code

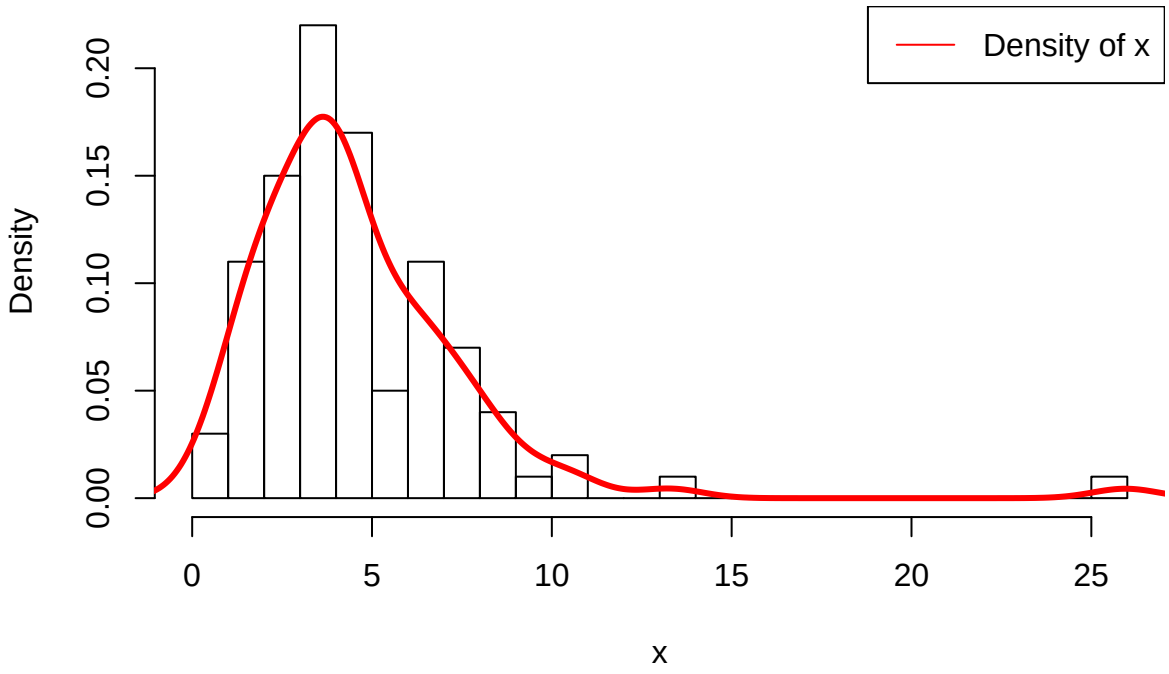
Yaqiong Yao

12/15/2018

Univariate Case

Generate dataset

```
n <- 100
set.seed(123)
u <- runif(100, 0, 1)
x <- as.numeric(u < 0.3) * rgamma(100, shape = 1, scale = 5) +
  as.numeric(u > 0.3) * rgamma(100, shape = 5, scale = 1)
#pdf("/Users/YaqiongYao/Dropbox/5361proj/Figure/plot1.pdf")
hist(x, breaks = 20, probability = TRUE, main = "")
lines(density(x), lty = 1, lwd = 3, col = "red")
legend("topright", lty = 1, col = "red", "Density of x")
```



