R-Howto

R-Howto

Version 8, 18.02.2024

Dr. Richard Ottermanns

Computational Ecotoxicology (COPE)

Institute for Environmental Research (IFER)

RWTH Aachen University

Worringerweg 1

52074 Aachen, Germany

ottermanns@ifer.rwth-aachen.de

Contents

Preface

This howto is meant as a quick reference guide to most common but also some advanced R-stuff. It is saved as a plain text-file (UTF-8, No BOM). Actually it is a markdown source file containing R-code snippets. You can use the document by simply opening it in you favorite markdown environment (e.g. pandoc) and convert it to pdf, doc, odt, html or whatever. Or you can open it in R or RStudio and execute the R-snippets you can find between the ~~~{.r}- and the ~~~-tags. For the latter never use the pdf-document you may have created, because it will contain characters incompatible with R.

To keep things as simple as possible all code is streamlined by removing **all** unnecessary spaces and indents. Spaces are only used where absolutely necessary and indents are only used for code lines that must be broken to fit onto the page so that you can always copy the code and transfer it to your R-console without further need of editing. Nevertheless, be aware of page brakes that can sneak into your copied code.

All data sets used in the examples are available from the data-folder preferring plain-text csv files over spreadsheet files or Rdata-objects.

Most examples are "basic" and as simple as possible, avoiding all additional packages that are not part of the base R installation. They might not be the most elegant solutions and will not give you the most glossy plots, but you can be sure that they will work with almost any version of the R environment. Often you will also find more "advanced" examples using a little code and eye candy. And if you are lucky you might also find some "all-in"-examples, that might be full of completely unnecessary stuff but nice to see.

The latest version of this howto can always be found in plain text and as a pre-compiled pdf under https://cope.rwth-aachen.de/#datalab_rhowto.

1. Install and config R

1.1 Download

http://cran.r-project.org/

1.2 Start and quit

Start R on the console

```
R
```

Quit R on the console

```
q()
```

Configure start-up etc.

see ~/.Rprofile

Set standard lib path in .Rprofile

Unix

```
.libPaths(c("/usr/lib/R/site-library","/usr/lib/R/library"))
```

Windows

```
.libPaths(c("C:/Program Files/R/R-4.0.3/library"))
```

1.3 Help

Help function

```
help(function name)
```

1.4 Set working directory

Linux

```
setwd('/home/user/tmp/data')
```

Windows

Mind the double backslahes!

```
setwd('C:\\users\\user\\tmp\\data'')
```

Mac

```
setwd('/Users/username/tmp/data')
```

Set working directory permanently

```
sudo vi /etc/R/Rprofile.site:
...
.First<-function() cat("n Welcome to R!nn")
setwd('/home/user/tmp')
.Last<-function() cat("n Goodbye!nn")
...
```

List files

```
dir()
```

Empty command window

```
STRG-l
```

1.5 Package management

Install package from internet (and load it)

```
install.packages('packagename')
library('packagename')
require('packagename')
```

Install package from the hard disc

```
install.packages('/pathtofile',repos=NULL,type="source")
```

Install package from archive

see e.g. https://stackoverflow.com/questions/24194409/how-do-i-install-a-package-that-has-been-archived-from-cran

e.g. https://cran.r-project.org/src/contrib/Archive/SimHap/SimHap_1.2.0.tar.gz:

```
sudo su - -c "R -e \"devtools::install_url('https://cran.r-project.org/src/contrib/
    Archive/SimHap/SimHap_1.2.0.tar.gz')\""
```

Install packages into R from command line

```
Rscript -e 'install.packages("drat",repos="https://cloud.r-project.org")'
```

Install packages into R from command line system-wide

```
sudo Rscript -e 'install.packages("tcl",lib="/usr/local/lib/R/site-library")'
```

Install packages into Rstudio from command line system-wide

```
sudo Rscript -e 'install.packages("tcl", lib="/usr/local/lib/R/site-library")'
```

Update packages

```
update.packages()
```

List version number of installed packages

```
installed.packages()
```

Get package version

```
packageVersion(<packagename>)
```

1.6 Scripts

Execute script from within R

```
source('mysamplescript.r')
```

Execute script from outside R (batch mode)

```
R CMD BATCH /home/user/tmp/myprog.r
```

1.7 Editors and GUIs

Install editor for Linux

```
sudo apt-get install r-cran-rcmdr
```

```
library(Rcmdr)
```

Send commands to R

Linux and Windows

`STRG-R`

Mac

`Cmd Return`

Edit a file from within R

```
file.edit('filename')
```

RStudio

https://posit.co

Measure runtime of a script

```
start<-Sys.time()
stop<-Sys.time()
time<-difftime(stop,start,units='secs')
time
```

2. Basic mathematical functions

2.1 Arithmetical operators

Addition: +

Subtraction: -

Multiplication: *

Division: /

Exponentiation: ^

2.2 Logical operators

And: AND / &

Or: OR

Xor: XOR

Larger: >

Smaller: <

Equal: ==

Not: !

2.3 Handle and convert objects and variables

Generate variables and assign values

```
a=2
b=3
```

```
a<-2
b<-3
```

Calculation and output to the command windows

```
a*b
```

Calculation and assignment to new variable

```
x=a*b
x<-a*b
```

Print content of a variable

```
x
```

Get class of variable

like data.frame, matrix etc.

```
class(data)
```

Get variable type

like single, double, string, ...

```
typeof(data)
```

Convert object to data frame

```
as.data.frame(data)
```

2.4 Some mathematical functions

Factorial

```
x<-3
factorialx<-gamma(x+1)
factorialx
```

Binomial coefficient x over y

```
x<-3
y<-2
choose(x,y)
```

Absolute value of a variable

```
abs(x)
```

Sign of a variable

```
sign(x)
```

Ranking data

```
x<-c(5.2,2.6,7.7,3.5)
rank(x)
```

3. Vectors and matrices

3.1 Generate vectors and matrices

Generate vector

```
x=c(2,3)
```

Generate matrix

```
mdat<-matrix(c(1,2,3,11,12,13),nrow=2,ncol=3,byrow=TRUE,dimnames=
    list(c("row1","row2"),c("C.1","C.2","C.3")))
```

Alternative

```
mdat<-rbind(x,y,z)
mdat<-cbind(x,y,z)
```

Generate 3x3 matrix with zeros

```
mdat<-mat.or.vec(3,3)
```

3.2 Vector algebra

Transpose matrix

```
tx=t(x)
```

Dot product (scalar product) of vectors

```
x<-c(1,2,3,4)
y<-c(5,6,7,8)
z<-x%*%y
```

Vector multiplication by components

```
x<-c(1,2,3,4)
y<-c(5,6,7,8)
z<-x*y
```

Print number of rows and columns of a matrix

```
nrow(as.matrix(z))
ncol(as.matrix(z))
```

Vector correlation

```
# Create vectors
a<-c(1,2,3)
a
b<-c(-7,8,9)
b
# Calculate scalar product=dot product
dp<-a%*%b
dp
# Calculate magnitude of vectors
na<-norm(a,type="2")
na
nb<-norm(b,type="2")
nb
```

```
# Correlatiion of vectors
corab<-dp/(na*nb)
corab
# Get angle between vectors in degrees
# use funktion deg from package circular
install.packages('circular')
library('circular')
alpha<-deg(acos(corab))
alpha
```

4. Data management

4.1 Import data

Text files with header

```
x<-read.table('daphnia.csv',sep=';',dec=',',header=T)
```

Text files with header and row names

```
x<-read.table('daphnia.csv',sep=';',header=T,row.names=1)
```

Text files with multicolumn header

skip 3 rows when reading csv (e.g. with wk1-header)

```
read.table('data.csv',header=TRUE,sep=',',skip=3)
```

Spreadsheet files

```
install.packages('gdata')
library('gdata')
data<-read.xls('mdata3.xls')
```

4.2 Manage data

Print available objects in memory (variables)

```
objects()
```

Delete all objects

```
rm(list=ls())
```

Print available variables within an object

```
names(x)
```

Call a variable (column) from an object

```
x<-read.table('daphnia.csv',sep=';',header=T)
x$dlength
x[1]
```

Make variables (columns) in objects directly accessible by column label (include in search path)

```
attach(x)
dlength
```

Remove objects from the search path

```
detach(x)
```

Data frames

```
lambda<-c(0.926,0.053,0.014,0.003)
index<-c(1,2,3,4)
dataplot<-data.frame(index,lambda)
plot(dataplot)
lines(dataplot$index,dataplot$lambda)
```

Create subsets per condition

```
data<-read.table('daphnia2.csv',sep=';',header=T)
attach(data)
subset1<-subset(data,treatment=='1'&dlength>2.1)
```

Create subset from multiple conditions

```
div.block1.2008<-subset(div,div$block=='1'&div$year=='2008')
```

