

Dina Research School:

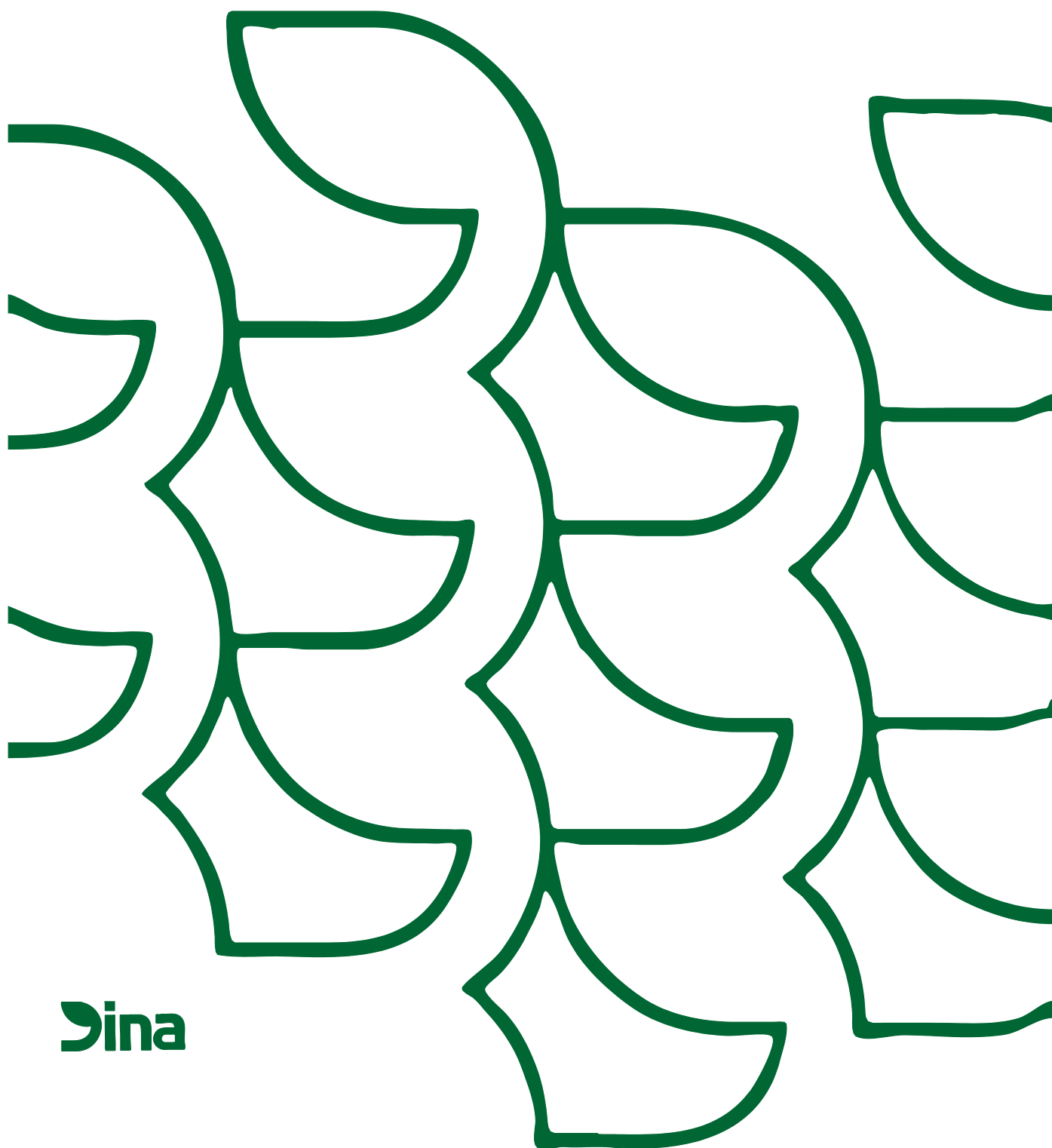
Workshop April 18-19, 2002

Dataserier, state-space models, and the Kalman filter

Koldkærgaard Landboskole, Skejby.

Edited by Erik Jørgensen & Rasmus Waagepetersen

Dina Notat No. 99 · April 2002



Dina

Dina Research School:

Workshop April 18-19, 2002

Dataserier, state-space models, and the Kalman filter

Koldkærgaard Landboskole, Skejby.

Edited by Erik Jørgensen & Rasmus Waagepetersen



Dina Notat No. 99 · April 2002

(Rev. May 2002)

This note is also available on www at URL:
<http://www.dina.dk/phd/w/w10/notat99.pdf>

Dina Research School
Department of Animal Science and Animal Health
Royal Veterinary and Agricultural University
Grønnegårdsvej 3, DK-1870
Frederiksberg C

Contents

| | |
|--|-----------|
| 1 Program | 1 |
| 1.1 Thursday, April 18 | 1 |
| 1.2 Friday, April 19 | 2 |
| 2 List of participants | 4 |
| 3 Exercises v. Søren Lundbye-Christensen & Rasmus Waagepetersen | 6 |
| A R - Programs for exercises Rasmus Waagepetersen | 12 |
| A.1 Initial | 12 |
| A.1.1 basic.R | 12 |
| A.2 Exercise 1 | 13 |
| A.2.1 growth.R | 13 |
| A.2.2 growth.play.R | 15 |
| A.2.3 growth.corrected.R | 17 |
| A.2.4 growth.Peter.R | 20 |
| A.3 Exercise 2 | 24 |
| A.3.1 simulation.R | 24 |
| A.4 Exercise 3 | 26 |
| A.4.1 simple.kalman.R | 26 |
| A.4.2 simple.kalman.run.R | 27 |
| A.5 Exercise 4 | 33 |
| A.5.1 likelihood.R | 33 |
| B Selected pages from: An introduction to R | 34 |

1 Program

1.1 Thursday, April 18

- 11.00 Arrival and accommodation.
- 12.00 Lunch

- 13.00 **Introduction and presentation of participants**
Erik Jørgensen, Dina Research School.

- 13.15 **Theory session I: Modelling of dataseries.**
Søren Lundbye-Christensen, Aalborg University.

- 14.00 Break.

- 14.15 **Theory session II: Estimation and prediction using the Kalman filter**
Søren Lundbye-Christensen, Aalborg University.

- 15.00 Coffee break.

- 15.30 **Computer exercises.**

- 17.00 **Theory session III: Approaches to monitoring.**
Susanne Gammelgaard Bøttcher, Aalborg University.

- 17.45 Dinner.

- 19.00 **Theory session IV: Recent trends in state-space modelling.**
Søren Lundbye-Christensen, Aalborg University

- 19.45 **Computer exercises (continued).**

- 21.45 Coffee and sandwich.