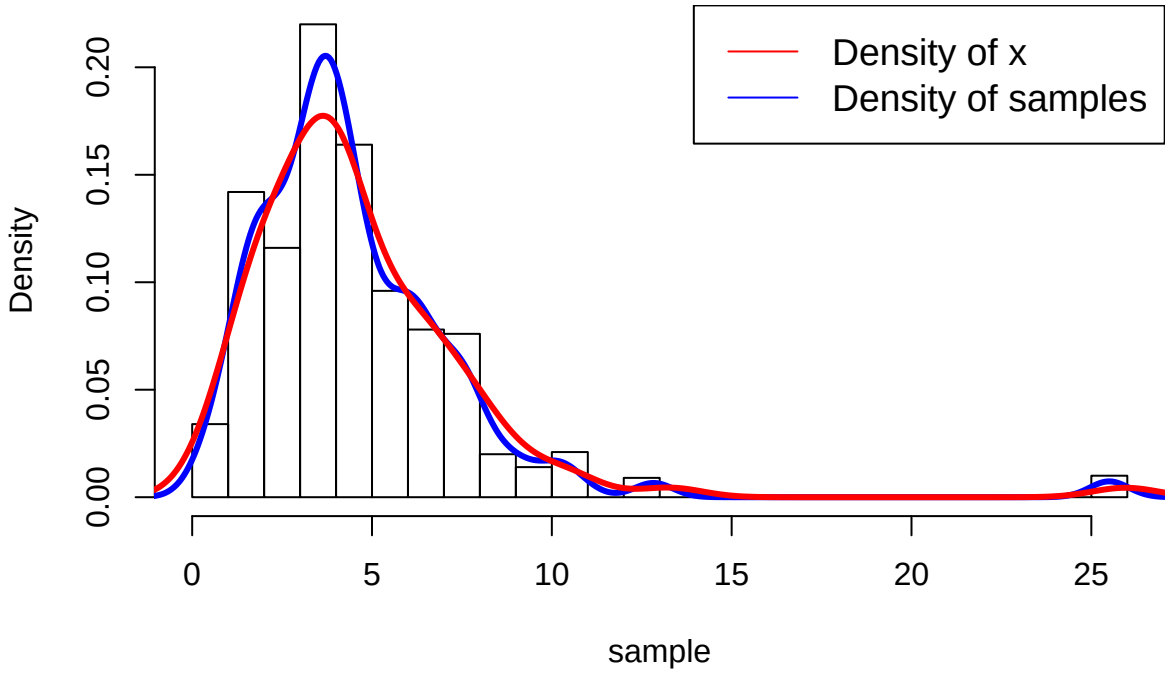

```
#dev.off()
```

Generate sample by estimated CDF by using KDE

Uniform Kernel

```
sample.kernel <- function(x, u, adj){
  y <- sort(x)
  bw <- adj*density(x)$bw
```

```r
  interval <- cbind(y-bw/2, y+bw/2)
  int.point <- NULL
  for (i in 1:dim(interval)[1]) {
    num.s <- sum((interval[1:i, 1] <= interval[i, 1]) &
                   (interval[1:i, 2] >= interval[i, 1]))
    num.e <- sum((interval[(i):dim(interval)[1], 1] <= interval[i, 2]) &
                   (interval[(i):dim(interval)[1], 2] >= interval[i, 2])) - 1
    num.m <- matrix(c(interval[i, 1], interval[i, 2], num.s, num.e), nrow = 2)
    int.point <- rbind(int.point, num.m)
  }
  int.point <- int.point[order(int.point[,1]),]
  int.point <- cbind(int.point, c((int.point[2:dim(int.point)[1], 1] - int.point[1:(dim(int.point)[1]-1
                                   int.point[1:(dim(int.point)[1]-1), 2]*(1/bw)*(1/length(x)), 0))
  int.point <- cbind(int.point, c(0,cumsum(int.point[,3])[-dim(int.point)[1]]))
  num.int <- sapply(1:length(u), function(i) sum(int.point[,4] < u[i]))
  sample <- int.point[num.int, 1] + (u - int.point[num.int, 4])/(int.point[num.int, 2]*(1/bw))
  return(sample)
}

N <- 1000
set.seed(123)
u <- runif(N, 0, 1)
sample <- sample.kernel(x, u, adj = 1)
#pdf("/Users/YaqiongYao/Dropbox/5361proj/Figure/sample1.pdf")
hist(sample, breaks = 20, probability = TRUE, main = "")
lines(density(sample), col = "blue", lty = 1, lwd = 3)
lines(density(x), col = "red", lty = 1, lwd = 3)
legend("topright", lty = c(1,1), col = c("red", "blue"), c("Density of x", "Density of samples"), cex =
```