Create subsets from row and column numbers

```
# data2<-data[FirstRow:LastRow,FirstColumn:LastColumn]
data2<-data[1:10,1:2]
```

Print dimension of an object

```
dim(data)
```

Print length of a vector

rows!

```
length(data)
```

Delete rows in data frame

```
newData<-myData[-c(2, 4, 6),]
```

Delete columns in data frame

```
newData<-myData[,-c(2, 4, 6)]
```

Convert table data to long format

```
data<-read.table('buttermilch.csv',sep=',',header=TRUE)
library(reshape)
melt(data)
```

Split variable names that a combination of multiple variables

```
library(reshape2)
x<-c("a_1","a_2","b_2","c_3")
vars<-colsplit(x,"_",c("trt","time"))
vars
```

4.3. Sort data

Ascending

```
y<-(data[order(data[,2]),])
```

Descending

```
y<-(data[order(-data[,2]),])
```

4.4. Data frames

Bind variabales

```
var1<-c(1,2,3)
var2<-c(4,5,6)
mydataframe<-cbind(var1,var2)
```

Use as matrix

```
mymatrix<-as.matrix(mydataframe)
```

4.5 Export data

4.5.1 Text files

```
write.table(data,'mydata.dat',sep=';',quote=F)
write.table(data,'mydata.dat',sep=';',row.names=F,col.names=F)
```

4.5.2. Spreadsheet files

```
install.packages('WriteXLS')
library('WriteXLS')
WriteXLS('data','myxls.xls')
```

4.5.3 Write log-files

Only outputs

```
sink('file.txt')
...
sink()
```

Inputs und outputs

```
install.packages('TeachingDemos')
library('TeachingDemos')
txtStart('file.txt')
...
txtStop()
```

5. Control statements

5.1 Loops

For loop

```
for(i in 1:(numberofspecies-1)){
for(j in (i+1):numberofspecies){
wert<-c(wert,cordata[i,j])
source<-c(source,tlabels[i])
target<-c(target,tlabels[j])
}
}
```

5.2 Conditions

If condition

Example 1:

```
if(cond){
do
}else if(cond){
do
}else{
}
```

Example 2:

```
if(abs(cordata[i,j])>th){
wert<-c(value,cordata[i,j])
}
```

Example 3:

```
if(a>0&&b>0){
wert<-c(value,cordata[i,j])
}
```

5.3 String operations

Text concatenation

```
a='Hallo'
b='World'
c=paste(a,' ',b,sep='')
```

Split a character vector

```
vec<-"A,B,C"
strsplit(vec,',')
```

5.4 Case statement

```
install.packages('memisc')
library('memisc')
df<-data.frame(x=rnorm(n=40),y=rnorm(n=40))
df<-df[do.call(order,df),]
(df<-transform(df,
x.2=cases(x>0,x<=0),
y.2=cases(y>0,y<=0),
z1=cases(
"Condition 1"=x<0,
"Condition 2"=(x>=0&y<0),
"Condition 3"=TRUE
```

```
),
z2=cases(x<0,(x>=0&y<0),(x>=0&y>=0))
))
xtabs(~x.2+y.2,data=df)
```

5.5 System calls

Execute system command

```
system('ls')
```

Save result of a system call to a variable

```
path<-system('pwd',intern=TRUE)
```

5.6 Input

From terminal

```
word<-readline(prompt="Enter a word: "); print(word)
```

6. Plots

6.1 Simple plots

Plot values

```
data<-read.table('daphnia.csv',sep=';',header=T)
attach(data)
stripchart(dlength)
stripchart(dlength,method='stack')
stripchart(dlength,vertical=TRUE)
stripchart(dlength,vertical=TRUE,method='jitter')
stripchart(dlength,vertical=TRUE,method='jitter',ylab='Daphnia length')
title('The daphnids')
```

Histogram

```
hist(dlength)
hist(dlength,breaks=8)
```

Histogram including normal distribution

```
hist(dlength)
xfit<-seq(min(dlength),max(dlength),length=40)
yfit<-dnorm(xfit,mean=mean(dlength),sd=sd(dlength))
yfit<-yfit*5
lines(xfit,yfit,col="blue",lwd=2)
```

Boxplot

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

Grouped boxplot

```
data<-read.table('daphnia2.csv',sep=';',header=T)
attach(data)
boxplot(dlength~treatment)
```

Add labels by hand

```
data<-read.table('daphnia2.csv',sep=';',header=T)
attach(data)
boxplot(dlength~treatment,axes=F)
axis(1,at=1:3,labels=c("Treatment 1","Treatment 2","Treatmant 3"))
axis(2)
box()
```

Rotate labels

```
boxplot(dlength~treatment,horizontal=T,las=2)
```

Change plot margins (below, left, above, right)

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

Barplot

```
barplot(table(treatment))
```

Barplot with error bars

```
x<-read.table('daphnia2.csv',sep=';',header=T)
attach(x)
xmeans<-tapply(dlength,treatment,mean)
xsds<-tapply(dlength,treatment,sd)
xbar<-barplot(xmeans)
maxxmeans=max(xmeans)
maxxsds=max(xsds)
barplot(xmeans,ylim=c(0,maxxmeans+maxxsds),xlab='Treatment',ylab='mean length per sort
    including standard deviation',names.arg=c('Treatment1','Treatment2','Treatment3'))
xupper=xmeans+xsds
xlower=xmeans-xsds
xlength=length(xmeans)
for(i in 1:xlength){
arrows(xbar[i],xlower[i],xbar[i],xupper[i],angle=90,code=3)
}
```

Plot variables pairwise (scatterplot)

```
data<-read.table('daphnia1.csv',sep=';',header=T)
attach(data)
plot(length,width)
```

Logarithmic axes

```
plot(length,width,log='xy')
```

3D scatter plot

```
space<-read.table('3dspace.csv',sep=';')
install.packages('scatterplot3d')
library(scatterplot3d)
attach(space)
scatterplot3d(V1,V2,V3,highlight.3d=TRUE,type="h")
```

Scatter plot matrix

```
gmodata<-read.table('gmo1.csv',sep=';',header=T)
pairs(gmodata)
```

Line plot

```
lambda<-c(0.926,0.053,0.014,0.003)
index<-c(1,2,3,4)
plot(index,lambda)
lines(index,lambda,type="l")
```

6.2 Export plots (Unix)

Set plot device

```
jpeg(filename="correlation.jpg",height=800)
png(filename="correlation.png",height=800)
bmp(filename="correlation.bmp",height=800)
plot ...
dev.off()
```

For EPS files

```
setEPS()
postscript("boxplot.eps")
plot ...
dev.off()
```

6.3 Format plots

Set axes labels

```
data<-read.table('daphnia1.csv',sep=';',header=T)
attach(data)
plot(length,width,xlab="Daphnia length [mm]", ylab="Daphnia width [mm]")
```

Scale axes

```
data<-read.table('daphnia1.csv',sep=';',header=T)
attach(data)
plot(length,width,xlim=c(0,23),ylim=c(0,6))
```

Set data point colour and symbol

```
data<-read.table('daphnia1.csv',sep=';',header=T)
attach(data)
plot(length,width,col="red",pch=21,cex=2)
```

pch symbols

pch = 0: square, 1: circle, 2: triangle point up, 3: plus, 4: cross, 5: diamond, 6: triangle point down, 7: square cross, 8: star, 9: diamond plus, 10: circle plus, 11: triangles up and down, 12: square plus, 13: circle cross, 14: square and triangle down, 15: filled square, 16: filled circle, 17: filled triangle point-up, 18: filled diamond, 19: solid circle, 20: bullet (smaller circle), 21: filled circle blue, 22: filled square blue, 23: filled diamond blue, 24: filled triangle point-up blue, 25: filled triangle point down blue

Line types

```
lty=("blank","solid","dashed","dotted","dotdash","longdash","twodash")
```

Add legend

```
legend("bottomright","Data set 1")
```

Set title

```
plot(length,width,main="My title")
```

Group a number of plots

```
data<-read.table('gmo1.csv',sep=';',header=T)
attach(data)
par(mfrow=c(3,1))
plot(data[,1],data[,2])
plot(data[,1],data[,3])
plot(data[,2],data[,3])
```

More sophisticated exapmles can be found here:

https://cran.r-project.org/web/packages/egg/vignettes/Ecosystem.html

Add lines

```
lines()
```

Add points

```
data<-read.table('daphnia1.csv',sep=';',header=T)
attach(data)
plot(length,width)
x=5.1
y=2.54
points(x,y,lty=1,col='red',cex=4)
```

Add a function to a plot

add=TRUE works only when adding functions, not with adding data

```
plot(funxy,add=T,col='green')
```

7. Univariate statistics

7.1. Explorative data analysis (EDA)

Simple statistics with eco(toxico)logical data (beans)

```
data<-read.table('daphnia2.csv',sep=';',header=T)
attach(data)
```