1.2 Friday, April 19

- 7.30 Breakfast.
- 8.30 **Discussion of computer exercises.**
- 9.00 **Case I: Applications of state-space models in data analysis.**
Flemming Skjøth, The Agricultural Advisory Centre.
- 9.45 Coffee Break.
- 10.00 **Case II: Monitoring water consumption of growing pigs.**
Thomas Nejsum Madsen, Danish Bacon and Meat Council
- 10.45 Break.
- 11.00 **Case III: Time series data - a nuisance or the whole story? personal experience**
Nic Friggins, Danish Institute of Agricultural Sciences, Foulum.
- 11.30 **Discussion and Closing.**
Erik Jørgensen, Dina Research School.
- 12.00 Lunch and departure.

2 List of participants

| Name | Email | Phone |
|--|--|-------------------------------|
| Stina West Andersen | swa@novonordisk.com | +45 36301191/+45 28259866 |
| Awuah Baffour | baffour.awuah@helsinki.fi | |
| Karsten Dalsgaard Bjerre | kdb@kvl.dk | +45 3528 2641 |
| Michael Brogaard | Michael.Brogaard@agrsci.dk | +45 58 34 42 |
| Carsten Enevoldsen | ce@kvl.dk | +45 35 28 28 30 |
| Leslie Foldager | Leslie.Foldager@agrsci.dk | +45 89 99 12 31 |
| Susanne Gammelgaard Bøttcher | alma@math.auc.dk | 96 35 99 83 |
| Anna Huda | ahu@mejeri.dk | +45 76 600 600 |
| Michael Karsten Høhle | hoehle@dina.kvl.dk | 35 28 30 84 |
| Steve Joyce | steve.joyce@resgeom.slu.se | +46 90 7866957 |
| Erik Jørgensen | Erik.Jorgensen@agrsci.dk | 89 99 19 00, dir: 89 99 12 30 |
| Anne Mette Kjeldsen | AMK@lr.dk | |
| Bjarke Klein | bjarke@statdem.sdu.dk | +45 6550 3607 |
| Inge Riis Korsgaard | IngeR.Korsgaard@agrsci.dk | 8999 1217 |
| Søren Lundbye-Christensen | s0ren@math.auc.dk | 96 35 88 60 |
| Thomas Nejsum Madsen | TNM@danskeslagterier.dk | +45 33 73 26 18 |
| Ben Marchant | ben.marchant@bbsrc.ac.uk | |
| Elise Norberg | Elise.Norberg@agrsci.dk | +4589991339 |
| Peter Løvendahl | Peter.Lovendahl@agrsci.dk | 89 99 13 38 |
| Peter Stoltze Petersen | pepe@kvl.dk | +45 35 28 36 33 |
| Horacio Javier Petruzzi | Horacio.Petruzzi@agrsci.dk | +45 89 99 13 85 |
| Peter Sestoft | sestoft@dina.kvl.dk | 35282334 |
| Thomas Algot Søllested | tas@kvl.dk | +45 35 28 30 84 |
| Nils Toft | nt@dina.kvl.dk | 35 28 30 24 |
| Rasmus Waagepetersen | rw@math.auc.dk | 96 35 88 76 |
| Søren Østergaard | Soren.Ostergaard@agrsci.dk | |

3 Exercises v. Søren Lundbye-Christensen & Rasmus Waagepetersen

1 Some comments on R

R is a free-ware version of Splus. A wide range of statistical methods are implemented in R and R is furthermore a very flexible programming language where new applications can easily be developed.

1.1 Entering commands

In your pc version commands can be executed via the command line. Longer sequences of commands or implementations of new functions can be written in a separate file. If the commands are stored in a file, `cmd.R` say, then the command line statement `source('cmd.R')` will execute the commands in `cmd.R` and load function code contained in `cmd.R`. You can also open `cmd.R` in your favourite text editor and use copy/paste to paste commands or sequences of commands into the command line.

1.2 Introduction and help

You may find it useful to run the commands in `basic.R` to learn more about some basic features of R. Help can be obtained by clicking the help-button. For help on a specific topic, say the function `scan`, you may alternatively just type `help(scan)` on the command line.

2 Exercises

In `c:/DinaForskorskole` you may find `.R` files with examples of how the exercises may be solved in R. For each exercise the relevant `.R` files are given in parantheses.

Exercise 1 (`growth.R,growth.play.R,growth.corrected.R`)

The purpose of this exercise is to familiarize you with programming in R and to show an implementation of the Kalman-filter and smoother for a certain state-space model for growth (for weight of pigs for example).

Let rate_t , lweight_t , and y_t denote the growth rate, the “true” weight on log scale, and the observed weight (at log scale) at time $t = 1, 2, \dots$. The model is given by the following equations:

$$\begin{aligned}\text{rate}_t &= \text{rate}_{t-1} + w_t \\ \text{lweight}_t &= \text{lweight}_{t-1} + \text{rate}_t \\ y_t &= \text{lweight}_t + v_t\end{aligned}$$

where $w_t, v_t, t = 1, 2, \dots$ are independent zero-mean Gaussian variables with variances σ_w^2 and σ_v^2 .

Let $\theta_t = (\text{rate}_t, \text{lweight}_t)^\top$ and $\tilde{w}_t = (w_t, v_t)^\top$. Then the matrix version of the model is:

$$\begin{aligned}\theta_t &= \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix} \theta_{t-1} + \tilde{w}_t \\ y_t &= [0 \ 1] \theta_t + v_t.\end{aligned}$$

Note that

$$\tilde{w}_t \sim N_2\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{bmatrix} \sigma_w^2 & \sigma_w^2 \\ \sigma_w^2 & \sigma_w^2 \end{bmatrix}\right).$$

1) Take a look of the R-functions `kalman.filter` and `kalman.smoother` in `growth.R`. The function `kalman.filter` takes as an argument a new observation and computes the predictive and filtered values for that observation (so it can be used for online-updating). The function `kalman.smoother` calculates the smoothed values based on a sequence of observations.

The functions contain five errors. Find these five errors and fix them (if you find more than five errors then the additional errors are unintentional!).

2) Load the corrected version of `growth.R` into R using `source()`. Play with the programme by running it on data supplied by yourself. Compare the one-step forecasts and their confidence intervals with the observations that you supply. What does the shape of the predictive bands tell you?

Also plot the smoothed values of the growth rate together with confidence bands.

3) The files `pig1.dat` and `pig2.dat` contains series of weights for two pigs. Can you spot the happy pig ?

Exercise 2 (`simulation.R`)

In this simulation exercise we consider a very simple state space model. The purpose of the exercise is to illustrate the effects of varying the variance parameters of the latent and observation process.

Consider the state space model

$$\begin{aligned}\theta_0 &= 0 \\ \theta_t &= \theta_{t-1} + w_t \\ y_t &= \theta_t + v_t.\end{aligned}$$

Let $\sigma_{w_t}^2 = \text{Var}(w_t)$ and $\sigma_{v_t}^2 = \text{Var}(v_t)$.

1) Simulate series y_1, \dots, y_{50} in the cases

- (i) $\sigma_{w_t}^2 = 0.02$ and $\sigma_{v_t}^2 = 1$.
- (ii) $\sigma_{w_t}^2 = 0.1$ and $\sigma_{v_t}^2 = 1$.
- (iii) $\sigma_{w_t}^2 = 1$ and $\sigma_{v_t}^2 = 1$.
- (iv) $\sigma_{w_t}^2 = 1$ and $\sigma_{v_t}^2 = 0.1$.