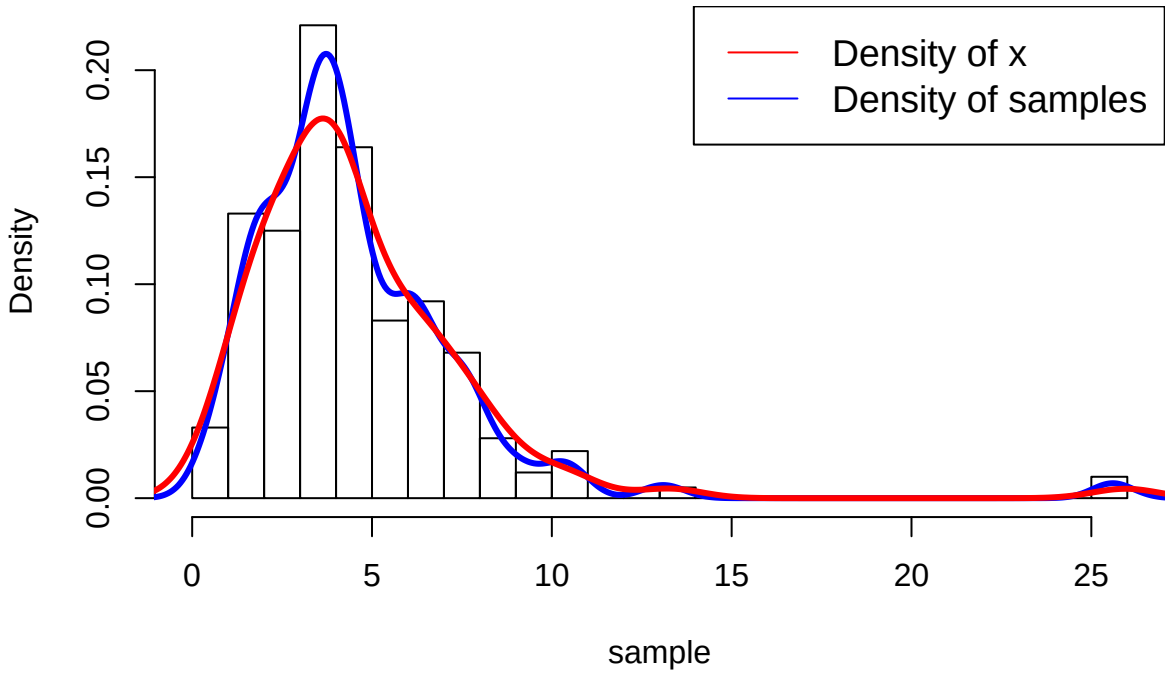



```r
#dev.off()
```

Triangular Kernel

```
sample.kernel.t <- function(x, u){
  y <- sort(x)
  bw <- density(x)$bw
  interval <- cbind(y-bw/2, y+bw/2)
  int.point <- NULL
  for (i in 1:dim(interval)[1]) {
    num.s <- sum((interval[1:i, 1] <= interval[i, 1]) &
                   (interval[1:i, 2] >= interval[i, 1]))
    m.num.s <- cbind(interval[i, 1], y[(i-num.s+1):i])
    num.y <- sum(abs(y-y[i]) <= bw/2)
    m.num.y <- cbind(y[i], y[which(abs(y-y[i]) <= bw/2)])
    num.e <- sum((interval[(i):dim(interval)[1], 1] <= interval[i, 2]) &
                   (interval[(i):dim(interval)[1], 2] >= interval[i, 2])) - 1
    m.num.e <- if(num.e != 0){
      cbind(interval[i, 2], y[(i+1): (i+num.e)])
    } else {
      cbind(interval[i, 2], NA)
    }
    int.point <- rbind(int.point, m.num.s, m.num.y, m.num.e)
  }
  int.point <- int.point[order(int.point[,1]),]
  int.point <- cbind(int.point, 0)
  int.point[int.point[,2]-int.point[,1] > 0,3] <- 1L
  int.point[int.point[,2]-int.point[,1] <= 0,3] <- -1L

  int.point <- cbind(int.point, (2*bw - 4*abs(int.point[,2] - int.point[,1]))/(bw^2))
  int.point.s <- as.data.frame(table(int.point[,1]))
  int.point.s[,1] <- as.numeric(levels(int.point.s[,1]))[int.point.s[,1]]
  j <- 0
  a <- NULL
  for (i in seq_along(int.point.s[,1])) {
    m.tmp <- matrix(int.point[(j + 1):(j + int.point.s[i,2]),], nrow = int.point.s[i,2])
    if(!is.na(m.tmp[1,4])){
