```
#####################################
#
#     Reinhard Hoepfner, WS 18/19
#
#     Praktikum zur 'Einfuehrung in die Stochastik'
#
#
#     2d-Daten aus iid Beobachtungen,
#     Kernschaetzer zur Schaetzung
#     der diesen Daten zugrundeliegende Dichte
#
#
#     Daten:  N  zweidimensional normalverteilte daten
#     mit Mittelwert  und    Kovarianz
#
#            0                1  rho
#            0                rho  1
#
#     mit unbekanntem Parameter -1 < rho < 1
#
#
#
#     Hintergrund in der nichtparametrischen Statistik siehe
#
#     A. Tsybakov,
#     Introduction to nonparametric statistics, Springer 2009
#
#     03.02.19
#
#####################################



#####################################
#
#  funktionsdefinitionen
#
#####################################


  # berechne 2d normaldichten
  # wie in vorlesung kapitel I.D :

bildenormaldichte_2d_rho_beliebig <- 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 ) ) ;
  } ; # dichte der 2d-normalverteilung
  # zentriert
  # varianz der komponenten = 1
  # kovarianz = rho



  # simuliere 2d normalverteilte zufallszahlen
  # wie in vorlesung kapitel I.D :

bildenormalzv_2d_rho_beliebig <- function(N,rho){
  x1 <- rnorm( N, 0, 1 ) ;
  x2 <- rho*x1 + rnorm( N, 0, sqrt(1-rho^2) ) ;
  matrix( c( x1, x2 ), nrow=2, ncol=N, byrow=T ) ;
  } # simuliert N  zufallszahlen
  # 2d-normalverteilt,  zentriert
  # varianz der komponenten = 1
  # kovarianz = rho


  # beachte: R parametrisiert mit standardabweichungen
  # und nicht mit varianzen
```

```
###################################
#
#   erzeuge einen datensatz der laenge  N
#   mit zufaellig gewaehltem 'wahrem'  rho
#
###################################


rho_wahr <- round( runif(1,-0.9,0.9) , 2 )  ; rho_wahr ;
  # waehle zufaellig eine kovarianz in (-0.9,0.9)
  # (schliesse dabei 'extreme' korrelationen aus)



  # waehle laenge des datensatzes
N <- 100 ;
# N <- 250 ;
# N <- 500 ;
# N <- 1000 ;
# N <- 2500 ;
# N <- 5000 ;
# N <- 10000 ;



  # den datensatz bilden
daten <- bildenormalzv_2d_rho_beliebig(N,rho_wahr) ;
dim(daten) ;
  # 2d-datensatz (X_i,Y_i) , 1 <= i <= N
  # daten X_i in der ersten zeile
  # daten Y_i in der ersten zeile


  # daten anschauen
par(pty="s") ;
plot( range(daten[1,]), range(daten[2,]), type="n", xlab="", ylab="", main="" ) ;
for(i in 1:N) points( daten[1,i],daten[2,i], pch=16, cex=0.5, col=2 ) ;
par(pty="m") ;




###################################
#
#   konstruiere einen kernschaetzer :
#
#   als bekannt sei vorausgesetzt (modellvoraussetzung):
#   2d-normalverteilte iid beobachtungen,
#   varianz der komponenten = 1
#   unbekannt sei die kovarianz/korrelation in (-1,1)
#
#   bilde kernschaetzer fuer die unbekannte dichte
#
###################################




###################################
#
#   wahl eines kerns
#
#   nichtparametrische statistik ( --> tsybakov):
#
```

```
