ภาคผนวก ก

โปรแกรมประมวลผล

โปรแกรมที่ใช้สำหรับประมวลผลการวิจัยในหัวข้อ การเลือกแบนวิดธ์สำหรับการประมาณฟังก์ชันความหนาแน่นแบบเคอร์เนลที่ขอบเขต ด้วยโปรแกรมคอมพิวเตอร์ R version 2.10.1 ประกอบด้วย การสร้างตัวแปรสุ่มจากฟังก์ชันการแจกแจงที่กำหนด ตามขนาดตัวอย่างที่ต้องการ การประมาณค่าตัวแปรต่าง ๆ ที่ใช้ในการวิจัย การประมาณค่าแบนวิดธ์จากวิธีการเลือกแบนวิดธ์ทั้ง 6 วิธี การประมาณฟังก์ชันความหนาแน่นของความน่าจะเป็นด้วยวิธีของ Zhang- และคณะ (1999) การหาค่าความคลาดเคลื่อนกำลังสองรวมเฉลี่ย และ ค่าแบนวิดธ์เฉลี่ย

****

สร้างตัวแปรสุ่มจากฟังก์ชันความหนาแน่นของความน่าจะเป็นรูปแบบที่ 1

$$f(x) = 5e^{-5x}$$

****

```
gen.x <- function (n)
{
  x <- rexp(n,rate=5)
  return(x)
}
```

****

สร้างลำดับเลขคณิต สำหรับการประมาณฟังก์ชันความหนาแน่นของความน่าจะเป็นรูปแบบที่ 1

$$f(x) = 5e^{-5x}$$

****

```
gen.y <- function (n,code)
{
  x <- gen.x(n)
  y <- seq(from=0,to=max(x)+0.055 ,by=0.005)
  return(y)
}
```

****

สร้างตัวแปรสุ่มจากฟังก์ชันความหนาแน่นของความน่าจะเป็นรูปแบบที่ 2

$$f(x)=\frac{5}{4}(1+15x)e^{-5x}$$

****

```
gen.x2 <- function (n)
{
  kernel=function(x){(5/4)*(1+15*x)*exp(-5*x)}
  norm.const=integrate(kernel,lower=-0,upper=Inf)$value
  pdf=function(x){kernel(x)/norm.const}
  cdf=function(x){integrate(pdf,lower=0,upper=x)$value}
        icdf=function(u)
        {
        h=function(x){cdf(x)-u}
        x=uniroot(h,interval=c(0,10))$root
        return(x)
        }
  u=runif(n)
  x=sapply(u,icdf)
  return(x)
}
```

****

สร้างลำดับเลขคณิต สำหรับการประมาณฟังก์ชันความหนาแน่นของความน่าจะเป็นรูปแบบที่ 2

$$f(x)=\frac{5}{4}(1+15x)e^{-5x}$$

****

```
gen.y2 <- function (n,code)
{
  x <- gen.x2(n)
  y <- seq(from=0,to=max(x)+0.055 ,by=0.005)
  return(y)
}
```

```
*****************************************************************************************************
```

กำหนดขอบเขตของค่าเพื่อใช้ในการชี้วัด สำหรับฟังก์ชันเคอร์เนลของจุดสิ้นสุด

```
*****************************************************************************************************

indic <- function(m)
{
  if (m==TRUE)
        {return(1)}
  else
        { return(0)}
}



*****************************************************************************************************
```