Write your own simulation programme or use the code in `simulation.R`.
Hint: `rnorm(n, 0, sqrt(0,02))` will produce a vector of n independent $N(0, 0.02)$'s.

Make a plot showing the latent process (as a solid line) and the observations (as dots).

2) Repeat 1) but for $t = 20$ let $\sigma_{w_t}^2$ be increased by a factor 100 and for $t = 30$ let $\sigma_{v_t}^2$ be increased by a factor 100.

Exercise 3 (`simple.kalman.R`, `simple.kalman.run.R`)

Implement the Kalman filter and smoother for the simple state space model of the previous exercise. The programme should return

- (i) The filtered values m_t and C_t .
 - (ii) The predictions f_t and Q_t .
 - (iii) The smoothed values \tilde{m}_t and \tilde{C}_t .
- 1) Run the filter and smoother on the simulated series from part 1) of the previous exercise and plot the true latent process together with the estimated latent process \tilde{m}_t and the confidence bands given by $\tilde{m}_t \pm 2\sqrt{\tilde{C}_t}$. Plot the simulated observations together with the predictions f_t and predictive intervals $f_t \pm 2\sqrt{Q_t}$.
 - 2) Run the filter and smoother on the simulated series from part 2) of the previous exercises *but* let (wrongfully) the variances be constant. Plot the output as in 1).
 - 3) Repeat 2) but using the correct variances.

If there is still time left you may try the last exercise on likelihood estimation using the Kalman filter - otherwise go for the late-night snack.

Exercise 4 (`likelihood.R`)

The command `library(ts)` loads the time series package and `data(lh)` makes the data set `lh` available. This dataserie consists of 48 measurements of lutein hormone in the blood of a woman measured with 10 minutes intervals. In this exercise we will try to estimate the two variance parameters in the simple state space model of Exercise 2 using these data. However, in this case we initialize with a vague prior on θ_0 : $\theta_0 \sim N(0, 100000)$.

- 1) Plot the data.
- 2) Make an R-programme that computes the likelihood of the two variance parameters using the Kalman filter.
- 3) Minimize the negated likelihood using e.g. the `nlm` procedure (`nlm` is an R-procedure for minimization of nonlinear functions - for further details consult the help page by typing `help(nlm)`).

4) Plot the likelihood on a grid of values of $\sigma_{w_t}^2$ and $\sigma_{v_t}^2$.

5) Condition on $\theta_0 = 0$ and compute the variance of y_t in terms of $\sigma_{v_t}^2$ and $\sigma_{w_t}^2$. Does the simple state space model seem to be a reasonable model for the lutein series ?

11.04.02 Søren Lundbye-Christensen and Rasmus Waagepetersen.

A R - Programs for exercises *Rasmus Waagepetersen*

A.1 Initial

A.1.1 basic.R

```
#In R the assignment operator is <-. Assignments within
#functions are local. A global assignment within a function may be
#performed using <<- (this is a bit dangerous programming style).

#create a vector x consisting of the numbers 1 2 and 3
x<-c(1,2,3)
x #x is printed...

#add 3 to all entries in x
z<-x+3
z

#multiply last entry in x by 7
x[3]<-x[3]*7
x

#create new vector by combining z and x
zx<-c(z,x)
zx

#create a 3 by 2 matrix with z in the first column and x in the second
A<-matrix(zx,3,2)

#take a look at A
A

#take a look at first row in A
A[1,]
```

```
#take a look at second column in A
A[,2]

#take a look at last 2 rows in A
A[c(2,3),]

#equivalent:
A[2:3,]

#create a list containing x, z, and A
l<-list(x=x,A=A,z=z)

#extract x from l
l$x

#compute square root of all entries in x
s<-sqrt(x)
s

#construct vector of 4 10's
ff<-rep(10,4)
ff
```

A.2 Exercise 1

A.2.1 growth.R

```
#Kalman filter and smoother for weight growth

#First we allocate the different variables required in the Kalman filter.
#We let maxt be an upper limit for the number of observations. We e.g.
#store the maxt R_t matrices on top of each other thus forming
#a 2*maxt x 2 matrix.
#(recall that the latent process is two-dimensional).
```

```
initvar<-function(maxt){
  a<-matrix(0,maxt,2)
  R<-matrix(0,2*maxt,2)
  f<-matrix(0,maxt,1)
  Q<-matrix(0,2*maxt,2)
  A<-matrix(0,maxt,2)
  e<-matrix(0,maxt,1)
  C<-matrix(0,2*maxt,2)
  m<-matrix(0,maxt,2)
  B<-matrix(0,2*maxt,2)
  mtilde<-matrix(0,maxt,2)
  Ctilde<-matrix(0,2*maxt,2)
  return("init ok")
}

#matrices for the system equations.
F<-matrix(c(0,1),1,2)
G<-matrix(c(1,1,0,1),2,2)

#R-code for Kalman filter. It takes as arguments a new observation y and
#the time t, and updates the value of a[t], R[t], etc. It returns y
#together with the prediction of y based on the previous observations.
#NB %*% is matrix multiplication and t(A) is transpose of a matrix A.
#solve(A) computes the inverse of A.

kalman.filter<-function(y,t,V,W,m0,C0,F,G){
  t2<-2*(t-1)+c(1,2) #Index for the 2 x 2 variables R, C,...

  if (t==1){
    a[t,]<-G%*%m0
    R[t2,]<-G%*%C0%*%t(G)+W
  } else {
    a[t,]<-G%*%m[t-1,]
    R[t2,]<-G%*%C[t2-c(2,2),]%*%t(G)+W
  }

  f[t]<-F%*%m[t,]
  Q[t]<-F%*%R[t2,]%*%t(G)+V

  A[t,]<-R[t2,]%*%t(F)%*%solve(Q[t])
  e[t]<-y-f[t]
  m[t,]<-a[t,]+A[t,]*f[t]
  C[t2,]<-R[t2,]-matrix(A[t,],2,1)%*%Q[t-1]%*%matrix(A[t,],1,2)
  #we need B matrices in Kalman smoother
  if (t>1)
    B[t2-c(2,2),]<-C[t2-c(2,2),]%*%G%*%solve(R[t2,])

  return(list(y=y,f=f[t],Q=Q[t]))
}
```

```
#computes smoothed values mtilde
kalman.smoother<-function(t){
  mtilde[t,]<-m[t-1,]
  Ctilde[2*(t-1)+c(1,2),]<-C[2*(t-1)+c(1,2),]
  for (i in (t-1):1){
    i2<-2*(i-1)+c(1,2)
    mtilde[i,]<-m[i,]+B[i2,]*%(mtilde[i+1,]-a[i+1,])
    Ctilde[i2,]<-C[i2,]+B[i2,]*%(Ctilde[i2+c(2,2),]-R[i2+c(2,2),])%*%t(B[i2,])
  }
}

#functions for plotting
kalman.plot.rate<-function(t){
  sd<-sqrt(Ctilde[2*c(1:t)-1,1])
  plot(c(1:t),mtilde[1:t,1],ylim=c(min(mtilde[1:t,1]-2*sd),max(mtilde[1:t,1]+2*sd)),
       type="l",lty=1)
  lines(c(1:t),mtilde[1:t,1]-2*sd,lty=2)
  lines(c(1:t),mtilde[1:t,1]+2*sd,lty=2)
}
kalman.plot.obs.and.pred<-function(y,t){
  sd<-sqrt(Q[1:t])
  plot(c(1:t),exp(y[1:t]),ylim=c(min(c(exp(y[1:t]),exp(f[1:t]-2*sd))),80))
  lines(c(1:t),exp(f[1:t]),lty=1)
  lines(c(1:t),exp(f[1:t]-2*sd),lty=2)
  lines(c(1:t),exp(f[1:t]+2*sd),lty=2)
}
```