Print number of values in vector

```
n=length(dlength)
n
```

Frequencies

```
freq=table(treatment)
freq
```

7.2 Measures of central tendency and dispersion

Simple statistics with eco(toxico)logical data (beans)

```
data<-read.table('daphnia.csv',sep=';',header=T)
attach(data)
n=length(dlength)
n
```

Mean

```
meandlength=mean(dlength)
meandlength
```

Standard deviation

```
sddlength=sd(dlength)
sddlength
```

Variance

```
var(dlength)
```

Median

```
median(dlength)
```

Maximum

```
max(dlength)
```

Minimum

```
min(dlength)
```

Maximum, minimum, quantiles, mean and median in one call

```
summary(dlength)
```

Summary for complete object column-wise

```
summary(data)
```

Standard error of mean

```
serror=sddlength/sqrt(n)
serror
```

95% confidence interval (from normal distribution)

```
error<-qnorm(0.975)*serror
leftnv<-meandlength-error
rightnv<-meandlength+error
leftnv
rightnv
```

95% confidence interval (from t-distribution)

```
error<-qt(0.975,df=n-1)*serror
leftt<-meandlength-error
rightt<-meandlength+error
leftt
rightt
```

7.3 Test on normal distribution

Skewness and Kurtosis

```
library(moments)
skewness(x)
kurtosis(x)
```

should be around (0,3)

QQ-plot

using qqnorm:

```
qqnorm(x)
qqline(x)
```

you should observe a good fit of the straight line

using qqp:

```
library('MASS')
xt<-x+1 (no zeros)
qqp(xt,'norm')
```

QQ-Plots for alternative distibutions

log-normal:

```
qqp(xt,'lnorm')
```

Poisson:

```
poisson<-fitdistr(xt,'Poisson')
qqp(xt,'pois',poisson$estimate)
```

Negative binomial:

```
nbinom<-fitdistr(xt,'Negative Binomial')
qqp(xt,'nbinom',size=nbinom$estimate[[1]],mu=nbinom$estimate[[2]])
```

Gamma:

```
gamma<-fitdistr(xt,'gamma')
qqp(xt,'gamma',shape=gamma$estimate[[1]],rate=gamma$estimate[[2]])
```

P-plot

```
probplot(x,qdist=qnorm)
```

You should observe a good fit of the straight line

Fit normal density

```
f.den<-function(t) dnorm(t,mean(x),sqrt(var(x)))
curve(f.den,xlim=c(6,14))
hist(x,prob=T,add=T)
```

Kolmogorov-Smirnov test on normal distribution

```
data<-read.table('daphnia.csv',sep=';',header=T)
attach(data)
meandlength=mean(dlength)
sddlength=sd(dlength)
ks.test(data[2],pnorm,alternative=c("two.sided"),meandlength,sddlength)
# alternative=c("two.sided","less","greater")
```

Critical values calculated by simulation

```
# According to Lilliefors' (1967)
library(KScorrect)
nreps<-9999
x<-rnorm(n)
Lc<-LcKS(x,"pnorm",nreps=nreps)
sim.Ds<-sort(Lc$D.sim)
crit<-round((1-alpha)*nreps,0)
dmaxkrit<-round(sim.Ds[crit], 3)
dmaxkrit
```

Shapiro-Wilks test on normal distribution

```
shapiro.test(dlength)
```

Anderson-Darling test on normal distribution

```
install.packages('nortest')
library('nortest')
data<-read.table('daphnia.csv',sep=';',header=T)
attach(data)
ad.test(dlength)
```

Lilliefors test on normal distribution

mean and variance unknown:

```
install.packages('nortest')
library('nortest')
lillie.test(dlength)
```

7.4 Test on variance homogeneity

a) For normally distributed data

Bartlett's Test

```
bartlett.test(div.block1.2008$nem~div.block1.2008$treatment)
```

F-Test for given variances

```
s1=3.0
s2=2.0
n1=16
n2=11
F=s1^2/s^2
qf(0.95,n1-1,n2-1)
```

F-Test for raw data

```
data<-read.table('bohnen.txt',header=T)
data1<-read.table('bohnen1.txt',header=T)
var.test(data,data1$laenge,ratio=1,alternative=c("two.sided","less","greater"
    ),conf.level=0.95,...)
```

b) For non-normally distributed data

Levene-Test

```
install.packages('car')
library('car')
leveneTest(div.block1.2008$nem~div.block1.2008$treatment)
```

Flinger-Test

```
fligner.test(div$nem~div$treatment)
```

Ansari-Test

```
ansari.test
```

Mood-Test

```
mood.test
```

7.5 z-test

z-Test for a mean when the standard deviation of the population is known

http://www.rforge.net/NCStats/InstallNCStats.R

```
library('NCStats')
data<-read.table('beans.csv',sep='\t',header=T)
sd.beans<-sd(data$length)
z.test(data$length,mu=21.5,sd=sd.beans,alternative=c("two.sided"),
    conf.level=0.95)
# alternative=c("two.sided","less","greater")
```

Alternatively

```
z=(192-200)/(35/sqrt(49))
qnorm(0.95,mean=0,sd=1,lower.tail=TRUE,log.p=FALSE)
```

7.6 t-test

2-sample t-test

here one-sided

```
sample1=data[1:500,2]
sample2=data[501:1000,2]
t.test(sample2,sample1,alternative="greater")
sample3=sample2+10
t.test(sample3,sample1,alternative="greater")
```

2-sample paired t-test

here one-sided

```
t.test(sample1,sample2,paired=T,alternative="greater")
```

t-test for given means

```
t=((2.6-3.4)-0)/(sqrt(0.97^2/10+0.97^2/10))
qt(0.975,18)
```

Welch t-test for unequal variances

From given means:

```
n1=10
s1=0.97
n2=10
s2=0.97
t=((2.6-3.4)-0)/(sqrt(0.97^2/10+0.97^2/10))
fg=(s1^2/n1+s2^2/n2)/(((s1^2/n1)^2)/(n1-1)+((s2^2/n2)^2)/(n2-1))
qt(0.975,fg)
```

From raw data:

```
data<-read.table('beans.csv',sep='\t',header=T)
data1<-read.table('beans1.csv',sep='\t',header=T)
t.test(data,data1$length)
```

7.7 Nonparametric tests

Wilcoxon signed rank test

```
data<-c(974,1044,1093,897,879,1157,839,824,796,767)
wilcox.test(data,a="less",mu=1000,conf.level=0.95)
```

Alternative beim Vorliegen von ties

```
library('exactRankTests')
wilcox.exact(data,alternative="less",mu=51,conf.level=0.95,correct=FALSE,exact=TRUE)
```

Kritische Werte

```
alpha<-0.05
n<-length(data)
qsignrank(alpha/2,n,lower.tail=TRUE)
```

Mann-Withney U-test

```
iso<-c(80,80,70,100,35,30,100,90,70,85,40,15)
bt<-c(0,15,5,0,5,25,40,5,20,5,15,15)
wilcox.test(iso,bt,a="two.sided",mu=0,paired=T,conf.level=0.95)
```

Kruskal-Wallis k sample test

```
x<-c(2.9,3.0,2.5,2.6,3.2)
y<-c(3.8,2.7,4.0,2.4)
z<-c(2.8,3.4,3.7,2.2,2.0)
kruskal.test(list(x,y,z))
```

```
data<-read.table('kruskaldata.csv',sep=',',header=TRUE)
kruskal.test(diameters~treatment,data=data)
```

Critical values for Kruskal-Wallis

```
alpha<-0.05
ns<-c(5,5,5)
library('NSM3')
cKW(alpha,n=ns,method=NA,n.mc=10000)
```

7.8 Chi-square tests

Contingency table

```
data<-matrix(c(110,29,30,1),2,2)
dimnames(data)<-list(c("Bt","Control"),c("alive","dead"))
data
chisq.test(data)
```

7.9 Binomial test

Exact binomial test

```
data<-matrix(c(110,29,30,1),2,2)
dimnames(data)<-list(c("Bt","Control"),c("alive","dead"))
nControl<-29+1
aliveControl<-29
paliveControl<-aliveControl/nControl
nBt<-110+30
aliveBt<-110
paliveBt<-aliveBt/nBt
binom.test(aliveBt,nBt,paliveControl,alternative='less',conf.level=0.95)
```

7.10 Moment statistics

```
# Read data
data<-read.table("bn.dist.csv",header=TRUE,sep=',')

# Histogramm with empirical density and fitted normal distribution
png("hist.png")
par=hist(data$x,main="",xlab="")
par(new=TRUE)
plot(density(data$x),xlab="dbh [cm]",ylab="",main="",axes=FALSE,col='black')
axis(4)
mtext("Density",4)
normden<-function(t) dnorm(t,mean(data$x),sqrt(var(data$x)))
curve(normden,xlim=c(0,80),col='red',add=TRUE)
dev.off()

# Calculate moments
# install.packages("moments")
library(moments)

mean(data$x)
moment(data$x,order=1)

sd(data$x)
moment(data$x,order=2)

# Test whether the skewness is significant
skewness(data$x)
moment(data$x,order=3)
# Test for less if skewness is positive (>0)
agostino.text(data$x)

kurtosis(data$x)
moment(data$x,order=4)
anscombe.test(data$x)
```