      a <- c(a, sum((2/bw^2)*(int.point.s[(i+1),1] - int.point.s[i,1]) *
                      (bw - (2*m.tmp[,2] - int.point.s[(i+1),1] - int.point.s[i,1])*m.tmp[,3])))
    } else {
      a <- c(a, 0)
    }
    j <- j + int.point.s[i,2]
  }

  int.point.s <- cbind(int.point.s, a/length(x))
  int.point.s <- cbind(int.point.s, c(0,cumsum(int.point.s[,3])[-dim(int.point.s)[1]]))
  colnames(int.point.s) <- NULL
  num.int <- sapply(1:length(u), function(i) sum(int.point.s[,4] < u[i]))
  u1 <- u - int.point.s[[4]][num.int]
  sample <- NULL
  for (i in 1:length(u)) {
    m.tmp <- matrix(int.point[(sum(int.point.s[[2]][1:(num.int[i]-1)])+1):sum(int.point.s[[2]][1:num.in
                    ncol = 4)
    sum <- sum(m.tmp[,4])
```

```
    n.s <- sum(m.tmp[,3])
    if(n.s >0){
      sample[i] <- -sum/(4*n.s/bw^2) + sqrt(sum^2 + 8*n.s*u1[i]/bw^2)/(4*n.s/bw^2) + m.tmp[1,1]
    } else if(n.s < 0){
      sample[i] <- sum/(4*n.s/bw^2) - sqrt(sum^2 - 8*n.s*u1[i]/bw^2)/(4*n.s/bw^2) + m.tmp[1,1]
    } else {
      sample[i] <- u1[i]/sum + m.tmp[1,1]
    }
  }
  return(sample)
}
N <- 1000
set.seed(123)
u <- runif(N, 0, 1)
sample <- sample.kernel.t(x, u)
#pdf("/Users/YaqiongYao/Dropbox/5361proj/Figure/sample2.pdf")
hist(sample, breaks = 20, probability = TRUE, main = "")
lines(density(sample), col = "blue", lty = 1, lwd = 3)
lines(density(x), col = "red", lty = 1, lwd = 3)
legend("topright", lty = c(1,1), col = c("red", "blue"), c("Density of x", "Density of samples"), cex =
```




```
#dev.off()
```