A.2.2 growth.play.R

```
source("c:/DinaForsknerskole/growth.corrected.R")

#initialize the various variables.

#variances and variance matrix for innovation and noise process

sigma2w<-0.000001

W<-matrix(rep(sigma2w,4),2,2) #innovation covariance matrix for latent process

V<-0.001 #noise variance

#mean of growth rate and log(weight) - i.e. population mean growth rate at
#time 0 is 0.01 and population mean log weight at time 0 is log(40).

m0<-c(0.01,log(40))

#prior population covariance matrix for growth rate and log(weight)

C0<-matrix(c(0.00005,0,0,2),2,2)
```

```
initvar(100)

#first observation is 40.278
kalman.filter(log(40.278),1,V,W,m0,C0,F,G)
#next observation is 41.59812
kalman.filter(log(41.59812),2,V,W,m0,C0,F,G)
#next observation is 40.446
kalman.filter(log(40.446),3,V,W,m0,C0,F,G)
#plot the 3 observations and their predictions
par(mfrow=c(1,2)) #two plots on one figure
y<-log(c(40.278,41.59812,40.446))
kalman.plot.obs.and.pred(y,3)
#compute smoothed values
kalman.smother(3)
#plot the estimated rate process
kalman.plot.rate(3)

#Now proceed with data from pig1.dat and pig2.dat
y<-log(scan("c:/DinaForskerskole/pig1.dat"))
initvar(100)
for (i in 1:20)
  kalman.filter(y[i],i,V,W,m0,C0,F,G)
kalman.smother(20)
kalman.plot.obs.and.pred(y,20)
kalman.plot.rate(20)
```

```
y<-log(scan("c:/DinaForskerskole/pig2.dat"))
initvar(100)
for (i in 1:20)
  kalman.filter(y[i],i,V,W,m0,C0,F,G)
kalman.smoother(20)
kalman.plot.obs.and.pred(y,20)
kalman.plot.rate(20)
```

A.2.3 growth.corrected.R

```
#Kalman filter and smoother for weight growth

#First we allocate the different variables required in the Kalman filter.
#We let maxt be an upper limit for the number of observations. We e.g.
#store the maxt R_t matrices on top of each other thus forming
#a 2*maxt x 2 matrix.
#(recall that the latent process is two-dimensional).

initvar<-function(maxt){
  a<-matrix(0,maxt,2)
  R<-matrix(0,2*maxt,2)
  f<-matrix(0,maxt,1)
```

```
Q<-matrix(0,2*maxt,2)
A<-matrix(0,maxt,2)
e<-matrix(0,maxt,1)
C<-matrix(0,2*maxt,2)
m<-matrix(0,maxt,2)
B<-matrix(0,2*maxt,2)
mtilde<-matrix(0,maxt,2)
Ctilde<-matrix(0,2*maxt,2)
return("init ok")
}

#matrices for the system equations.
F<-matrix(c(0,1),1,2)
G<-matrix(c(1,1,0,1),2,2)

#R-code for Kalman filter. It takes as arguments a new observation y and
#the time t, and updates the value of a[t], R[t], etc. It returns y
#together with the prediction of y based on the previous observations.
#NB %% is matrix multiplication and t(A) is transpose of a matrix A.

kalman.filter<-function(y,t,V,W,m0,C0,F,G){
  t2<-2*(t-1)+c(1,2) #Index for the 2 x 2 variables R, C,...

  if (t==1){
    a[t,]<-G%%m0
```

```
R[t2, ]<-G**C0**t(G)+W
} else {
  a[t, ]<-G**m[t-1, ]
  R[t2, ]<-G**C[t2-c(2,2), ]**t(G)+W
}

f[t]<-F**a[t, ]
Q[t]<-F**R[t2, ]**t(F)+V

A[t, ]<-R[t2, ]**t(F)**solve(Q[t])
e[t]<-y-f[t]
m[t, ]<-a[t, ]+A[t, ]*e[t]
C[t2, ]<-R[t2, ]-matrix(A[t, ],2,1)**Q[t]**matrix(A[t, ],1,2)
#we need B matrices in Kalman smoother
if (t>1)
  B[t2-c(2,2), ]<-C[t2-c(2,2), ]**t(G)**solve(R[t2, ])

return(list(y=y, f=f[t], Q=Q[t]))
}

#computes smoothed values mtilde
kalman.smother<-function(t){
  mtilde[t, ]<-m[t, ]
  Ctilde[2*(t-1)+c(1,2), ]<-C[2*(t-1)+c(1,2), ]
  for (i in (t-1):1){
    i2<-2*(i-1)+c(1,2)
```

```
    mtilde[i,]<-m[i,]+B[i2,]*%(mtilde[i+1,]-a[i+1,])
    Ctilde[i2,]<-C[i2,]+B[i2,]*%(Ctilde[i2+c(2,2),]-R[i2+c(2,2),])*%t(B[i2,])
  }
}

#functions for plotting
kalman.plot.rate<-function(t){
  sd<-sqrt(Ctilde[2*c(1:t)-1,1])
  plot(c(1:t),mtilde[1:t,1],ylim=c(min(mtilde[1:t,1]-2*sd),max(mtilde[1:t,1]+2*sd)),
       type="l",lty=1)
  lines(c(1:t),mtilde[1:t,1]-2*sd,lty=2)
  lines(c(1:t),mtilde[1:t,1]+2*sd,lty=2)
}

kalman.plot.obs.and.pred<-function(y,t){
  sd<-sqrt(Q[1:t])
  plot(c(1:t),exp(y[1:t]),ylim=c(min(c(exp(y[1:t]),exp(f[1:t]-2*sd))),80))
  lines(c(1:t),exp(f[1:t]),lty=1)
  lines(c(1:t),exp(f[1:t]-2*sd),lty=2)
  lines(c(1:t),exp(f[1:t]+2*sd),lty=2)
}
```