8. Distributions and sampling

8.1 Empirical cummulative density functions

Generate ECDFs

```
# Calculate empirical cummulative density distribution
data<-read.table('daphnia.csv',sep=';',header=T)
attach(data)
my.cdf<-ecdf(dlength)
my.cdf
```

Apply the ECDF

```
# Return the proportion less than or equal to the mean
meandlength=mean(dlength)
my.cdf(meandlength)
# Calculate the number of beans less than or equal to the mean
n=length(dlength)
my.cdf(meandlength)*n
# Return the proportion greater than the mean
1-my.cdf(meandlength)
```

Plot ECDF

```
plot(my.cdf)
```

Kernel density plot

```
d<-density(dlength)
plot(d)
```

Histogram with density plot on secondary axis

```
par=hist(dlength)
par(new=TRUE)
plot(density(dlength),xlab="",ylab="",main="",axes=FALSE,col='red')
axis(4)
mtext("Density",4)
```

8.2 Sampling from observational populations

Take sample from a (statistical) population

```
rndsample<-sample(dlength,20)
rndsample
```

Create sample distributions

Means from a population:

```
meandist<-replicate(128,mean(sample(dlength,20)))
hist(meandist)
# Lilliefors p-values from a population
pvalues<-replicate(128,lillie.test(sample(dlength,20))$p.value)
hist(pvalues)
```

8.3 Sampling from theoretical distributions

Take sample from a theoretical normal distribution

```
# number of observations
n=1e+06
# distribution mean
a=log(23)
# distribution standard deviation
b=1
# sample size
k=200
rndsample<-sample((rnorm(n,mean=a,sd=b)),k)
hist(rndsample)
```

Take sample from a theoretical Poisson distribution

```
# number of observations
n=1e+06
# distribution mean
l=log(23)
# sample size
k=200
rndsample<-sample((rpois(n,lambda=l)),k)
hist(rndsample)
```

Broken-stick model

```
lambda<-c(0.926,0.053,0.014,0.003)
index<-c(1,2,3,4)
screeplot<-data.frame(index,lambda)
plot(index,lambda,xlab="Axis",pch=16,ylab="Eigenvalue",main="Screeplot PCA")
install.packages('vegan')
library(vegan)
bmodel<-bstick(10)
points(bmodel,col="red")
lines(bmodel,type="l",col="red")
legend("topright","red: Broken-stick model")
```

9. Bivariate statistics

9.1 Covariance

```
data<-read.table('daphnia1.csv',sep=';',header=T)
attach(data)
cov(length,width)
```

9.2 Correlation

Pearson correlation

```
data<-read.table('daphnia1.csv',sep=';',header=T)
attach(data)
cor(length,width)
# Alternative
cov(length,width)/(sd(length)*sd(width))
```

Spearman rank correlation

```
cor(length,width,method="spearman")
# Alternative
ranklength=rank(length)
rankwidth=rank(width)
cor(ranklength,rankwidth)
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
cor(temp,length)
```

9.3 ANOVA models

1-one way ANOVA using function aov (treatment as fixed factor)

```
data<-read.table('daphnia2.csv',sep=';',header=T)
attach(data)
y<-aov(dlength~as.factor(treatment),data=data)
summary(y)
```

Non-parametric (rank-based) 1-way ANOVA using function aov

```
ranklength<-rank(dlength)
y<-aov(ranklength~treatment)
summary(y)
```

Non-parametric (rank-based) 1-way ANOVA using function lm

```
ranklength<-rank(dlength)
g.rank<-lm(ranklength~treatment)
y<-anova(g.rank)
summary(y)
```

2-way non-parametric ANOVA using lm

```
g.rank<-lm(ranklength~treatment+site)
anova(g.rank)
```

Repeated measurement ANOVA

```
data<-read.table('repANOVA.csv',sep=';',header=T)
attach(data)
am1<-aov(dv~factor+Error(subject/factor),data=data)
summary(am1)
```

Rank-based repeated measurement ANOVA

```
data<-read.table('repANOVA.csv',sep=';',header=T)
attach(data)
am1<-aov(rank~factor+Error(subject/factor),data=data)
summary(am1)
```

MANOVA

```
# Read data in cross table format
cross<-read.table('cross.csv',header=TRUE,sep=';')
# Calculate MANOVA
fit<-manova(as.matrix(cross[,2:5])~cross[,1])
summary(fit,test="Pillai")
```

9.4 Multiple tests (post-hoc tests)

1-one way ANOVA using function aov (treatment as fixed factor)

```
data<-read.table('daphnia2.csv',sep=';',header=T)
attach(data)
y<-aov(length~as.factor(treatment),data=data)
summary(y)
```

Unadjusted pairwise t-test

```
pairwise.t.test(length,as.factor(treatment),p.adjust="none",pool.sd=T)
```

Bonferroni correction

```
pairwise.t.test(length,as.factor(treatment),p.adjust="bonferroni",pool.sd=T)
```

Holm correction

```
pairwise.t.test(length,as.factor(treatment),p.adjust="holm",pool.sd=T)
```

Tukey pairwise post-hoc test

```
TukeyHSD(y)
plot(TukeyHSD(aov(length~as.factor(treatment)),conf.level=.95))
```

Scheffe's pairwise test

```
install.packages('agricolae')
library(agricolae)
df<-df.residual(y)
MSerror<-deviance(y)/df
Fc<-summary(y)["treatment",4]
st<-scheffe.test(length,treatment,df,MSerror,Fc,group=TRUE)
summary(st)
```

Dunnett's test against control group

```
install.packages('multcomp')
library('multcomp')
Group=as.factor(treatment)
summary(glht(aov(length~Group),linfct=mcp(Group="Dunnett")))
# Alternative:
install.packages('DunnettTests')
library('DunnettTests')
data<-read.table('daphnia2.csv',header=TRUE,sep=';')
qvSDDT(data)
```

Non-parametric multiple test against control (Kruskal-MC)

see publication Kais et al. (2017)

```
sudo apt-get install libgdal-dev libproj-dev
```

```
# install.packages('pgirmess')
library('pgirmess')
v3.clp.kombi.72h<-read.table('v3.clp.kombi.72h.tsv',header=TRUE,sep='\t')
Group=as.factor(v3.clp.kombi.72h$treatment)
Group<-relevel(Group,ref="pk")
kruskalmc(v3.clp.kombi.72h$erod~Group,cont="one-tailed")
```

10. Statistical modelling

10.1 Linear models

Linear model with one factor (= 1-way ANOVA)

```
data<-read.table('daphnia2.csv',sep=';',header=T)
attach(data)
lm.anova1.treatment<-lm(length~treatment)
summary(lm.anova1.treatment)
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
lm.anova.type<-lm(length~type)
summary(lm.anova.type)
```

Linear model with two factors (= 2-way ANOVA)

```
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
lm.anova2.treatment.type<-lm(length~treatment*type)
summary(lm.anova2.treatment.type)
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
lm.anova2.type.treatment<-lm(length~type*treatment)
summary(lm.anova2.type.treatment)
```

Linear model with one metric predictor (= linear regression)

```
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
lm.regression.length.temp<-lm(length~temp)
summary(lm.regression.length.temp)
plot(temp,length)
abline(lm.regression.length.temp,col="black")
```

Linear model with one factor and one covariable (= 1-way ANCOVA)

```
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
lm.ancova.treatment.temp<-lm(length~treatment+temp)
summary(lm.ancova.treatment.temp)
```

10.2 Fit deterministic functions

Logarithmic function

```
# Logarithmic fit for x
x=c(61,610,1037,2074,3050,4087,5002,6100,7015)
y=c(0.974206,1.16716,1.19879,1.28192,1.30739,1.32019,1.35494,1.36941,1.37505)
logEstimate=lm(y~log(x))
plot(x,predict(logEstimate),type='l',col='blue')
lines(x,predict(logEstimate),col='red')
points(x,y)
summary(logEstimate)

# Logarithmic fit for y
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
orderdata<-(data[order(temp),])
x=orderdata[,6]
y=orderdata[,2]
logy=log(orderdata[,2])
logfit<-lm(logy~x)
summary(logfit)
par(mfrow=c(1,2))
plot(x,logy,xlab='temp',ylab='log(length)')
points(x,predict(logfit),col='blue')
lines(x,logfit$fitted.values,col='blue')
legend('topleft','black:raw data,blue:logfit')
plot(x,y,xlab='temp',ylab='length')
points(x,exp(predict(logfit)),col='blue')
lines(x,exp(logfit$fitted.values),col='blue')
legend('topleft','black:raw data, blue:predictions')
```