ประมาณค่า d ด้วย d_n

```
*****************************************************************************************************

dn <- function (x,y,n,code)
{
a <- rep(0,n)
b <- rep(0,n)
nb <- 1000
hmax <- 1.144 * sqrt(var(x)) * (length(x)^(-1/5))
lower <- 0.05 * hmax
upper <- 1.2 * hmax
tol <- 0.1 * lower

if (code == 1) { h <- bw.SJ(x,nb=1000,lower=0.05 * hmax,
        upper=1.2 * hmax,method = "dpi")}
if (code == 2) { h <- 1.06 * sqrt(var(x)) * (length(x)^(-1/5))}
if (code == 3) { h <- bw.ucv(x,nb=1000,lower=0.05 * hmax,
        upper=1.2 * hmax,tol=0.1 * lower)}
if (code == 4) { h <- bw.nrd0(x)}
if (code == 5) { h <- bw.bcv(x,nb=1000,lower=0.05 * hmax,
        upper=1.2 * hmax,tol=0.1 * lower)}
if (code == 6) { h <- bw.SJ(x,nb=1000, lower=0.05 * hmax,
        upper=1.2 * hmax,method = "ste",tol=0.1 * lower)}

fnh.star <- rep(0,n)
fn0.star <- rep(0,n)

for(j in 1:n)
{
```

```
a[j] <- (h-x[j])/h
b[j] <- (-x[j])/(4.427786896*h)
fnh.star[j] <- (1/sqrt(2*pi))*(exp((-1/2)*(a[j]^2)))
fn0.star[j] <- (6+(18*b[j])+12*((b[j])^2))*indic(abs(b[j]+0.5)<=0.5)
}
ans.fnhstar <- sum(fnh.star)/(n*h)
ans.fn0star <- sum(fn0.star)/(4.427786896*n*h)

  if(ans.fn0star > (1/(n^2)))
        { fn0 <- ans.fn0star }
  else
        { fn0 <- (1/(n^2)) }

fnh <- ans.fnhstar+(1/(n^2))
dn  <- (log(fnh)-log(fn0))/h
return(dn)
}
```

****

ประมาณความหนาแน่นด้วยวิธีของ Zhang และคณะ (1999)

****

```
fy <- function(x,y,n,code)
{
gx <- rep(0,n)
m  <- rep(0,n)
p  <- rep(0,n)
km <- rep(0,n)
kn <- rep(0,n)
ku <- rep(0,n)
u  <- rep(0,n)
l=length(y)
fhat <- rep(0,l)
for(k in 1:length(y))
{
nb <- 1000
hmax <- 1.144 * sqrt(var(x)) * (length(x)^(-1/5))
lower <- 0.05 * hmax
upper <- 1.2 * hmax
tol <- 0.1 * lower

if (code == 1) { h <- bw.SJ(x,nb=1000,lower=0.05 * hmax,
        upper=1.2 * hmax,method = "dpi")}
```

```
if (code == 2) { h <- 1.06 * sqrt(var(x)) * (length(x)^(-1/5))}
if (code == 3) { h <- bw.ucv(x,nb=1000,lower=0.05 * hmax,
        upper=1.2 * hmax,tol=0.1 * lower)}
if (code == 4) { h <- bw.nrd0(x)}
if (code == 5) { h <- bw.bcv(x,nb=1000,lower=0.05 * hmax,
        upper=1.2 * hmax,tol=0.1 * lower)}
if (code == 6) { h <- bw.SJ(x,nb=1000, lower=0.05 * hmax,
        upper=1.2 * hmax,method = "ste",tol=0.1 * lower)}

d  <- dn(x,y,n,code)
   for(j in 1:n)
   {
     gx[j] <- x[j]+(d*(x[j]^2))+(0.55*(d^2)*(x[j]^3))
        m[j]  <- ((y[k]-x[j])/h)
        p[j]  <- ((y[k]+gx[j])/h)
        km[j] <- (1/sqrt(2*pi))*(exp((-1/2)*(m[j]^2)))
        kn[j] <- (1/sqrt(2*pi))*(exp((-1/2)*(p[j]^2)))
        ku[j] <- (km[j]+kn[j])
   } ## Loop j

fhat[k] <- sum(ku)/(n*h)
} ## Loop k
return(fhat,h)
} ## Loop Function ##
```

****

ทำการประมวลผลเพื่อหาค่าความคลาดเคลื่อนกำลังสองรวมเฉลี่ยและค่าแบนวิดธ์เฉลี่ย
ที่เกิดขึ้นจากการประมาณความหนาแน่นด้วยวิธีของ Zhang และคณะ (1999)