A.2.4 growth.Peter.R

Contribution from Peter Sestofte using three dimensional arrays

```
#Kalman filter and smoother for weight growth (here using arrays)

#First we allocate the different variables required in the Kalman
#filter. We let maxt be an upper limit for the number of
```

```
#observations. We store the maxt R_t matrices on top of each other
#using a three-dimensional array.

initvar<-function(maxt){
  a<-matrix(0,maxt,2)
  R<-array(0, c(maxt,2,2))
  f<-matrix(0,maxt,1)
  Q<-matrix(0,maxt,2)
  A<-matrix(0,maxt,2)
  e<-matrix(0,maxt,1)
  C<-array(0, c(maxt,2,2))
  m<-matrix(0,maxt,2)
  B<-array(0, c(maxt,2,2))
  mtilde<-matrix(0,maxt,2)
  Ctilde<-array(0, c(maxt,2,2))
  return("init ok")
}

#matrices for the system equations.
F<-matrix(c(0,1),1,2)
G<-matrix(c(1,1,0,1),2,2)

#R-code for Kalman filter. It takes as arguments a new observation y and
#the time t, and updates the value of a[t], R[t], etc. It returns y
#together with the prediction of y based on the previous observations.
```

#NB `%%` is matrix multiplication and `t(A)` is transpose of a matrix A.

```
kalman.filter<-function(y,t,V,W,m0,C0,F,G){
  if (t==1){
    a[t,]<-G%%m0
    R[t,,]<-G%%C0%%t(G)+W
  } else {
    a[t,]<-G%%m[t-1,]
    R[t,,]<-G%%C[t-1,,]%%t(G)+W
  }

  f[t]<-F%%a[t,]
  Q[t]<-F%%R[t,,]%%t(F)+V

  A[t,]<-R[t,,]%%t(F)%%solve(Q[t])
  e[t]<-y-f[t]
  m[t,]<-a[t,]+A[t,]*e[t]
  C[t,,]<-R[t,,]-matrix(A[t,],2,1)%%Q[t]%%matrix(A[t,],1,2)
  #we need B matrices in Kalman smoother
  if (t>1)
    B[t-1,,]<-C[t-1,,]%%t(G)%%solve(R[t,,])

  return(list(y=y,f=f[t],Q=Q[t]))
}

#computes smoothed values mtilde
```

```
kalman.smoother<-function(t){
  mtilde[t,]<-m[t,]
  Ctilde[t,,]<-C[t,,]
  for (i in (t-1):1){
    mtilde[i,]<-m[i,]+B[i,,]%*(mtilde[i+1,]-a[i+1,])
    Ctilde[i,,]<-C[i,,]+B[i,,]%*(Ctilde[i+1,,]-R[i+1,,])%*t(B[i,,])
  }
}

#functions for plotting
kalman.plot.rate<-function(t){
  sd<-sqrt(Ctilde[1:t,1,1])
  plot(c(1:t),mtilde[1:t,1],ylim=c(min(mtilde[1:t,1]-2*sd),max(mtilde[1:t,1]+2*sd)),
       type="l",lty=1)
  lines(c(1:t),mtilde[1:t,1]-2*sd,lty=2)
  lines(c(1:t),mtilde[1:t,1]+2*sd,lty=2)
}

kalman.plot.obs.and.pred<-function(y,t){
  sd<-sqrt(Q[1:t])
  plot(c(1:t),exp(y[1:t]),ylim=c(min(c(exp(y[1:t]),exp(f[1:t]-2*sd))),80))
  lines(c(1:t),exp(f[1:t]),lty=1)
  lines(c(1:t),exp(f[1:t]-2*sd),lty=2)
  lines(c(1:t),exp(f[1:t]+2*sd),lty=2)
}
```

A.3 Exercise 2

A.3.1 simulation.R

#for convenience we write an R-function for simulation of the simple
#state space model.

```
sim<-function(V,W,size){  
  #First simulate theta process  
  theta<-rep(0,size+1)  
  theta[1]<-0 #theta_0 is first entry  
  theta[2:(size+1)]<-rnorm(size,mean=0,sd=sqrt(W))  
  theta<-cumsum(theta) #we sum up the w_t's  
  #now obtain y by adding noise  
  y<-theta+rnorm(size+1,mean=0,sd=sqrt(V)) #add noise  
  plot(y)  
  lines(theta)  
  return(list(y=y,theta=theta))  
}
```

```
par(mfrow=c(2,2))  
#(i)  
sim1<-sim(1,0.02,50)  
#(ii)  
sim2<-sim(1,0.1,50)  
#(iii)  
sim3<-sim(1,1,50)  
#(iv)  
sim4<-sim(0.1,1,50)
```

```
#new function for part 2) of exercise
simb<-function(V,W){
  #First simulate theta process
  theta<-rep(0,51)
  theta[1]<-0 #theta_0 is first entry
  theta[2:51]<-rnorm(50,mean=0,sd=sqrt(W))
  #increased variance for latent variable no. 20
  theta[21]<-rnorm(1,mean=0,sd=10*sqrt(W))
  theta<-cumsum(theta) #we sum up the w_t's
  #now obtain y by adding noise
  y<-rep(0,51)
  y[-31]<-theta[-31]+rnorm(50,mean=0,sd=sqrt(V))
  #increased variance for observation no. 30
  y[31]<-theta[31]+rnorm(1,mean=0,sd=10*sqrt(V))
  plot(y)
  lines(theta)
  return(list(y=y,theta=theta))
}

par(mfrow=c(2,2))
#(ib)
sim1b<-simb(1,0.02)
#(iib)
sim2b<-simb(1,0.1)
```

```
 #(iib)
 sim3b<-simb(1,1)
 #(ivb)
 sim4b<-simb(0.1,1)
```

A.4 Exercise 3

A.4.1 simple.kalman.R

```
#Kalman filter for simple state space model
#matrices for the system equations just consist of a 1.
#V and W contains the variances for y and theta
#in the procedure kalman.simple y is a series of observations.
kalman.simple<-function(y,V,W,m0,C0){

  maxt<-length(y)
  a<-rep(0,maxt)
  R<-rep(0,maxt)
  f<-rep(0,maxt)
  Q<-rep(0,maxt)
  A<-rep(0,maxt)
  e<-rep(0,maxt)
  C<-rep(0,maxt)
  m<-rep(0,maxt)
  B<-rep(0,maxt)

  #kalman filter
  for (t in 1:maxt){
    if (t==1){
      a[t]<-m0
      R[t]<-C0+W[t]
    } else {
      a[t]<-m[t-1]
      R[t]<-C[t-1]+W[t]
    }
    f[t]<-a[t]
    Q[t]<-R[t]+V[t]
    A[t]<-R[t]/Q[t]
    e[t]<-y[t]-f[t]
    m[t]<-a[t]+A[t]*e[t]
    C[t]<-R[t]-Q[t]*A[t]^2
    B[t-1]<-C[t-1]/R[t]
  }
}
```

```
#kalman smoother
mtilde<-rep(0,maxt)
Ctilde<-rep(0,maxt)
mtilde[maxt]<-m[maxt]
Ctilde[maxt]<-C[maxt]
for (i in (t-1):1){
  mtilde[i]<-m[i]+B[i]*(mtilde[i+1]-a[i+1])
  Ctilde[i]<-C[i]+(Ctilde[i+1]-R[i+1])*B[i]^2
}
return(list(y=y,f=f[1:maxt],Q=Q[1:maxt],m=m[1:maxt],C=C[1:maxt],
           mtilde=mtilde[1:maxt],Ctilde=Ctilde[1:maxt]))
}

kalman.plot.latent<-function(theta,mtilde,Ctilde,t){
  ymin<-min(theta[1:t],mtilde[1:t]-2*sqrt(Ctilde[1:t]))
  ymax<-max(theta[1:t],mtilde[1:t]+2*sqrt(Ctilde[1:t]))
  plot(c(1:t),theta[1:t],ylim=c(ymin,ymax),type="l",lty=3)
  lines(c(1:t),mtilde[1:t],lty=1)
  lines(c(1:t),mtilde[1:t]-2*sqrt(Ctilde[1:t]),lty=2)
  lines(c(1:t),mtilde[1:t]+2*sqrt(Ctilde[1:t]),lty=2)
}

kalman.plot.obs.and.pred.simple<-function(y,f,Q,t){
  ymin<-min(y[1:t],f[1:t]-2*sqrt(Q[1:t]))
  ymax<-max(y[1:t],f[1:t]+2*sqrt(Q[1:t]))
  plot(c(1:t),y[1:t],ylim=c(ymin,ymax))
  lines(c(1:t),f[1:t],lty=1)
  lines(c(1:t),f[1:t]-2*sqrt(Q[1:t]),lty=2)
  lines(c(1:t),f[1:t]+2*sqrt(Q[1:t]),lty=2)
}
```