Polynomials

```
# 2nd order polynomial
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
y<-(data[order(temp),])
fit2<-lm(y[,2]~poly(y[,6],2))
summary(fit2)
par(mfrow=c(2,2))
plot(y[,6],y[,2],xlab='temp',ylab='length')
lines(y[,6],predict(fit2))
plot(fit2)
# 3rd order polynomial
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
y<-(data[order(temp),])
fit3<-lm(y[,2]~poly(y[,6],3))
summary(fit3)
par(mfrow=c(2,2))
```

```
plot(y[,6],y[,2],xlab='temp',ylab='length')
lines(y[,6],predict(fit3))
plot(fit3)
# 4th order polynomial
data<-read.table('daphnia3.csv',sep=';',header=T)
attach(data)
y<-(data[order(temp),])
fit4<-lm(y[,2]~poly(y[,6],4))
summary(fit4)
par(mfrow=c(2,2))
plot(y[,6],y[,2],xlab='temp',ylab='dlength')
lines(y[,6],predict(fit4))
plot(fit4)
```

10.3 Generalized linear models (GLM)

Linear model

```
# family=gaussian, link function: identity
data<-read.table('drc.csv',header=TRUE,sep=';')
attach(data)
alive<-t0-mort
fitlinear<-glm(mort/(mort+alive)~conc)
summary(fitlinear)
plot(conc,mort/(mort+alive))
points(conc,fitlinear$fitted.values,col='blue')
lines(conc,fitlinear$fitted.values,col='blue')
legend('bottomright','black:raw data,red:,blue:linear GLM')

# Alternative using lm:
fitlm<-lm(mort/(mort+alive)~conc)
plot(conc,mort/(mort+alive))
points(conc,fitlm$fitted.values,col='blue')
lines(conc,fitlm$fitted.values,col='blue')
```

Logistic regression for binary outcomes (logit models)

```
# link function:
$Pr(Y=1|X)=[1+e^{(-X'\beta)}]^{-1}$
data<-read.table('drc.csv',sep=';',header=T)
attach(data)
alive<-t0-mort
fitlogit<-glm(cbind(mort,alive)~conc,family=binomial)
summary(fitlogit)
confint(fitlogit)
```

Probit models for binary outcomes

```
# link function:
$Pr(Y=1|X)=\theta(X'\beta)$ (Cumulative normal pdf)
data<-read.table('drc.csv',sep=';',header=T)
attach(data)
alive<-t0-mort
fitprobit<-glm(cbind(mort,alive)~conc,family=binomial(link=probit))
plot(conc,mort/(mort+alive))
abline(fitprobit, col="black")

# Compare logit, probit and linear
plot(conc,mort/(mort+alive))
points(conc,predict(fitlogit,type='response'),col='red')
lines(conc,predict(fitlogit,type='response'),col='red')
points(conc,fitprobit$fitted.values,col='green')
lines(conc,fitprobit$fitted.values,col='green')
points(conc,fitlinear$fitted.values,col='blue')
lines(conc,fitlinear$fitted.values,col='blue')
legend('bottomright','black:raw data,red:logit GLM,green: probit GLM,blue:linear model')
```

Test residual deviance on significance

```
1-pchisq(fitlogit$deviance,length(conc))
1-pchisq(fitprobit$deviance,length(conc))
1-pchisq(fitlinear$deviance,length(conc))
```

Poisson regression for count data

```
data<-read.table('drc.csv',sep=';',header=T)
attach(data)
alive<-t0-mort
fitpoisson<-glm(mort~conc,family=poisson())
# not corrected for t0-->
summary(fitpoisson)
# Fit linear model to compare
fitlinearmort<-glm(mort~conc)
summary(fitlinearmort)
# Compare normal vs. Poisson
plot(conc,mort)
points(conc,fitpoisson$fitted.values,col='red')
lines(conc,fitpoisson$fitted.values,col='red')
points(conc,fitlinearmort$fitted.values,col='blue')
lines(conc,fitlinearmort$fitted.values,col='blue')
legend('bottomright','black:raw data,red:Poisson GLM,blue:Gaussian GLM')
1-pchisq(fitpoisson$deviance,length(conc))
1-pchisq(fitlinearmort$deviance,length(conc))
```

10.4 Comparing models

Akaike information criterion (AIC)

```
AIC (lm.anova1.treatment,lm.anova2.treatment.type,lm.ancova.treatment.temp)
AIC (logfit,fit2,fit3,fit4)
AIC(fitlogit,fitprobit,fitlinear)
AIC(fitlinear,fitpoisson)
```

Bayesian information criterion (BIC)

```
AIC (lm.anova1.treatment,lm.anova2.treatment.type,lm.ancova.treatment.temp, k=log(10))
AIC(fitlogit,fitprobit,fitlinear,k=log(10))
```

Chi-square test on model deviance

```
anova(fitlogit,fitprobit,test='Chisq')
```

Likelihood ratio test

```
install.packages('lmtest')
library('lmtest')
lrtest(model1,model2,model3)
```

10.5 Non-parametric regression

Moving average

```
data<-read.table('daphnia1.csv',sep=';',header=T)
# sort data.frame ascending due to dlength
y<-(data[order(data[,2]),])
# install.packages('caTools')
install.packages('caTools')
library('caTools')
col=c('black','red','green','blue','magenta','cyan')
# Compute a moving average for dwidth with windows size=2 and print the values
runmean(y[,3],2)
# Compare windows sizes in plot
plot(y[,2],y[,3])
lines(y[,2],runmean(y[,3],2),col=col[2])
lines(y[,2],runmean(y[,3],3),col=col[3])
lines(y[,2],runmean(y[,3],4),col=col[4])
lines(y[,2],runmean(y[,3],5),col=col[5])
lines(y[,2],runmean(y[,3],6),col=col[6])
lab=c('data','k=2','k=3','k=4','k=5','k=6')
legend('bottomright',lab,col=col,lty=1)
```

Kernel regression

Nadaraya–Watson kernel (=Gaussian?) kernels:

```
data<-read.table('daphnia1.csv',sep=';',header=T)
# sort data.frame ascending due to length
y<-(data[order(data[,1]),])
plot(y$length,y$width,ylab='Daphnia width [mm]',xlab='Daphnia length [mm]')
lines(ksmooth(y$length,y$width,kernel='normal',bandwidth=0.5),col='red')
lines(ksmooth(y$length,y$width,kernel='normal',bandwidth=1.0),col='green')
lines(ksmooth(y$length,y$width,kernel='normal',bandwidth=2.0),col='blue')
legend('bottomright','red:bw=0.5,green:bw=1.0,blue:bw=2.0')
# Get the values for bw=1.0
bw=1.0
values<-ksmooth(data[,1],data[,3],kernel='normal',bandwidth=bw,n.points=100)
values$x
values$y
```

Spline regression

```
data<-read.table('daphnia1.csv',sep=';',header=T)
# sort data.frame ascending due to length
y<-(data[order(data[,1]),])
plot(y$length,y$width,ylab='Daphnia width',xlab='Daphnia length')
lines(smooth.spline(y$length,y$width,df=7))
```

10.6 Dose response models

DRC-package

```
data<-read.table('drc.csv',sep=';',header=T)
attach(data)
# Install package drc
install.packages('drc')
library('drc')
# List available functions
getMeanFunctions(noParm=NA,fname=NULL,flist=NULL,display=TRUE)
# Fit models
model.probit<-drm(mort/t0~conc,weights=t0,data=data,fct=LL.2(),type="binomial")
model.logit<-drm(mort/t0~conc,fct=LL.2(),type="binomial")
model.weibull<-drm(mort/t0~conc,fct=W1.4())
```

Plot dose response curve

```
par(mfrow=c(3,1))
plot(model.probit,main='probit',log='x',xlim=c(0.01,100),ylim=c(0,1))
plot(model.logit,main='logit',log='x',xlim=c(0.01,100),ylim=c(0,1))
plot(model.weibull,main='weibull',log='x',xlim=c(0.01,100),ylim=c(0,1))
```

Model summary

```
summary(model.probit)
summary(model.logit)
summary(model.weibull)
```

Get EC50 values

```
ED(model.probit,c(10,50,90),interval="delta")
ED(model.logit,c(10,50,90),interval="delta")
ED(model.weibull,c(10,50,90),interval="delta")
```

Compare models

```
AIC(model.weibull,model.logit,model.probit)
AIC(model.weibull,model.logit,model.probit,k=log(10))
```

11. Multivariate statistics

11.1 Explorative data analysis (EDA)

Covariance matrix

```
data<-read.table('envCachemistry.csv',header=TRUE,sep=';')
labels<-data[,1]
data<-data[,-1]
tdata=t(data)
covdata<-cov(tdata)
covdata
install.packages('lattice')
require(lattice)
levelplot(covdata)
```