****

```
compare.h <- function(M)
{
library (stats)
library (KernSmooth)
library (MASS)
set.seed(1234)
n <- c(30,50,100,150,200)
# code = 1 : hDPI
# code = 2 : hROT
# code = 3 : hLSCV
# code = 4 : hSROT
# code = 5 : hBCV
# code = 6 : hSTE
```

```
code  <- c(1,2,3,4,5,6)
cat('\t','**************************************************************
**************************','\n')
cat('\t\t',' Bandwidth selection for kernel density estimation at the boundary. ','\n')
cat('\t','**************************************************************
**************************','\n')
cat('\t','n','\t','code','\t\t','amise1','\t','hbar1','\t\t','\t','amise2','\t','hbar2','\n')

for (j in 1:length(n))
{
for (t in 1:length(code))
{
    if (code[t] == 1) {label <- 'h Direct plug-in'}
    if (code[t] == 2) {label <- 'h Rules of thumb'}
    if (code[t] == 3) {label <- 'h Least squares Cross-validation'}
    if (code[t] == 4) {label <- 'h Silverman's rules of thumbs'}
    if (code[t] == 5) {label <- 'h Biased cross validation'}
    if (code[t] == 6) {label <- 'h Solve the equation plug-in'}

mise1 <- rep(0,M)
mise2 <- rep(0,M)
h1 <- rep(0,M)
h2 <- rep(0,M)

for (i in 1:M)
{
x1 <- gen.x(n[j])
x2 <- gen.x2(n[j])
y1 <- gen.y(n[j],code[t])
y2 <- gen.y2(n[j],code[t])
hatf1  <- fy(x1,y1,n[j],code[t])
hatf2  <- fy(x2,y2,n[j],code[t])
fhat1  <- hatf1$fhat
fhat2  <- hatf2$fhat
ftrue1 <- (5*(exp((-5)*y1)))
ftrue2 <- ((5/4)*(1+(15*y2))*(exp((-5)*y2)))

#******MISE and Bandwidth for 1st density and 2nd density******
mise1[i] <- sum((fhat1-ftrue1)^2)/length(y1)
h1[i]<-hatf1$h
mise2[i] <- sum((fhat2-ftrue2)^2)/length(y2)
h2[i]<-hatf2$h
} ## loop i

#*Average MISE and Average Bandwidth for 1st density and 2nd density*
amise1 <- sum(mise1)/M
hbar1 <- sum(h1)/M
```

```
amise2 <- sum(mise2)/M
hbar2 <- sum(h2)/M

cat('\t',n[j], '\t',code[t], '\t', '\t',amise1, '\t',hbar1, '\t', '\t',amise2, '\t',hbar2,'\n')
} ## loop t
} ## loop j
} ## function ##
```

****

การเลือกแบนวิดธ์ด้วยวิธี Direct plug-in (DPI) และ วิธี Solve the equation plug-in (STE)

****

```
bw.SJ
function (x, nb = 1000, lower = 0.05 * hmax, upper = 1.2 * hmax, method =
c("ste","dpi"), tol = 0.1 * lower)
{
   if ((n <- length(x)) < 2)
      stop("need at least 2 data points")
   if (!is.numeric(x))
      stop("invalid 'x'")
   storage.mode(x) <- "double"
   method <- match.arg(method)
   fSD <- function(h, x, alph2, c1, n, d) (c1/SDh(x, alph2 *
      h^(5/7), n, d))^(1/5) - h
   SDh <- function(x, h, n, d) .C(R_band_phi4_bin, as.integer(n),
      as.integer(length(x)), as.double(d), x, as.double(h),
      u = double(1))$u
   TDh <- function(x, h, n, d) .C(R_band_phi6_bin, as.integer(n),
      as.integer(length(x)), as.double(d), x, as.double(h),
      u = double(1))$u
   Z <- .C(R_band_den_bin, as.integer(n), as.integer(nb),
         d = double(1), x, cnt = integer(nb))
   d <- Z$d
   cnt <- as.integer(Z$cnt)
   scale <- min(sd(x), IQR(x)/1.349)
   a <- 1.24 * scale * n^(-1/7)
   b <- 1.23 * scale * n^(-1/9)
   c1 <- 1/(2 * sqrt(pi) * n)
   TD <- -TDh(cnt, b, n, d)
   if (!is.finite(TD) || TD <= 0)
      stop("sample is too sparse to find TD")
   if (method == "dpi")
      res <- (c1/SDh(cnt, (2.394/(n * TD))^(1/7), n, d))^(1/5)
```

```
   else {
      if (bnd.Miss <- missing(lower) || missing(upper)) {
         hmax <- 1.144 * sd(x) * n^(-1/5)
      }
      alph2 <- 1.357 * (SDh(cnt, a, n, d)/TD)^(1/7)
      if (!is.finite(alph2))
         stop("sample is too sparse to find alph2")
      itry <- 1
      while (fSD(lower, cnt, alph2, c1, n, d) * fSD(upper,
         cnt, alph2, c1, n, d) > 0) {
         if (itry > 99 || !bnd.Miss)
         stop("no solution in the specified range of bandwidths")
         if (itry%%2)
            upper <- upper * 1.2
         else lower <- lower/1.2
         if (getOption("verbose"))
            message(gettextf("increasing bw.SJ() search interval (%d) to [%.4g,%.4g]",
itry, lower, upper), sep = "", domain = NA)
         itry <- itry + 1
      }
      res <- uniroot(fSD, c(lower, upper), tol = tol, x = cnt,
         alph2 = alph2, c1 = c1, n = n, d = d)$root
   }
   res
}
```

