```
#################################################################
#
#    Reinhard Hoepfner -- Schaetzer und Tests -- Sommer 12
#
#    Uebungsblatt 3 - Aufgabe 1
#
#    log-likelihoodflaechen fuer normalverteilte Datensaetze
#
#    29.05.12
#
#    Beachte: die in R vorgegebene Parametrisierung des
Normalverteilungsmodells
#    (durch $\sigma$ statt durch $\sigma^2$)
#    ist die eines Lokations- und Skalenmodells
#
#################################################################3



###################################
#
#    datensatz und gitter
#
###################################

 # waehle einen datensatz :
m0 <- 0.5 ;
s0 <- 1.5 ; # beachte: R will eingabe des skalenparameters
N <- 8 ;
gesicherterdatensatz <- round( rnorm(N, m0, s0), 2 ) ;
gesicherterdatensatz ;

 # definiere funktion zur berechnung der loglikelihood im iid lokations- und
skalenmodell
bildeloglikelihood <- function(datensatz, m, s){
  dat <- datensatz ;
  hilf <- log( dnorm( dat , m, s ) ) ;
  max ( sum(hilf), -1e12 ) ;
  } ; # funktion berechnet die log-likelihoodratio an der stelle (m,s)
  # fuer die gegebenen daten 'datensatz', trunkiert 'nahe' minus unendlich
  # fuer das normalverteilungsmodell als lokations- und skalenmodell
  # fuer andere verteilungsfamilien vom lokations- und skalentyp ersetze z.b.
  # hilf <- log( dcauchy( datensatz , m, s ) ) ;
  # hilf <- log( meineselbstdefiniertedichte( datensatz , m, s, alpha=0.25 ) )
;

 # waehle ein gitter von werten fuer den lokations- und den skalenparameter
mgitter <- seq( m0-2, m0+2, length.out=10) ; sgitter <- seq( 0.5*s0, 2.5*s0,
length.out=10 )  ;


###################################
#
#    berechnung der log likelihood flaeche
#    fuer den datensatz 'gesichertedaten' auf dem durch 'mgitter' und
'sgitter' definierten raster
#    mit normalverteilungsmodell als lokations- und skalenmodell
#
###################################
```

```
loglikelihoodflaeche <- array( 0, dim=c(length(mgitter),length(sgitter)) ) ;
loglikelihoodflaeche ; # startwert zum ueberschreiben
  # matrix mit 'length(mgitter)' zeilen und 'length(sgitter)' spalten

for( i in 1:length(mgitter) )
  for( j in 1:length(sgitter) )
     loglikelihoodflaeche[i,j] <- bildeloglikelihood(gesicherterdatensatz,
mgitter[i], sgitter[j]) ;
 # jetzt ist die loglikelihoodflaeche berechnet und abgespeichert
loglikelihoodflaeche ;  # kontrollausdruck


##################################
#
#   plots perspektivisch
#
##################################

xleg <- "lokationsparameter" ;
yleg <- "skalenparameter" ;
zleg <- "log-likelihood" ;
mainleg <- "normalverteilungsmodell als lokations- und skalenmodell:\n
beobachtungen generiert unter m=&&&& und s=%%%%" ;
mainleg <- gsub("", N, mainleg) ;
mainleg <- gsub("&&&&", round(m0,2), mainleg) ;
mainleg <- gsub("%%%%", round(s0,2), mainleg) ;
mainleg ;

persp( mgitter, sgitter, loglikelihoodflaeche ) ;
persp( mgitter, sgitter, loglikelihoodflaeche,
  xlab=xleg, ylab=yleg, zlab=zleg ) ;
persp( mgitter, sgitter, loglikelihoodflaeche,
  xlab=xleg, ylab=yleg, zlab=zleg,
  xlim=1.1*range(mgitter), ylim=1.0*range(sgitter),
zlim=1.1*range(loglikelihoodflaeche),
  ticktype="detailed", nticks=10, cex.axis=0.5 ) ;

  # die perspektive drehen mit rotation 'theta' und untersicht 'phi'
persp( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
zlab=zleg, theta=30, phi=-10, col=2 ) ;
persp( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
zlab=zleg, theta=120, phi=-10, col=2 ) ;
persp( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
zlab=zleg, theta=210, phi=-10, col=2 ) ;

 # die perspektive drehen mit rotation 'theta' und aufsicht 'phi'
persp( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
zlab=zleg, theta=30, phi=30, col=2 ) ;
persp( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
zlab=zleg, theta=120, phi=30, col=2 ) ;
persp( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
zlab=zleg, theta=210, phi=30, col=2 ) ;

   # ein schoenes bild besseren achsenbeschriftungen und titel
persp( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
zlab=zleg, theta=30, phi=30, col=2, shade=0.5,
  xlim=1.1*range(mgitter), ylim=1.0*range(sgitter),
zlim=1.1*range(loglikelihoodflaeche),
```

```
  ticktype="detailed", nticks=10, cex.axis=0.5, main=mainleg ) ;

  # see 'help(persp)' fuer moegliche einstellungen, z.b. fuer aestethen
persp( mgitter, sgitter, loglikelihoodflaeche, theta=30, phi=30, col=2,
shade=0.5, box=F, main=mainleg ) ;


##################################
#
#   contour plots etc
#
##################################

  # raster farbcodiert :
  # farben und grad der abstufung waehlbar: heat.colors, topo.colors,
terrain.colors, ...
image( mgitter, sgitter, loglikelihoodflaeche) ;
image( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
col=heat.colors(12) ) ;
image( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
col=heat.colors(48) ) ;

 # linear interpoliert und farbcodiert :
filled.contour( mgitter, sgitter, loglikelihoodflaeche, xlab=xleg, ylab=yleg,
main=mainleg, col=terrain.colors(48) ) ;
 # der wahre parameter :
 # points( m0, s0, col=2 ) ;
 # der maximum likelihood estimator
 # points( mean(gesicherterdatensatz), sqrt(var(gesicherterdatensatz)),
pch=18, col=2, cex=1.5 ) ;
 # vorsicht: diese fertige funktion produziert skalierungsfehler !!!

  # oder
contour( mgitter, sgitter, loglikelihoodflaeche ) ;
contour( mgitter, sgitter, loglikelihoodflaeche, nlevels=100, xlab=xleg,
ylab=yleg, main=mainleg ) ;
  # mit wahrem parameter
points( m0, s0, col=2 ) ;
  # und maximum likelihood estimator
points( mean(gesicherterdatensatz), sqrt(var(gesicherterdatensatz)), pch=18,
col=2, cex=1.5 ) ;


#######################  ende  29.05.12
```