```
#############################################################
#
#     Reinhard Hoepfner
#
#     Praktikum zu den 'Grundlagen der Stochastik'
#
#     Erzeugung von 2-dim normalverteilten Zufallszahlen
#
#     12.11.18
#
#############################################################




#################################
#
#   vorueberlegung zu 2x2 matritzen
#
#   symmetrisch und strikt positiv definit
#
#   kovarianzmatrix mit beiden diagonalelementen  =  1
#   und nichtdiagonalelement  =  rho
#
#################################


rho <- -0.95 ;
  # muss zwischen -1 und 1 sein


  # bilde die kovarianzmatrix 'Lambda'
bilde <- function(rho){
  matrix( c( 1, rho, rho, 1 ) , nrow=2, ncol=2, byrow=T ) ;
  } ;

Lambda <- bilde(rho) ;
Lambda ;
#        [,1]  [,2]
#  [1,]  1.00  -0.95
#  [2,] -0.95   1.00



  # die inverse dazu braucht man in formel fuer normaldichte
Lambda_inv <- solve(Lambda) ;
Lambda_inv ;
#            [,1]     [,2]
#  [1,] 10.25641  9.74359
#  [2,]  9.74359 10.25641

  # diese inverse ist
(1-rho^2)^(-1) * matrix( c( 1, -rho, -rho, 1 ) , nrow=2, ncol=2, byrow=T ) ;
#            [,1]     [,2]
#  [1,] 10.25641  9.74359
#  [2,]  9.74359 10.25641

  # aus der formel fuer die normaldichte mit kovarianzmatrix 'Lambda' :

normaldichte_2d_rho <- function( x1, x2, rho ){
```

```
  1/(2*pi*sqrt(1-rho^2)) * exp( -0.5*( x1^2 + x2^2 - 2*x1*x2*rho )/( 1-rho^2 ) ) ;
  } ;
meinrho <- rho ;
normaldichte_2d_rhofest <- function(x1,x2) normaldichte_2d_rho(x1,x2,rho=meinrho) ;

x1 <- seq( -4, 4, 0.1 ) ;
x2 <- x1 ;
werte <- outer( x1, x2, normaldichte_2d_rhofest ) ;
persp( x1, x2, werte,
  xlim=c(-4.25,4.25), ylim=c(-4.25,4.25), zlim=c(0,0.5),
  ticktype="detailed", nticks=8, cex.axis=0.7,
  col=7, shade=0.2,
  xlab="", ylab="", zlab="", main=leg ) ;




##################################
#
#   weitere vorueberlegung :
#
#   square root einer 2x2 matrix
#   symmetrisch und strikt pos def
#
#   beide diagonalelemente = 1 , nichtdiagonalelement = rho
#
##################################


  # quadratwurzel zu Lambda :
  # vorsicht: 'sqrt(Lambda)' geht nicht
  # genausowenig geht 'Lambda^(1/2)'

  # geht so :

bildesqrt <- function(rho){
  d1 <- ifelse( rho!=0 , (1 - sqrt(1-rho^2) )/rho , 0 ) ;
  d2 <- 1 / sqrt(1+d1^2) ;
  matrix( c( d2, d1*d2, d1*d2, d2 ) , nrow=2, ncol=2, byrow=T )  ;
  } ;


Lambda_sqrt <- bildesqrt(rho) ;
Lambda_sqrt ;
#       [,1]       [,2]
#  [1,]  0.8100154 -0.5864086
#  [2,] -0.5864086  0.8100154


Lambda_sqrt %*% Lambda_sqrt ;
#        [,1]  [,2]
#  [1,]  1.00 -0.95
#  [2,] -0.95  1.00
Lambda ;
#        [,1]  [,2]
#  [1,]  1.00 -0.95
#  [2,] -0.95  1.00
```

```
 # dabei gibt es gewisse rundungsfehler denn
Lambda_sqrt %*% Lambda_sqrt == Lambda ;
#         [,1]  [,2]
#  [1,]  TRUE FALSE
#  [2,] FALSE  TRUE



##################################
#
#  bemerkung : eine beliebige 2x2 kovarianzmatrix
#
#  mit diagonalelementen 'sigma1^2' und 'sigma2^2'
#
#  und nichtdiagonalelement  'rho * sigma1 * sigma2'
#
#  kann man immer auf die oben betrachtete gestalt transformieren
#
##################################


sigma1 <- 1.7 ;
sigma2 <- 0.8 ;

covarianzmatrix <- function( sigma1, sigma2, rho ){