#   fuer $d$-dimensionale daten $x_1,...,x_n$
#   mit dichte der glattheitsordnung $\beta$
#   benutzt man einen kern $K$ der ordnung $\floor(\beta)$
#   und eine bandbreite $h$ der ordnung
#   $n^{ - \frac{ 1 }{ 2\beta + d } }$
#
#   der kernschaetzer fuer die dichte ist dann
#   $$
#   \mathbb{R}^d  \ni
#   y  \mapsto
#   \frac{1}{ n h^d } \sum_{j=1}^n  K( \frac{ y - x_j }{ h } )
#   $$
#   und die konvergenzgeschwindigkeit des schaetzers ist
#   $$
#   $n^{ \frac{ \beta }{ 2\beta + d } }$
#   $$
#
#   mache das hier einfach, mit $\beta=1$ und $d=2$
#
################################


 # eine bandbreite fuer den kernschaetzer waehlen
h <- round( 2 * N^(-1/4), 3 ) ; h ;
 # nichtparametrische statistik :
 # man weiss wie die bandbreite gewaehlt werden muss
 # (abhaengig von der laenge N des satzes von iid daten,
 # von der glattheit der zugrundeliegenden dichte,
 # und von der dimension)



 # fuer  N  und  h  wie eben definiert
 # definiere einen kern
 # zur konstruktion des kernschaetzers
 # (muss bei N daten die masse 1/N bekommen) :

kernNh <- function(x1,y1){
 (1/N) * (2*pi*h^2)^(-1) * exp( - 0.5 * ( (x1)^2 + (y1)^2 ) / h^2 )
 }  ; # ende funktionsdefinition
 # kern ist zentriert in  0  (2d)
 # radialsymmetrisch mit bandbreite  h
 # hat masse 1/N  weil man mit N datenpunkten arbeiten will
 # 3*h  ist die typische breite dieses (normalverteilungs-) kerns



  # fenstergroesse zur visualisierung des gesamten datensatzes
  # und der zu bildenden kernschaetzer :
r <- 5 ; r ;

 # damit perspektivplots den gewaehlten kern angemessen abbilden,
 # braucht man 10 gitterlinien pro typische breite des kerns
step <- 10/(3*h) ; step ;

 # benoetigte anzahl von gitterlinien
 # zur guten darstellung einer flaeche ueber  $[-r,r]\times[-r,r]$
 # als vielfaches von 10
l <- 10 * ceiling( ( 2*r*step ) / 10 ) ; l;
```

```
 # nachpruefen ob der gewaehlte kern
 # auf diese weise brauchbar abgebildet wird :

gridx <- seq( -r, r, length.out=l ) ;
gridy <- seq( -r, r, length.out=l ) ;
hilfz <- outer( gridx, gridy, kernNh ) ;

persp( gridx, gridy, hilfz,
 xlab="x", ylab="y", zlab="",
 zlim = 1.1*range(hilfz),
 col=7,
 ticktype="detailed", nticks=12 ) ;



par( pty="s") ;
contour( gridx, gridy, hilfz,
 xlab="x", ylab="y",
 col=3, nlevels=25,
 xlim=c(-r,r), ylim=c(-r,r) ) ;
points( 0, 0, pch=16, col=2  ) ;
par( pty="m") ;



filled.contour( gridx, gridy, hilfz,
  xlab="x", ylab="y",
  col=topo.colors(36) ) ;



########################################
#
#    bilde mit diesem kern einen kernschaetzer
#    fuer den datensatz
#
########################################


  # demonstrationslauf zur definition des kernschaetzers :

surface <- matrix( 0, nrow=l, ncol=l ) ;
dim( surface) == dim( outer(gridx,gridy) );
  # startwerte zum ueberschreiben
surfacealt <- surface ;
i <- 1 ;


  # testlauf: kern der folgenden programmschleife
i ;
surfacealt <- surface ;
xhilf <- gridx - daten[1, i]*rep(1,l) ;
yhilf <- gridy - daten[2,i]*rep(1,l) ;
  # man muss den kern ueber dem i-ten datenpunkt zentriert einrechnen
zhilf <- outer( xhilf, yhilf, kernNh ) ;

surface <- surfacealt + zhilf  ;
leg_i <- "zwischenstand nach einrechnen der ersten %%%% von &&&& datenpunkten" ;
leg_i <- gsub( "%%%%", i, leg_i ) ;
leg_i <- gsub( "&&&&", N, leg_i ) ;
