`, `floor`,
  `ceiling`, `signif`, `formatC`
- strings (character vectors) can be edited with `paste`, `substr`, `nchar`,
  `strsplit`, `toupper`, `tolower`, `sub`
- time dates can be generated with `as.POSIXlt` and `strptime`, e.g.
  `as.POSIXlt( strptime("20050101","%Y%m%d"))+(0:364)*86400`
  creates all days of the year 2005;

```
d <- as.POSIXlt( strptime("20050926","%Y%m%d")); d$wday
```

shows the weekday of Sep 26, 2005

- `system` executes an OS command, e.g. under Linux
  `system("cal 09 2005")`
- `xtable` (package: `xtable`) and `latex` (package: `Hmisc`) can save R object into LaTeX code
- `eval` and `parse` evaluate strings as expressions, e.g.

```
eval( parse( text=paste("x.",as.character(1:2)," <- 0",sep="") ) )
print(x.1)
```

- there are two methods for OOP in R: S3- and S4-classes; for obtaining information about the components of a S3 class (former approach) one uses `class` and `attributes` while for a S4 class (newer approach) `getClass`, `slot`, `slotNames` are useful
- methods can be class-dependent, e.g. `methods(print)` gives all functions belonging to the print function

References

Becker, R. A. and Chambers, J. M. (1984). *S. An Interactive Envrionment for Data Analysis and Graphics*, Wadsworth and Brooks/Cole, Monterey.

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988). *The New S Language*, Chapman & Hall, London.

Chambers, J. M. and Hastie, T. J. (1992). *Statistical Models in S*, Chapman & Hall, London.

Dalgaard, P. (2002). *Introductory Statistics with R*, Springer. ISBN 0-387-95475-9.
URL: *http://www.biostat.ku.dk/ pd/ISwR.html*

Ligges, U. (2005). *Programmieren mit R*, Springer-Verlag, Heidelberg. ISBN 3-540-20727-9, in German.
URL: *http://www.statistik.uni-dortmund.de/ ligges/PmitR/*

Murrell, P. (2005). *R Graphics*, Chapman & Hall/CRC, Boca Raton, FL. ISBN 1-584-88486-X.
URL: *http://www.stat.auckland.ac.nz/ paul/RGraphics/rgraphics.html*

Venables, W. N. and Ripley, B. D. (2000). *S Programming*, Springer. ISBN 0-387-98966-8.
URL: *http://www.stats.ox.ac.uk/pub/MASS3/Sprog/*

Venables, W. N. and Ripley, B. D. (2002). *Modern Applied Statistics with S. Fourth Edition*, Springer. ISBN 0-387-95457-0.
URL: *http://www.stats.ox.ac.uk/pub/MASS4/*