Correlation matrix

```
data<-read.table('envCachemistry.csv',header=TRUE,sep=';')
labels<-data[,1]
data<-data[,-1]
tdata=t(data)
cordata<-cor(tdata)
cordata
require(lattice)
levelplot(cordata,col.regions=grey(100:0/100))
```

Distance matrix

```
data<-read.table('envCachemistry.csv',header=TRUE,sep=';')
labels<-data[,1]
data<-data[,-1]
distmat<-dist(data,method="euclidean",diag=TRUE,upper=FALSE,p=2)
distmat
require(lattice)
levelplot(as.matrix(distmat),col.regions=grey(100:0/100))
# Alternative
install.packages('vegan')
library('vegan')
distmat<-vegdist(tdata,method="euclidean",binary=FALSE)
require(lattice)
levelplot(as.matrix(distmat),col.regions=grey(100:0/100))
```

Similarity matrix

```
data<-read.table('veg.csv',header=TRUE,sep=';')
data<-data[,-1]
tdata=t(data)
install.packages('proxy')
library('proxy')
distmat<-simil(tdata,method="Jaccard")
require(lattice)
levelplot(as.matrix(distmat),col.regions=grey(100:0/100))
# Alternative
data<-read.table('veg.csv',header=TRUE,sep=';')
data<-data[,-1]
tdata<-t(data)
install.packages('vegan')
library('vegan')
distmatrix<-vegdist(tdata,method="bray",binary=TRUE)
simmatrix=1-distmatrix
require(lattice)
levelplot(as.matrix(simmatrix),col.regions=grey(100:0/100))
```

Association matrix

```
data<-read.table('veg.csv',header=TRUE,sep=';')
labels<-data[,1]
data<-data[,-1]
tdata<-t(data)
install.packages('proxy')
library('proxy')
similmat<-simil(data,method="Jaccard")
require(lattice)
levelplot(as.matrix(similmat),col.regions=grey(100:0/100))
```

Scatterplot matrix

```
data<-read.table('envCachemistry.csv',header=TRUE,sep=';')
labels<-data[,1]
data<-data[,-1]
tdata=t(data)
pairs(tdata)
```

11.2 Classification

Agglomerative classification

Ward hierarchical clustering:

```
data<-read.table('envCachemistry.csv',header=TRUE,sep=';')
data<-data[,-1]
tdata=t(data)
d<-dist(tdata,method="euclidean") # alternative: maximum, manhattan, canberra,
    #binary, minkowski
fit<-hclust(d,method="ward") # alternative:single,complete,average,mcquitty,
    #median, centroid
# Display dendogram
plot(fit)
# Cut tree into 3 clusters
groups<-cutree(fit,k=3)
# Draw dendogram with red borders around the 3 clusters
rect.hclust(fit,k=3,border="red")
```

Partitioning (k-means)

```
data<-read.table('veg.csv',header=TRUE,sep=';')
data<-data[,-1]
tdata=t(data)
veg.kmean<-kmeans(tdata,3)
veg.kmean
veg.kmean$cluster
veg.kmean$centers
veg.kmean$withinss
veg.kmean$size
```

Fuzzy classification

```
install.packages('cluster')
# vegan may not be loaded
library('cluster')
x<-rbind(cbind(rnorm(10,0,0.5),rnorm(10,0,0.5)),cbind(rnorm(15,5,0.5),
    rnorm(15,5,0.5)),cbind(rnorm(3,3.2,0.5),rnorm(3,3.2,0.5)))
x
fannyx<-fanny(x, 2)
```

```
fannyx
fannyx$membership
```

11.3 Ordination

Correspondence analysis (CA)

```
library(vegan)
vegdata<-read.table('veg.It.class.txt')
tvegdata=t(vegdata)
colclass<-ncol(tvegdata)
class=tvegdata[,colclass]
cavegdata<-cca(tvegdata)
summary(cavegdata)
plot(cavegdata,display="sites")
ordihull(cavegdata,class,lty=1,col="red")
```

Ordination with plot and movable labels

```
vegdata<-read.table('veg.It.class.txt')
tvegdata=t(vegdata)
colclass<-ncol(tvegdata)
class=tvegdata[,colclass]
library(vegan)
cavegdata<-cca(tvegdata)
# scores in different colors and plotting symbols
pl<-ordipointlabel(cavegdata)
dev.off()
orditkplot(pl,mar=c(4,4,1,1)+.1,font=3)
```

Detrended correspondence analysis (DCA)

```
# see vegantutor
library(vegan)
vegdata<-read.table('veg.It.class.txt')
dcaveg<-decorana(vegdata)
summary(dcaveg)
```

Canonical correspondence analysis (CCA)

```
# see vegantutor
library(vegan)
vegdata<-read.table('veg.Ca.txt')
structuredata<-read.table('env.Ca.structure.txt')
attach(vegdata)
attach(structuredata)
tstructuredata=t(structuredata)
tvegdata=t(vegdata)
lsvol=tstructuredata[,9]
```

```
bulkd=tstructuredata[,1]
colclass=ncol(tstructuredata)
class=tstructuredata[,colclass]
caord<-cca(tvegdata)
ccaord1<-cca(tvegdata~lsvol)
ccaord2<-cca(tvegdata~lsvol+bulkd)
summary(caord)
summary(ccaord1)
summary(ccaord2)
par(mfrow=c(3,1))
plot(caord,display="sites",xlim=c(-5,5))
ordihull(caord,class,lty=1,col="red")
plot(ccaord1,display="sites",xlim=c(-5,5))
ordihull(ccaord1,class,lty=1,col="red")
plot(ccaord2,display="sites",xlim=c(-5,5))
ordihull(ccaord2,class,lty=1,col="red")
```

Principal component analysis (PCA)

```
# see vegantutor
library(vegan)
structuredata<-read.table('env.Ca.structure.txt')
attach(structuredata)
tstructuredata=t(structuredata)
attach(tstructuredata)
col.class<-ncol(tstructuredata)
class=tstructuredata[,col.class]
pcastructuredata<-rda(tstructuredata[,1:(ncol(tstructuredata)-1)])
pcastructuredata
summary(pcastructuredata)
scores(pcastructuredata,display='species')
plot(pcastructuredata,display="sites")
ordihull(pcastructuredata,class,lty=1,col="red")
```

Redundancy analysis (RDA)

```
# see vegantutor
library(vegan)
structuredata<-read.table('env.Ca.structure.txt')
vegdata<-read.table('veg.Ca.txt')
attach(structuredata)
attach(vegdata)
tstructuredata=t(structuredata)
tvegdata=t(vegdata)
attach(tstructuredata)
attach(tvegdata)
lsvol=tstructuredata[,9]
bulkd=tstructuredata[,1]
colclass<-ncol(tstructuredata)
class=tstructuredata[,colclass]
```

```
pcavegdata<-rda(tvegdata)
summary(pcavegdata)
rdavegdata1<-rda(tvegdata~lsvol)
summary(rdavegdata1)
rdavegdata2<-rda(tvegdata~lsvol+bulkd)
summary(rdavegdata2)
par(mfrow=c(3,1))
plot(pcavegdata,display="sites",main='PCA Veg',xlim=c(-5,5))
ordihull(pcavegdata,class,lty=1,col="red")
plot(rdavegdata1,display="sites",main='RDA Veg (veg~lsvol)',xlim=c(-5,5))
ordihull(rdavegdata1,class,lty=1,col="red")
plot(rdavegdata2,display="sites",main='RDA Veg (veg~lsvol+bulkd)',xlim=c(-5,5))
ordihull(rdavegdata2,class,lty=1,col="red")
```

Principal Response Curve (PRC)

1. Example: Ditches

https://rdrr.io/rforge/vegan/man/prc.html

```
# Chlorpyrifos experiment and experimental design: Pesticide treatment in ditches
#(replicated) and followed over from 4 weeks before to 24 weeks after exposure
library('vegan')
data(pyrifos)
week<-gl(11,12,labels=c(-4,-1,0.1,1,2,4,8,12,15,19,24))
dose<-factor(rep(c(0.1,0,0,0.9,0,44,6,0.1,44,0.9,0,6),11))
ditch<-gl(12,1,length=132)
# PRC
mod<-prc(pyrifos, dose, week)
mod # RDA
summary(mod) # PRC
logabu<-colSums(pyrifos)
plot(mod,select=logabu>100)
dev.off()
# Ditches are randomized, we have a time series, and are only interested in the first axis
ctrl<-how(plots=Plots(strata=ditch,type="free"),within=
    Within(type="series"),nperm=99)
anova(mod,permutations=ctrl,first=TRUE)
```

2. Example: Mice

```
data<-read.table('datafile.csv')
treatment<-as.factor(data$treatment)
time<-as.factor(data$time)
data.prc<-prc(response=data[,4:7],treatment=treatment,time=time)
summary(data.prc) ## Contains species scores
plot(data.prc)
# Arguments for plot(): scaling = 0 - 4
# 0 = none; y-axis only scaled on effect, not on species scores
# 1 = sites; 2 = species; 3 = symmetric primary and secondary y-axis coupled
```