  a <- sigma1^2 ;
  b <- sigma2^2 ;
  d <- sigma1 * sigma2 * rho ;
  matrix( c( a, d, d, b ) , nrow=2, ncol=2, byrow=T ) ;
  } ;

Sigma <- covarianzmatrix( sigma1, sigma2, rho ) ;
Sigma ;
#         [,1]   [,2]
#  [1,]  2.890 -1.292
#  [2,] -1.292  0.640


  # die oben benutzte gestalt erhaelt man durch
  # transformation mit diagonalmatrix :

A <- diag( c( 1/sigma1, 1/sigma2 ) , 2 ) ;
A ;
#           [,1] [,2]
# [1,] 0.5882353 0.00
# [2,] 0.0000000 1.25

Lambda <- A %*% Sigma  %*%  A ;
Lambda ;
#        [,1]  [,2]
# [1,]  1.00 -0.95
# [2,] -0.95  1.00



##############################
#
```

```
#  2d zufallszahlen mit kovarianzmatrix 'Lambda'
#  und mit mittelwert null erzeugen
#
##############################


rho <- 0.77 ;
# rho <- 0.24 ;
# rho <- -0.24 ;
# rho <- -0.77 ;
# rho <- -0.95 ;

Lambda <- bilde(rho) ;
Lambda_sqrt <- bildesqrt(rho) ;

 # erzeuge  N  2d-normalverteilte ZV mit kovarianzmatrix 'Lambda'
 # und speichere diese in den spalten einer matrix ab

N <- 10000 ;
# N <- 1000 ;
# N <- 250 ;
# N <- 50 ;

daten <- matrix( 0, nrow=2, ncol=N ) ;
 # format zum ueberschreiben : in die spalte 'k'
 # wird die 'k'-te 2d zufallszahl eingetragen
for ( j in 1:N ) daten[,j] <- Lambda_sqrt %*% rnorm(2,0,1) ;
dim(daten) ;

x <- daten[1,] ;
y <- daten[2,] ;

xleg <- "x-komponente" ;
yleg <- "y-komponente" ;
leg <- "  2-dim normalverteilte zufallszahlen mit  rho = %%%%" ;
leg <- gsub( "", N, leg ) ;
leg <- gsub( "%%%%", rho, leg ) ;
leg ;

# par( pty="m" ) ;
par( pty="s" ) ;
plot( c(-5,5), c(-5,5), type="n", xlab=xleg, ylab=yleg, main=leg ) ;
points( x, y, col=2, pch=16 ) ;




###############################
#
#  rechne eine lineare regression fuer diese daten
#
###############################


xquer <- mean(x) ; round(xquer,2) ;
yquer <- mean(y) ; round(yquer,2) ;
  # durch diesen punkt geht die regressionsgerade

sigmax <- sqrt( mean( (x-xquer)^2 ) ) ; round(sigmax,2) ;
sigmay <- sqrt( mean( (y-yquer)^2 ) ) ; round(sigmay,2) ;
```

```
  # wurzel aus den empirischen varianzen

covxy <- mean( (x-xquer)*(y-yquer) ) ; round(covxy,2) ;
  # die empirische kovarianz

rhoxy <- covxy / (sigmax * sigmay) ; round(rhoxy,2) ;
  # die korrelation in den daten

rhoxy^2 ;
  # masszahl fuer guete der approximation durch die regressionsgerade



a <- covxy / sigmax^2 ; a ;
  # die steigung der regressionsgerade  g(x) = ax+b
b <- yquer - a*xquer ; b ;
  # der absolutterm der regressionsgerade  g(x) = ax+b


Mx <- max( x ) ;
mx <- min( x ) ;
t <- c( 1.1*mx, 1.1*Mx ) ;

lines( t, a*t+b , lwd=2, lty=6, col=3 ) ;
  # die regressionsgerade
points( xquer, yquer, col=3, pch=8, lwd=2 ) ;
  # die empirischen mittelwerte der komponenten

ergebnisleg <- " korrelation emp : rho_n =  , steigung regressionsgerade : %%%% \n
relativer verbleibender fehler : 1-rho_n^2 = &&&& " ;
ergebnisleg <- gsub( "", round(rhoxy,4), ergebnisleg ) ;
ergebnisleg <- gsub( "%%%%", round( a, 4 ), ergebnisleg ) ;
ergebnisleg <- gsub( "&&&&", 1-round(rhoxy^2,4), ergebnisleg ) ;
legend( c(-5,5), c(-5,-4), ergebnisleg, cex=0.8, bg = 8 ) ;



  # das jetzt mit anderen werten von 'rho' durchspielen



#####  12.11.18 ########
```