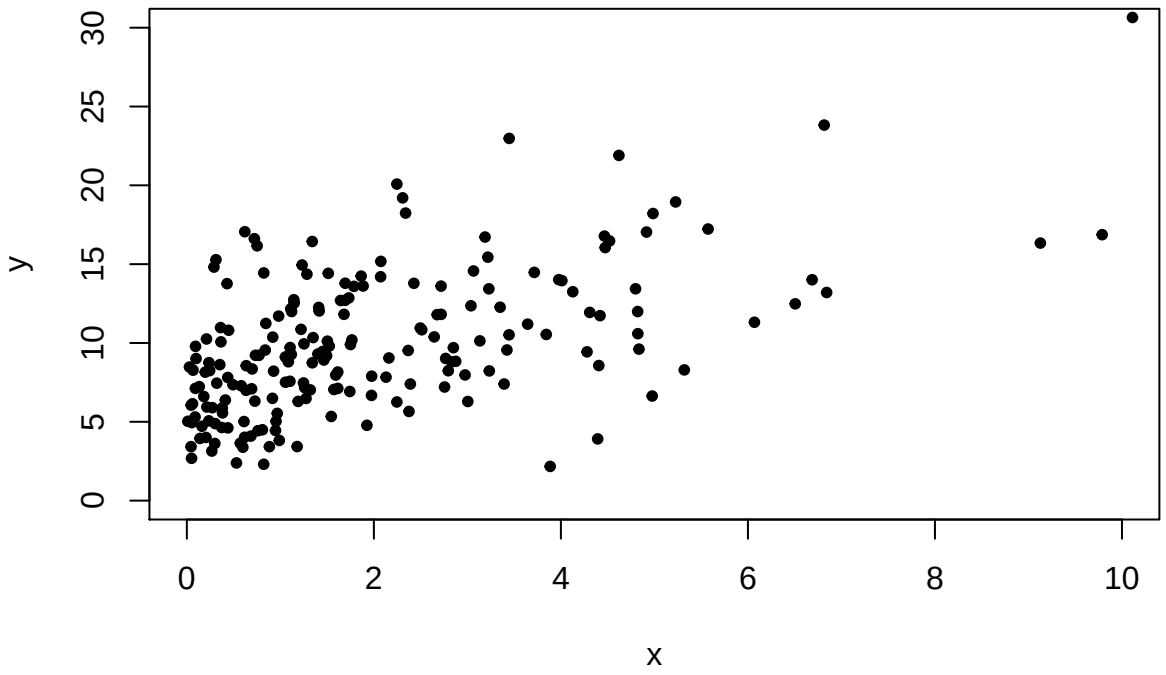
Multivariate Case

Generate Dataset

```
require(mvtnorm)
```

```
## Loading required package: mvtnorm
```

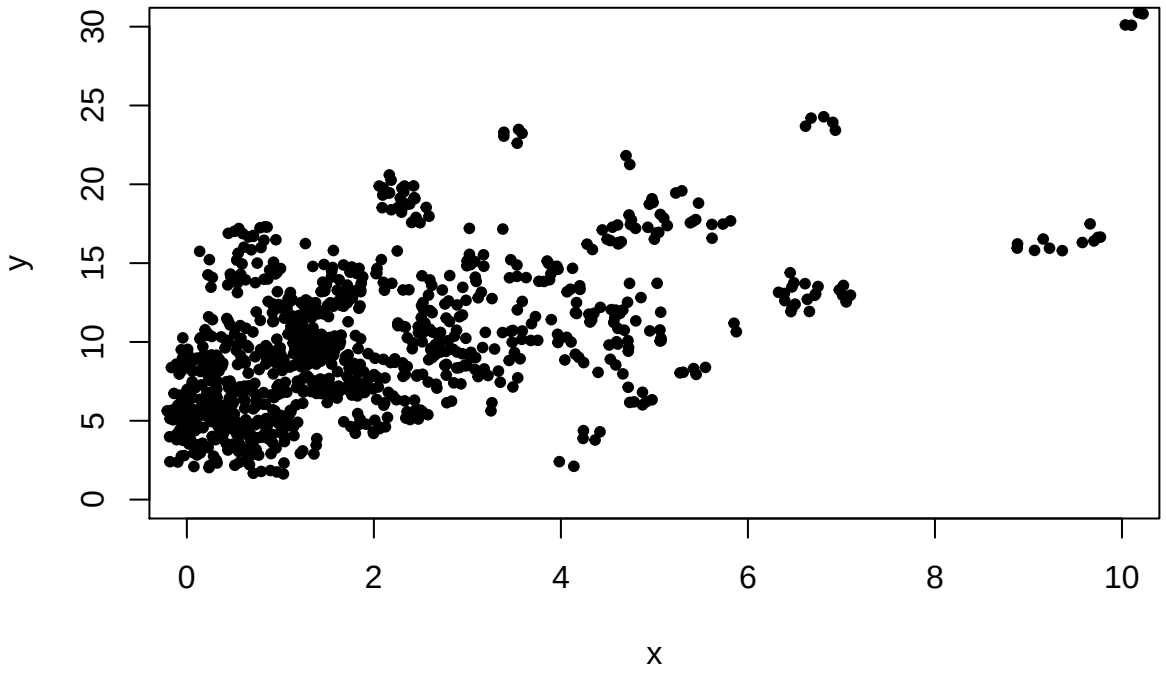
```r
n <- 200
sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
set.seed(098)
dat <- rmvnorm(n, sigma = sigma)
dat <- pnorm(dat)
x <- qgamma(dat[,1], shape = 1, scale = 2)
y <- qchisq(dat[,2], df = 10)
#pdf("/Users/YaqiongYao/Dropbox/5361proj/Figure/plot2.pdf")
plot(x, y, pch = 20, xlim = c(0, 10), ylim = c(0, 30))
```




```r
#dev.off()
```