Non-metric multidimensional scaling (NMDS)

1. MonoMDS:

see vegantutor

```
library(vegan)
vegdata<-read.table('veg.Ca.txt')
tvegdata=t(vegdata)
vare.dis<-vegdist(tvegdata)
vare.mds0<-monoMDS(vare.dis)
par(mfrow=c(2,1))
stressplot(vare.mds0,vare.dis)
ordiplot(vare.mds0,type="t")
```

2. MetaMDS

see vegantutor

```
library(vegan)
vegdata<-read.table('veg.Ca.txt')
tvegdata=t(vegdata)
vare.mds<-metaMDS(tvegdata,trace=FALSE)
vare.mds
par(mfrow=c(2,1))
stressplot(vare.mds,vare.dis)
plot(vare.mds,type="t")
```

Convex hulls

```
data <- matrix(stats::rnorm(2000), ncol = 2)
par(mfrow=c(1,1))
plot(data,cex=0.5)
hpts<-chull(data)
hpts<-c(hpts, hpts[1])
lines(data[hpts,])
# Alternative
# s. ordihull
```

11.4 Spider plots

```
# Based on: https://r-graph-gallery.com/spider-or-radar-chart.html

# Only do once
install.packages(fmsb)
# You might also have to install OpenMX

library(fmsb)

# Read data
data<-read.table('data_models.csv',header=TRUE,sep=',')
```

```
# Build a data frame with simple variable names
modeldata<-as.data.frame(cbind(data$tmax_s_ground,data$vol_deadwood,data$mean_soil_moist_agg
names(modeldata)<-c("tmax","dead","moist","reju")
# May like to use the attach command to make everything easier
# attach(modeldata)

# Get some descriptives
summary(modeldata)
# Add two lines at the beginning for min and max
modeldata<-rbind(c(35,500,0.1,20000),modeldata)
modeldata<-rbind(c(0,0,0,0),modeldata)

# Colors
colors_border=c(rgb(0.2,0.5,0.5,0.9),rgb(0.8,0.2,0.5,0.9))
colors_in=c(rgb(0.2,0.5,0.5,0.4),rgb(0.8,0.2,0.5,0.4))

# Plot radar chart
png('radar.png')
radarchart(modeldata[c(1:3,46),],axistype=1,
    #custom polygon
    pcol=colors_border,pfcol=colors_in,plwd=4,plty=1,
    #custom the grid
    cglcol="grey",cglty=1,axislabcol="grey",caxislabels=seq(0,20,5),cglwd=0.8,
    #custom labels
    vlcex=0.8)
dev.off()
```

11.5 Species area curves

```
# Read data
adu<-read.table('spec_adu.csv',header=TRUE,sep=',')
adu<-adu[,-1]
# Cut the column with the species labels
juv<-read.table('spec_juv.csv',header=TRUE,sep=',')
juv<-juv[,-1]
# Calculate species area curves
# install.packages('vegan')
library('vegan')
acuadu<-specaccum(adu,method="rarefaction",permutations=100,conditioned=TRUE,gamma="jack1")
acujuv<-specaccum(juv,method="rarefaction",permutations=100,conditioned=TRUE,gamma="jack1")
png('acuadu.png')
plot(acuadu,ylim=c(0,35),xlab='Observations',ylab='Species number')
dev.off()
svg('acuadu.svg')
plot(acuadu,ylim=c(0,35),xlab='Observations',ylab='Species number')
dev.off()
png('acujuv.png')
plot(acujuv,ylim=c(0,35),xlab='Observations',ylab='Species number')
dev.off()
svg('acujuv.svg')
```

```
plot(acujuv,ylim=c(0,35),xlab='Observations',ylab='Species number')
dev.off()
png('acu.png')
plot(acujuv,ylim=c(0,35),col='green',xlab='Observations',ylab='Species number')
lines(acuadu,col='red')
dev.off()
svg('acu.svg')
plot(acujuv,ylim=c(0,35),col='green',xlab='Observations',ylab='Species number')
lines(acuadu,col='red')
dev.off()
# Calculate the extrapolated number of species
# install.packages('fossil)
library('fossil')
chao1(adu)
# > [1] 49.64286
chao2(adu)
# > [1] 55.72222
jack1(adu)
# > [1] 51.99573
jack2(adu)
# > [1] 48.00426
chao1(juv)
# > [1] 37.33333
chao2(juv)
# > [1] 41
jack1(juv)
# > [1] 39.9882
jack2(juv)
# > [1] 37.98225
```

12. Functions

Minimal

```
myfunc<-function(x,y){
y<-x
return(y)
}
myfunc(1)
```

Simple

```
myfunc<-function(x,y){
y<-3*x^2+2*x+1
return(y)
}
myfunc(2)
```

Including start values

```
myfunc<-function(x=3,y=7){
z<-x+y
return(z)
}
myfunc()
myfunc(1,1)
```

Example t-statistic

```
tstatistic=function(x,y)
{
m=length(x)
n=length(y)
sp=sqrt(((m-1)*sd(x)^2+(n-1)*sd(y)^2)/(m+n-2)) # pooled standard deviation
t.stat=(mean(x)-mean(y))/(sp*sqrt(1/m+1/n))
return(t.stat)
}
```

13. Time series analysis

13.1 ARIMA models

```
data<-read.table('daphnia.time.series.csv',sep=';',header=T)
# Plot it
png('daphnia.time.series.png')
plot(data[,1],data[,2],xlab="Time",ylab="Daphnia population size")
lines(data[,1],data[,2],type="l")
dev.off()
data<-data[50:365,]
png('daphnia.time.series.notransient.png')
plot(data[,1],data[,2],xlab="Time",ylab="Daphnia population size")
lines(data[,1],data[,2],type="l")
dev.off()
# Fit linear model to check for linear trend
linfit<-lm(data[,2]~data[,1])
summary(linfit)
# Plot data including linear regression
png('daphnia.linear.trend.regression.png')
plot(data[,1],data[,2],xlab='Time [d]',ylab='Daphnia population size')
lines(data[,1],linfit$fitted.values)
dev.off()
```

13.2 Autocorrelation function (ACF) and partial ACF (PCAF)

```
png('daphnia.acf.pacf.png')
par(mfrow=c(2,1))
acf(data[,2])
```

```
pacf(data[,2])
dev.off()
# Difference the series and plot it
dpopsize<-diff(data[,2])
ndpopsize<-length(dpopsize)
png('daphnia.diff.png')
plot(1:ndpopsize,dpopsize,type='l')
dev.off()
# Check whether linear trend is removed using regression analysis
ndtransient<-length(data[,1])
lindfit<-lm(dpopsize~data[2:ndtransient,1])
summary(lindfit)
# Plot differentiated data including linear regression
png('daphnia.linear.trend.removed.regression.png')
plot(1:ndpopsize,dpopsize)
lines(data[2:ndtransient,1],lindfit$fitted.values)
dev.off()
# Check using ACF
# Plot ACF (autocorrelation function) and PACF (partial ACF) for difference
png('daphnia.acf.pacf.diff.png')
par(mfrow=c(2,1))
acf(dpopsize)
pacf(dpopsize)
dev.off()
# Fit the ARIMA model concerning the found out configuration
# Fit different models to compare
# Without MA
daphnia.ts.ar.sar.fit<-arima(data[,2],order=c(1,1,0),seasonal=list(order=c(1,1,0),
    period=4),include.mean=FALSE)
# With MA and period=4
daphnia.ts.ar.sar.ma.p4.fit<-arima(data[,2],order=c(1,1,1),seasonal=list(order=c(1,1,1),
    period=4),include.mean=FALSE)
# With MA and period=5
daphnia.ts.ar.sar.ma.p5.fit<-arima(data[,2],order=c(1,1,1),seasonal=list(order=c(1,1,1),
    period=5),include.mean=FALSE)
# Look at outputs
daphnia.ts.ar.sar.fit
daphnia.ts.ar.sar.ma.p4.fit
daphnia.ts.ar.sar.ma.p5.fit
# Use AIC to compare (slightly different from ARIMA-AIC
AIC(daphnia.ts.ar.sar.fit,daphnia.ts.ar.sar.ma.p4.fit,daphnia.ts.ar.sar.ma.p5.fit)
# Create predictions using the best model for 12 days ahead
daphnia.ts.ar.sar.ma.p4.pred<-predict(daphnia.ts.ar.sar.ma.p4.fit,n.ahead=12)
myprediction.pred<-c(data[length(data[,2]),2],daphnia.ts.ar.sar.ma.p4.pred$pred)
myprediction.se<-c(NA,daphnia.ts.ar.sar.ma.p4.pred$se)
index<-c(364:376)
png('daphnia.prediction.png')
par(mfrow=c(2,1))
plot(data[,1],data[,2],xlim=c(50,380),ylim=c(100,280),type='l',xlab='Time [d]',
    ylab='Daphnia population size')
lines(index,myprediction.pred,col="red")
```

```
lines(index,myprediction.pred+2*myprediction.se,col="blue")
lines(index,myprediction.pred-2*myprediction.se,col="blue")
plot(data[,1],data[,2],xlim=c(340,380),ylim=c(130,310),type='l',xlab='Time [d]',
    ylab='Daphnia population size')
lines(index,myprediction.pred,col="red")
lines(index,myprediction.pred+2*myprediction.se,col="blue")
lines(index,myprediction.pred-2*myprediction.se,col="blue")
legend('bottomright','black=observation, red=prediction, blue=prediction +- 2*s.e.')
dev.off()
```