persp( gridx, gridy, surface,
  xlab="x", ylab="y", zlab="",
  zlim = 1.1*range(surface),
  col=7, main=leg_i,
  ticktype="detailed", nticks=12 ) ;
i <- i+1 ;
  # ende testlauf
```

```
  # berechnung des kernschaetzers aus allen N daten

surface <- matrix( 0, nrow=l, ncol=l ) ;
surfacealt <- surface ;
for (i in 1:N) {
  surfacealt <- surface ;
  xhilf <- gridx - daten[1,i]*rep(1,l) ;
  yhilf <- gridy - daten[2,i]*rep(1,l) ;
  zhilf <- outer( xhilf, yhilf, kernNh ) ;
  surface <- surfacealt + zhilf  ;
  } ; # ende berechnung kernschaetzer
leg <- "kernschaetzer mit bandbreite  basierend auf allen %%%% daten" ;
leg <- gsub ( "", h, leg ) ;
leg <- gsub ( "%%%%", N, leg ) ;



  # erste darstellung
persp( gridx, gridy, surface,
  xlab="x", ylab="y", zlab="",
  zlim = 1.1*range(surface),
  col=7, main=leg,
  ticktype="detailed", nticks=12 ) ;


  # zweite darstellung
par( pty="s") ;
contour( gridx, gridy, surface,
  xlab="x", ylab="y",
  col=3, nlevels=25,
  xlim=c(-r,r), ylim=c(-r,r), main=leg  ) ;
points( daten[1,], daten[2,], pch=16, cex=0.5, col=2  ) ;
par( pty="m") ;


  # dritte darstellung
filled.contour( gridx, gridy, surface,
  xlab="x", ylab="y",
  col=topo.colors(36), main=leg ) ;







##################################
#
#   vergleiche den bis hier errechneten kernschaetzer
#   mit der wahren dichte der einzelbeobachtung
#
##################################


bildenormaldichte_2d_rho_wahr <- function( x1, x2 ){
  bildenormaldichte_2d_rho_beliebig( x1, x2, rho=rho_wahr ) ;
  } ;
dichte_2d_wahr <- outer( gridx, gridy, bildenormaldichte_2d_rho_wahr ) ;
m_2d_wahr <- max(dichte_2d_wahr) ; m_2d_wahr ;
```

```
 # erster vergleich :
 # (rechts und links wird dieselbe hoehe der z-achse befohlen)
par( mfrow=c(1,2) ) ;
persp( gridx, gridy, dichte_2d_wahr,
  col=7, main="wahre dichte",
  zlim = 1.1*range(dichte_2d_wahr),
  ticktype="detailed", nticks=12 ) ;
persp( gridx, gridy, surface,
  col=7, main="geschaetzte dichte",
  zlim = 1.1*range(dichte_2d_wahr),
  ticktype="detailed", nticks=12 ) ;
par( mfrow=c(1,1) ) ;
```

```
  # zweiter vergleich :
  # (rechts und links wird dieselbe skala von hoehenlinien befohlen)
par( mfrow=c(1,2) ) ;
par( pty="s") ;
contour( gridx, gridy, dichte_2d_wahr,
 xlab="x", ylab="y",
 col=3, #nlevels=25,
 levels = seq( 0, m_2d_wahr, length.out=25 ),
 xlim=c(-r,r), ylim=c(-r,r), main="wahre dichte"  ) ;
contour( gridx, gridy, surface,
 xlab="x", ylab="y",
 col=3, #nlevels=25,
 levels = seq( 0, m_2d_wahr, length.out=25 ),
 xlim=c(-r,r), ylim=c(-r,r), main="geschaetzte dichte"  ) ;
points( daten[1,], daten[2,], pch=16, cex=0.5, col=2  ) ;
par( pty="m") ;
par( mfrow=c(1,1) ) ;
```

```
  # dritter vergleich :
  # (vergleiche die skalierungsangaben rechts)

filled.contour( gridx, gridy, dichte_2d_wahr,
  xlab="x", ylab="y",
  col=topo.colors(36), main="wahre dichte" ) ;

filled.contour( gridx, gridy, surface,
  xlab="x", ylab="y",
  col=topo.colors(36), main="geschaetzte dichte" ) ;
```

```
##################################
#
#  BIS HIER  03.02.19
#
##################################
```