A.4.2 simple.kalman.run.R

```
source("c:/DinaForskerskole/simple.kalman.R")

par(mfrow=c(2,2))

#(i)

y<-sim1$y[2:51]

theta<-sim1$theta[2:51]

t<-0

V<-rep(1,50)

W<-rep(0.02,50)

out<-kalman.simple(y,V,W,0,0)
```

```
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
```

```
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)
```

```
 #(ii)
```

```
y<-sim2$y[2:51]
```

```
theta<-sim2$theta[2:51]
```

```
t<-0
```

```
V<-rep(1,50)
```

```
W<-rep(0.1,50)
```

```
out<-kalman.simple(y,V,W,0,0)
```

```
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
```

```
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)
```

```
 #(iii)
```

```
y<-sim3$y[2:51]
```

```
theta<-sim3$theta[2:51]
```

```
t<-0
```

```
V<-rep(1,50)
```

```
W<-rep(1,50)
```

```
out<-kalman.simple(y,V,W,0,0)
```

```
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
```

```
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)
```

```
 #(iv)
```

```
y<-sim4$y[2:51]
```

```
theta<-sim4$theta[2:51]
```

```
t<-0
V<-rep(0.1,50)
W<-rep(1,50)
out<-kalman.simple(y,V,W,0,0)
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)

# simulations from part 2) but with wrong variances in filter and smoother
#(ib)
y<-sim1b$y[2:51]
theta<-sim1b$theta[2:51]
t<-0
V<-rep(1,50)
W<-rep(0.02,50)
out<-kalman.simple(y,V,W,0,0)
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)

#(iib)
y<-sim2b$y[2:51]
theta<-sim2b$theta[2:51]
t<-0
V<-rep(1,50)
W<-rep(0.1,50)
out<-kalman.simple(y,V,W,0,0)
```

```
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)

#(iiib)
y<-sim3b$y[2:51]
theta<-sim3b$theta[2:51]
t<-0
V<-rep(1,50)
W<-rep(1,50)
out<-kalman.simple(y,V,W,0,0)
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)

#(ivb)
y<-sim4b$y[2:51]
theta<-sim4b$theta[2:51]
t<-0
V<-rep(0.1,50)
W<-rep(1,50)
out<-kalman.simple(y,V,W,0,0)
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)

#Now simulations from part 2) with right variances
```

```
 #(ib)
 y<-sim1b$y[2:51]
 theta<-sim1b$theta[2:51]
 t<-0
 V<-rep(1,50)
 W<-rep(0.02,50)
 W[20]<-W[20]*100
 V[30]<-V[30]*100
 out<-kalman.simple(y,V,W,0,0)
 kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
 kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)
```

```
 #(iib)
 y<-sim2b$y[2:51]
 theta<-sim2b$theta[2:51]
 t<-0
 V<-rep(1,50)
 W<-rep(0.1,50)
 W[20]<-W[20]*100
 V[30]<-V[30]*100
 out<-kalman.simple(y,V,W,0,0)
 kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
 kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)
```

```
 #(iiib)
```

```
y<-sim3b$y[2:51]
theta<-sim3b$theta[2:51]
t<-0
V<-rep(1,50)
W<-rep(1,50)
W[20]<-W[20]*100
V[30]<-V[30]*100
out<-kalman.simple(y,V,W,0,0)
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)

#(ivb)
y<-sim4b$y[2:51]
theta<-sim4b$theta[2:51]
t<-0
V<-rep(0.1,50)
W<-rep(1,50)
W[20]<-W[20]*100
V[30]<-V[30]*100
out<-kalman.simple(y,V,W,0,0)
kalman.plot.obs.and.pred.simple(out$y,out$f,out$Q,50)
kalman.plot.latent(theta,out$mtilde,out$Ctilde,50)
```

A.5 Exercise 4

A.5.1 likelihood.R

```
#computation of (negated) likelihood under simple state space model
#using Kalman filter.

source("c:/DinaForskerskole/simple.kalman.R")
library(ts)
data(lh)

plot(lh)

likelihood<-function(param){

  maxt<-length(lh)

  V<-rep(param[1],maxt)
  W<-rep(param[2],maxt)

  #initialisation
  m0<-0
  C0<-100000

  temp<-kalman.simple(lh,V,W,m0,C0)
  f<-temp$f
  Q<-temp$Q

  likelihood<-sum(log(Q))+sum((lh-f)^2/Q)

  return(likelihood)
}

luteinfit<-nlm(likelihood,c(0.03,0.3))
estimate<-luteinfit$estimate
estimate

#plot likelihood
plot.like<-function(param){
  v<-c(1:10)*2*param[1]/10
  w<-c(1:10)*2*param[2]/10
  grid<-expand.grid(v,w)
  like<-rep(0,dim(grid)[1])
  for (i in 1:dim(grid)[1])
    like[i]<-likelihood(as.numeric(grid[i,]))
  like<-matrix(like,10,10)
  #first column corresponds to first value of v etc.
  contour(v,w,like)
  #y-axis:v x-axis:w
  return(like)
}
plot.like(estimate)
```

B Selected pages from: An introduction to R

An Introduction to R

Notes on R: A Programming Environment for Data Analysis and Graphics
Version 1.4.1 (2002-01-30)

**W. N. Venables, D. M. Smith
and the R Development Core Team**

Copyright © 1990 W. N. Venables
Copyright © 1992 W. N. Venables & D. M. Smith
Copyright © 1997 R. Gentleman & R. Ihaka
Copyright © 1997, 1998 M. Maechler
Copyright © 1999–2001 R Development Core Team

Permission is granted to make and distribute verbatim copies of this manual provided the copyright notice and this permission notice are preserved on all copies.