13.3 Time series embedding

```
# Read data
data<-read.table('daphnia.time.series.csv',sep=';',header=T)
# Remove transient data at the beginning $\rightarrow$ cut of first 50 time steps
data<-data[50:365,]
install.packages('rgl')
# sudo apt-get install libglu1-mesa-dev
library('rgl')
# Define a function
myColorRamp<-function(colors,values){
v<-(values-min(values))/diff(range(values))
x<-colorRamp(colors)(v)
rgb(x[,1],x[,2],x[,3],maxColorValue=255)
}
# Set the variables for the function
n<-length(data[,1])
z<-data[1:n-1,1]
x<-data[1:n-1,2]
y<-data[2:n,2]
cols<-myColorRamp(c("red","blue"),z)
# Plot embedded time series
plot3d(x,y,z,col=cols,xlab='Popsize(t)',ylab='Popsize(t-1)',zlab='Day',type='l')
```

14. Python integration

Call Python from R

https://www.r-bloggers.com/run-python-from-r/

https://cran.r-project.org/web/packages/reticulate/vignettes/calling_python.html

```
# Install reticulate package
install.packages("reticulate")
# Load reticulate package
library(reticulate)
# Set Python version to use
use_python('/usr/bin/python3')
# Check Python
py_available()
```

```
# Get working directory
os<-import('os')
os$getcwd()
# List files
os<-import('os')
os$listdir()
# Install modules
conda_create('r-reticulate')
conda_install('r-reticulate','numpy')
# Load module
numpy<-import('numpy')
```

15. Julia integration

Call Julia from R

https://rdrr.io/cran/JuliaCall/f/README.md

```
# Install JuliaCall package
install.packages("JuliaCall")
# Load JuliaCall package
library(JuliaCall)
# Initial setup
julia<-julia_setup(JULIA_HOME="/home/basho/bin/julia-1.6.1/bin")
# -> error: segmentation fault
# -> get RCall working in Julia first?
```

16. MATLAB integration

Call MATLAB from R

https://mandymejia.com/2014/08/18/three-ways-to-use-matlab-from-r/

```
## Alternative: Use the system command
system("matlab -nodisplay -r 'stuff; to; do; in; matlab;'")

## Alternative: Use R.matlab
# load the R.matlab library and start MATLAB server
library(R.matlab)
Matlab$startServer()
matlab<-Matlab()
isOpen<-open(matlab)

## Alternative: Include MATLAB-code directly
# Set a variable in R and save in a csv file
x<-10
write.table(x,file='~/x.csv',sep=",",row.names=FALSE,col.names=FALSE)
# Make a vector where each element is a line of MATLAB code
# Matlab code reads in our variable x, creates two variables y and z,
```

```
# and write z in a csv file
matlab.lines<-c(
"x=csvread('~/x.csv')",
"y=20",
"z=x+y",
"csvwrite('~/z.csv',z)")
# Create a MATLAB script containing all the commands in matlab.lines
writeLines(matlab.lines,con="~/myscript.m")
# Run the MATLAB script
system("matlab -nodisplay -r \"run('~/myscript.m'); exit\"")
```

17. Use R in Jupyter

```
sudo apt-get install jupyter-client
sudo R
install.packages("devtools")
devtools::install_github("IRkernel/IRkernel")
IRkernel::installspec(user = FALSE)
```

18. Path analysis and Structural Equation models

18.1 Path analysis

```
# Only do once
# packages<-c("lavaan","semPlot","GGally")
# install.packages(packages)
# You might also have to install OpenMX

library(lavaan)
library(semPlot)
library(GGally)

# Read data
data<-read.table('data_models.csv',header=TRUE,sep=',')

# Build a data frame with simple variable names
modeldata<-as.data.frame(cbind(data$tmax_s_ground,data$vol_deadwood,data$mean_soil_moist_agg
names(modeldata)<-c("tmax","dead","moist","reju")
# You may want to use the attach command to make everything easier
# attach(modeldata)

# Get some descriptives
png('hist_tmax.png')
hist(modeldata$tmax)
# attached: hist(tmax)
dev.off()
png('hist_dead.png')
hist(modeldata$dead)
dev.off()
```

```
png('hist_moist.png')
hist(modeldata$moist)
dev.off()
png('hist_reju.png')
hist(modeldata$reju)
dev.off()
summary(modeldata)

# How to standardize a variable
# data$rej_total_std<-(data$rej_total-mean(data$rej_total))/sd(data$rej_total)

# How to normalize a variable
# data$rej_total_norm<-(data$rej_total-max(data$rej_total))/(max(data$rej_total)-min(data$r

# Build some simple pairwise models
# Specify a model
model<-'tmax~dead'
# Fit the model
fit<-cfa(model,data=modeldata)
# attached: fit<-cfa(model)
# View the results
summary(fit,fit.measures=TRUE,standardized=T,rsquare=T)
# Compare to simple linear model
summary(lm(model,data=modeldata))
# attached: summary(lm(model))

model<-'moist~tmax'
fit<-cfa(model,data=modeldata)
summary(fit,fit.measures=TRUE,standardized=T,rsquare=T)
summary(lm(model,data=modeldata))

model<-'reju~moist'
fit<-cfa(model,data=modeldata)
summary(fit,fit.measures=TRUE,standardized=T,rsquare=T)
summary(lm(model,data=modeldata))

# Build a Structural Equation Model (SEM)
model<-'
tmax~dead
moist~tmax
reju~moist
'
fit<-cfa(model,data=modeldata)
summary(fit,fit.measures=TRUE,standardized=T,rsquare=T)
# Plot
png('path1.png')
semPaths(fit,'std',layout='circle')
dev.off()
# Maybe easier to read if models get large
png('path2.png')
semPaths(fit,"std",layout='tree',edge.label.cex=.9,curvePivot=TRUE)
```

```
dev.off()

# What do the arrows and values between the independent variables represent?
png('cor.png')
ggcorr(modeldata,nbreaks=6,label=T,low="red3",high="green3",label_round=2,name="Correlation
dev.off()
```

19. apply-functions

19.1 lapply

```
# Example list of 3 Vectors
exmpllist <- list(vec1 = c(1:10), vec2 = c(11:20), vec3 = c(21:30))
# Perform operation on all vectors (=elements in list)
# 1. Alternative: Use a loop:
listmeans<-list()
for (i in 1:length(exmpllist)) {
listmeans[[i]] <- mean(exmpllist[[i]])
}
names(listmeans) <- names(exmpllist)
# 2. Alternative: Use lapply
listmeans<-apply(exmpllist,mean)
# 3. Alternative: Use lapply if data are a data frame
vec1<-c(1:10)
vec2<-c(11:20)
vec3<-c(21:30)
data<-as.data.frame(cbind(vec1,vec2,vec3))
means<-lapply(data, mean)
# 4. Alternative: Use a loop if data are a data frame
ncols<-dim(data)[2]
means<-c()
for (i in 1:ncols){
means<-c(means,mean(data[,i]))
}
```

Appendix

A1: Data sets used

Aktualisieren

3dspace.csv

beans.csv

beans1.csv

bohnen.txt

bohnen1.txt

cross.csv

daphnia.csv

daphnia.time.series.csv

daphnia1.csv

daphnia2.csv

daphnia3.csv

data.csv

datafile.csv

drc.csv

env.Ca.structure.txt

envCachemistry.csv

gmo1.csv

repANOVA.csv

v3.clp.kombi.72h.tsv

veg.Ca.txt

veg.csv

veg.It.class.txt

A2: Packages used

Aktualisieren

agricolae

caTools

cluster

drc

exactRankTests

funny

gdata

gdata

KScorrect

lattice

latticeExtra

lmtest

memisc

mutlcomp

nortest

NSM3

pgirmess

plyr

proxy

rcmdr -> opt

reshape

rgl

scatterplot3d

TeachingDemos

vegan

WriteXLS

A3: Online help and literature

- http://www.statmethods.net/
- Faes G. (2010): Einführung in R: Ein Kochbuch zur statistischen Datenanalyse mit R. 3. Auflage. Books on Demand, Norderstedt.
- About plots: see http://www.harding.edu/fmccown/r/