****

การเลือกแบนวิดธ์ด้วยวิธี Least squares cross-validation (LSCV)

****

```
bw.ucv
function (x, nb = 1000, lower = 0.05 * hmax, upper = 1.2 * hmax,   tol = 0.1 * lower)
{
   if ((n <- length(x)) < 2)
      stop("need at least 2 data points")
   if (!is.numeric(x))
      stop("invalid 'x'")
   fucv <- function(h, x, n, d) .C(R_band_ucv_bin, as.integer(n),
      as.integer(length(x)), as.double(d), x, as.double(h),
      u = double(1))$u
   hmax <- 1.144 * sqrt(var(x)) * n^(-1/5)
   storage.mode(x) <- "double"
```

```
   Z <- .C(R_band_den_bin, as.integer(n), as.integer(nb), d = double(1), x, cnt =
integer(nb))
   d <- Z$d
   cnt <- as.integer(Z$cnt)
   h <- optimize(fucv, c(lower, upper), tol = tol, x = cnt,
      n = n, d = d)$minimum
   if (h < lower + tol | h > upper - tol)
      warning("minimum occurred at one end of the range")
   h
}
```

****

การเลือกแบนวิดธ์ด้วยวิธี Silverman's rules of thumb (SROT)

****

```
bw.nrd0
function (x)
{
   if (length(x) < 2)
      stop("need at least 2 data points")
   hi <- sd(x)
   if (!(lo <- min(hi, IQR(x)/1.34)))
      (lo <- hi) || (lo <- abs(x[1L])) || (lo <- 1)
   0.9 * lo * length(x)^(-0.2)
}
```

****

การเลือกแบนวิดธ์ด้วยวิธี Biased cross-validation (BCV)

****

```
bw.bcv
function (x, nb = 1000, lower = 0.05 * hmax, upper = 1.2 * hmax,   tol = 0.1 * lower)
{
   if ((n <- length(x)) < 2)
      stop("need at least 2 data points")
   if (!is.numeric(x))
      stop("invalid 'x'")
   fbcv <- function(h, x, n, d) .C(R_band_bcv_bin, as.integer(n),
      as.integer(length(x)), as.double(d), x, as.double(h),
```

```
        u = double(1))$u
    hmax <- 1.144 * sqrt(var(x)) * n^(-1/5)
    storage.mode(x) <- "double"
    Z <- .C(R_band_den_bin, as.integer(n), as.integer(nb), d = double(1), x, cnt =
integer(nb))
    d <- Z$d
    cnt <- as.integer(Z$cnt)
    h <- optimize(fbcv, c(lower, upper), tol = tol, x = cnt,
        n = n, d = d)$minimum
    if (h < lower + tol | h > upper - tol)
        warning("minimum occurred at one end of the range")
    h
}
```