Permission is granted to copy and distribute modified versions of this manual under the conditions for verbatim copying, provided that the entire resulting derived work is distributed under the terms of a permission notice identical to this one.

Permission is granted to copy and distribute translations of this manual into another language, under the above conditions for modified versions, except that this permission notice may be stated in a translation approved by the R Development Core Team.

Table of Contents

| | |
|---|-----------|
| Preface | 1 |
| 1 Introduction and preliminaries | 2 |
| 1.1 The R environment | 2 |
| 1.2 Related software and documentation | 2 |
| 1.3 R and statistics | 2 |
| 1.4 R and the window system | 3 |
| 1.5 Using R interactively | 3 |
| 1.6 An introductory session | 4 |
| 1.7 Getting help with functions and features | 4 |
| 1.8 R commands, case sensitivity, etc. | 5 |
| 1.9 Recall and correction of previous commands | 5 |
| 1.10 Executing commands from or diverting output to a file | 6 |
| 1.11 Data permanency and removing objects | 6 |
| 2 Simple manipulations; numbers and vectors .. | 7 |
| 2.1 Vectors and assignment | 7 |
| 2.2 Vector arithmetic | 8 |
| 2.3 Generating regular sequences | 8 |
| 2.4 Logical vectors | 9 |
| 2.5 Missing values | 10 |
| 2.6 Character vectors | 10 |
| 2.7 Index vectors; selecting and modifying subsets of a data set | 11 |
| 2.8 Other types of objects | 12 |
| 3 Objects, their modes and attributes | 13 |
| 3.1 Intrinsic attributes: mode and length | 13 |
| 3.2 Changing the length of an object | 14 |
| 3.3 Getting and setting attributes | 14 |
| 3.4 The class of an object | 15 |
| 4 Ordered and unordered factors | 16 |
| 4.1 A specific example | 16 |
| 4.2 The function <code>tapply()</code> and ragged arrays | 16 |
| 4.3 Ordered factors | 17 |

| | | |
|----------|---|-----------|
| 5 | Arrays and matrices | 19 |
| 5.1 | Arrays | 19 |
| 5.2 | Array indexing. Subsections of an array | 19 |
| 5.3 | Index arrays | 20 |
| 5.4 | The <code>array()</code> function | 21 |
| | 5.4.1 Mixed vector and array arithmetic. The recycling rule | 21 |
| 5.5 | The outer product of two arrays | 22 |
| 5.6 | Generalized transpose of an array | 22 |
| 5.7 | Matrix facilities | 23 |
| | 5.7.1 Matrix multiplication | 23 |
| | 5.7.2 Linear equations and inversion | 24 |
| | 5.7.3 Eigenvalues and eigenvectors | 24 |
| | 5.7.4 Singular value decomposition and determinants .. | 24 |
| | 5.7.5 Least squares fitting and the QR decomposition .. | 25 |
| 5.8 | Forming partitioned matrices, <code>cbind()</code> and <code>rbind()</code> | 25 |
| 5.9 | The concatenation function, <code>c()</code> , with arrays | 26 |
| 5.10 | Frequency tables from factors | 26 |
| 6 | Lists and data frames | 27 |
| 6.1 | Lists | 27 |
| 6.2 | Constructing and modifying lists | 28 |
| | 6.2.1 Concatenating lists | 28 |
| 6.3 | Data frames | 28 |
| | 6.3.1 Making data frames | 28 |
| | 6.3.2 <code>attach()</code> and <code>detach()</code> | 29 |
| | 6.3.3 Working with data frames | 29 |
| | 6.3.4 Attaching arbitrary lists | 30 |
| | 6.3.5 Managing the search path | 30 |
| 7 | Reading data from files | 31 |
| 7.1 | The <code>read.table()</code> function | 31 |
| 7.2 | The <code>scan()</code> function | 32 |
| 7.3 | Accessing builtin datasets | 32 |
| | 7.3.1 Loading data from other R packages | 33 |
| 7.4 | Editing data | 33 |
| 8 | Probability distributions | 34 |
| 8.1 | R as a set of statistical tables | 34 |
| 8.2 | Examining the distribution of a set of data | 35 |
| 8.3 | One- and two-sample tests | 37 |

| | | |
|-----------|--|-----------|
| 9 | More language features. Loops and conditional execution | 41 |
| 9.1 | Grouped expressions | 41 |
| 9.2 | Control statements | 41 |
| 9.2.1 | Conditional execution: <code>if</code> statements | 41 |
| 9.2.2 | Repetitive execution: <code>for</code> loops, <code>repeat</code> and <code>while</code> | 41 |
| 10 | Writing your own functions | 43 |
| 10.1 | Simple examples | 43 |
| 10.2 | Defining new binary operators | 44 |
| 10.3 | Named arguments and defaults | 44 |
| 10.4 | The ‘ <code>...</code> ’ argument | 45 |
| 10.5 | Assignments within functions | 45 |
| 10.6 | More advanced examples | 45 |
| 10.6.1 | Efficiency factors in block designs | 45 |
| 10.6.2 | Dropping all names in a printed array | 46 |
| 10.6.3 | Recursive numerical integration | 47 |
| 10.7 | Scope | 47 |
| 10.8 | Customizing the environment | 49 |
| 10.9 | Classes, generic functions and object orientation | 50 |
| 11 | Statistical models in R | 52 |
| 11.1 | Defining statistical models; formulae | 52 |
| 11.1.1 | Contrasts | 54 |
| 11.2 | Linear models | 55 |
| 11.3 | Generic functions for extracting model information | 55 |
| 11.4 | Analysis of variance and model comparison | 56 |
| 11.4.1 | ANOVA tables | 57 |
| 11.5 | Updating fitted models | 57 |
| 11.6 | Generalized linear models | 58 |
| 11.6.1 | Families | 58 |
| 11.6.2 | The <code>glm()</code> function | 59 |
| 11.7 | Nonlinear least squares and maximum likelihood models | 62 |
| 11.7.1 | Least squares | 62 |
| 11.7.2 | Maximum likelihood | 63 |
| 11.8 | Some non-standard models | 64 |

| | | |
|-----------|---|-----------|
| 12 | Graphical procedures | 65 |
| 12.1 | High-level plotting commands | 65 |
| 12.1.1 | The <code>plot()</code> function | 65 |
| 12.1.2 | Displaying multivariate data | 66 |
| 12.1.3 | Display graphics | 66 |
| 12.1.4 | Arguments to high-level plotting functions | 67 |
| 12.2 | Low-level plotting commands | 68 |
| 12.2.1 | Mathematical annotation | 70 |
| 12.2.2 | Hershey vector fonts | 70 |
| 12.3 | Interacting with graphics | 70 |
| 12.4 | Using graphics parameters | 71 |
| 12.4.1 | Permanent changes: The <code>par()</code> function | 71 |
| 12.4.2 | Temporary changes: Arguments to graphics functions | 72 |
| 12.5 | Graphics parameters list | 72 |
| 12.5.1 | Graphical elements | 72 |
| 12.5.2 | Axes and tick marks | 73 |
| 12.5.3 | Figure margins | 74 |
| 12.5.4 | Multiple figure environment | 75 |
| 12.6 | Device drivers | 77 |
| 12.6.1 | PostScript diagrams for typeset documents | 77 |
| 12.6.2 | Multiple graphics devices | 78 |
| 12.7 | Dynamic graphics | 79 |
| A | A sample session | 80 |
| B | Invoking R | 84 |
| B.1 | Invoking R under UNIX | 84 |
| B.2 | Invoking R under Windows | 87 |
| B.3 | Invoking R on a Macintosh | 89 |
| C | The command line editor | 92 |
| C.1 | Preliminaries | 92 |
| C.2 | Editing actions | 92 |
| C.3 | Command line editor summary | 92 |
| D | Function and variable index | 94 |
| E | Concept index | 97 |
| F | References | 99 |

Appendix A A sample session

The following session is intended to introduce to you some features of the R environment by using them. Many features of the system will be unfamiliar and puzzling at first, but this puzzlement will soon disappear. This is written for the UNIX user. Those using Windows or Macintosh will need to adapt the discussion slightly.

Login, start your windowing system. You should also have the file ‘`morley.tab`’ in your working directory. If not, seek the local expert (or get it yourself from the ‘`base/data`’ subdirectory of the default R library tree). If you have, proceed.

`$ R` Start R as appropriate for your platform.

The R program begins, with a banner.

(Within R, the prompt on the left hand side will not be shown to avoid confusion.)

`help.start()`

Start the HTML interface to on-line help (using a web browser available at your machine). You should briefly explore the features of this facility with the mouse.

Iconify the help window and move on to the next part.

`x <- rnorm(50)`

`y <- rnorm(x)`

Generate two pseudo-random normal vectors of x - and y -coordinates.

`plot(x, y)`

Plot the points in the plane. A graphics window will appear automatically.

`ls()`

See which R objects are now in the R workspace.

`rm(x, y)` Remove objects no longer needed. (Clean up).

`x <- 1:20` Make $x = (1, 2, \dots, 20)$.

`w <- 1 + sqrt(x)/2`

A ‘weight’ vector of standard deviations.

`dummy <- data.frame(x=x, y= x + rnorm(x)*w)`

`dummy` Make a *data frame* of two columns, x and y , and look at it.

`fm <- lm(y ~ x, data=dummy)`

`summary(fm)`

Fit a simple linear regression of y on x and look at the analysis.

`fm1 <- lm(y ~ x, data=dummy, weight=1/w^2)`

`summary(fm1)`

Since we know the standard deviations, we can do a weighted regression.

`attach(dummy)`

Make the columns in the data frame visible as variables.

`lrf <- lowess(x, y)`

Make a nonparametric local regression function.

```

plot(x, y)
    Standard point plot.

lines(x, lrf$y)
    Add in the local regression.

abline(0, 1, lty=3)
    The true regression line: (intercept 0, slope 1).

abline(coef(fm))
    Unweighted regression line.

abline(coef(fm1), col = "red")
    Weighted regression line.

detach() Remove data frame from the search path.

plot(fitted(fm), resid(fm),
     xlab="Fitted values",
     ylab="Residuals",
     main="Residuals vs Fitted")
    A standard regression diagnostic plot to check for heteroscedasticity. Can you
    see it?

qqnorm(resid(fm), main="Residuals Rankit Plot")
    A normal scores plot to check for skewness, kurtosis and outliers. (Not very
    useful here.)

rm(fm, fm1, lrf, x, dummy)
    Clean up again.

    The next section will look at data from the classical experiment of Michaelson and Morley
    to measure the speed of light.

file.show("morley.tab")
    Optional. Look at the file.

mm <- read.table("morley.tab")
mm
    Read in the Michaelson and Morley data as a data frame, and look at it. There
    are five experiments (column Expt) and each has 20 runs (column Run) and s1
    is the recorded speed of light, suitably coded.

mm$Expt <- factor(mm$Expt)
mm$Run <- factor(mm$Run)
    Change Expt and Run into factors.

attach(mm)
    Make the data frame visible at position 2 (the default).

plot(Expt, Speed, main="Speed of Light Data", xlab="Experiment No.")
    Compare the five experiments with simple boxplots.

fm <- aov(Speed ~ Run + Expt, data=mm)
summary(fm)
    Analyze as a randomized block, with 'runs' and 'experiments' as factors.

```

```
fm0 <- update(fm, . ~ . - Run)
anova(fm0, fm)
    Fit the sub-model omitting 'runs', and compare using a formal analysis of vari-
    ance.

detach()
rm(fm, fm0)
    Clean up before moving on.
```

We now look at some more graphical features: contour and image plots.

```
x <- seq(-pi, pi, len=50)
y <- x      x is a vector of 50 equally spaced values in  $-\pi \leq x \leq \pi$ . y is the same.

f <- outer(x, y, function(x, y) cos(y)/(1 + x^2))
    f is a square matrix, with rows and columns indexed by x and y respectively,
    of values of the function  $\cos(y)/(1 + x^2)$ .

oldpar <- par(no.readonly = TRUE)
par(pty="s")
    Save the plotting parameters and set the plotting region to "square".

contour(x, y, f)
contour(x, y, f, nlevels=15, add=TRUE)
    Make a contour map of f; add in more lines for more detail.

fa <- (f-t(f))/2
    fa is the "asymmetric part" of f. (t() is transpose).

contour(x, y, fa, nint=15)
    Make a contour plot, ...

par(oldpar)
    ... and restore the old graphics parameters.

image(x, y, f)
image(x, y, fa)
    Make some high density image plots, (of which you can get hardcopies if you
    wish), ...

objects(); rm(x, y, f, fa)
    ... and clean up before moving on.
```

R can do complex arithmetic, also.

```
th <- seq(-pi, pi, len=100)
z <- exp(1i*th)
    1i is used for the complex number i.

par(pty="s")
plot(z, type="l")
    Plotting complex arguments means plot imaginary versus real parts. This
    should be a circle.
```

```
w <- rnorm(100) + rnorm(100)*1i
```

Suppose we want to sample points within the unit circle. One method would be to take complex numbers with standard normal real and imaginary parts ...

```
w <- ifelse(Mod(w) > 1, 1/w, w)
```

... and to map any outside the circle onto their reciprocal.

```
plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+", xlab="x", ylab="y")
```

```
lines(z)
```

All points are inside the unit circle, but the distribution is not uniform.

```
w <- sqrt(runif(100))*exp(2*pi*runif(100)*1i)
```

```
plot(w, xlim=c(-1,1), ylim=c(-1,1), pch="+", xlab="x", ylab="y")
```

```
lines(z)
```

The second method uses the uniform distribution. The points should now look more evenly spaced over the disc.

```
rm(th, w, z)
```

Clean up again.

```
q()
```

Quit the R program. You will be asked if you want to save the R workspace, and for an exploratory session like this, you probably do not want to save it.