Using method proposed by whuber (2018)

```r
sample.kernel.w <- function(n, x, y, adj){
  bw_x <- adj * density(x)$bw/2
  bw_y <- adj * density(y)$bw/2
  ind <- sample(1:length(x), n, replace = TRUE)
  x_kde <- x[ind] + runif(n, -bw_x, bw_x)
  y_kde <- y[ind] + runif(n, -bw_y, bw_y)
  cbind(x_kde, y_kde)
}
adj = 1
N <- 1000
set.seed(23)
xy_kde <- sample.kernel.w(N, x, y, adj = 1)
#pdf("/Users/YaqiongYao/Dropbox/5361proj/Figure/sample_w.pdf")
plot(xy_kde[,1], xy_kde[,2], xlim = c(0, 10), ylim = c(0, 30), pch = 20, xlab = "x", ylab = "y")
```

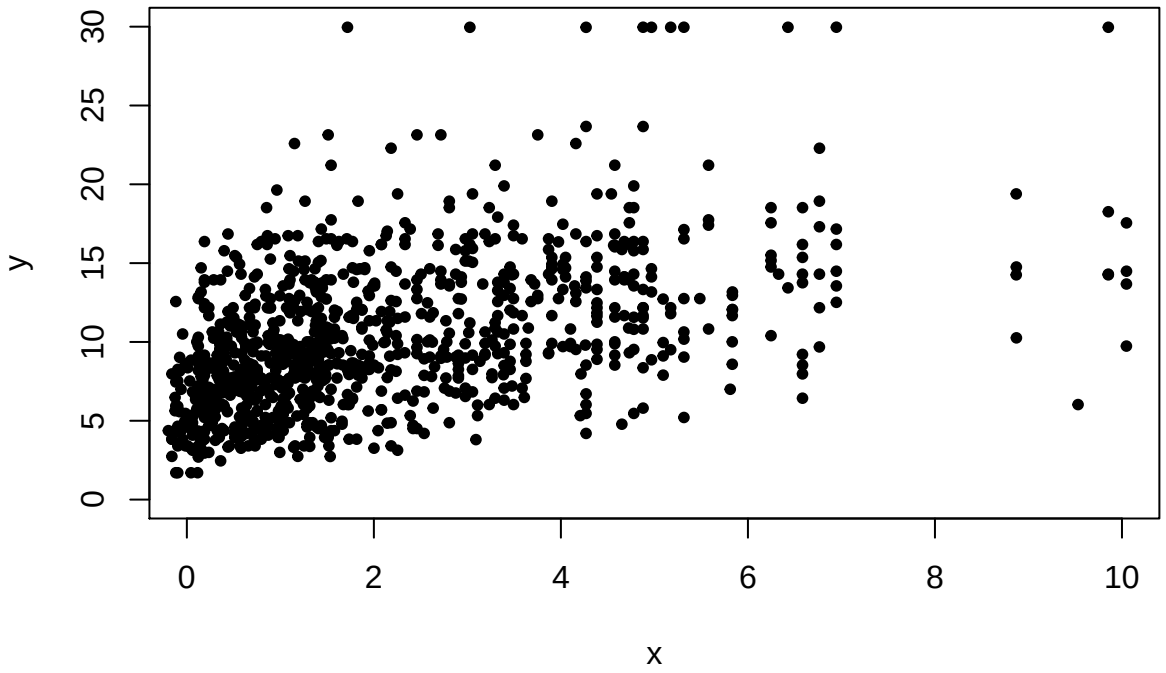



```
#dev.off()
```

Generate Samples by estimated CDF

Uniform Kernel

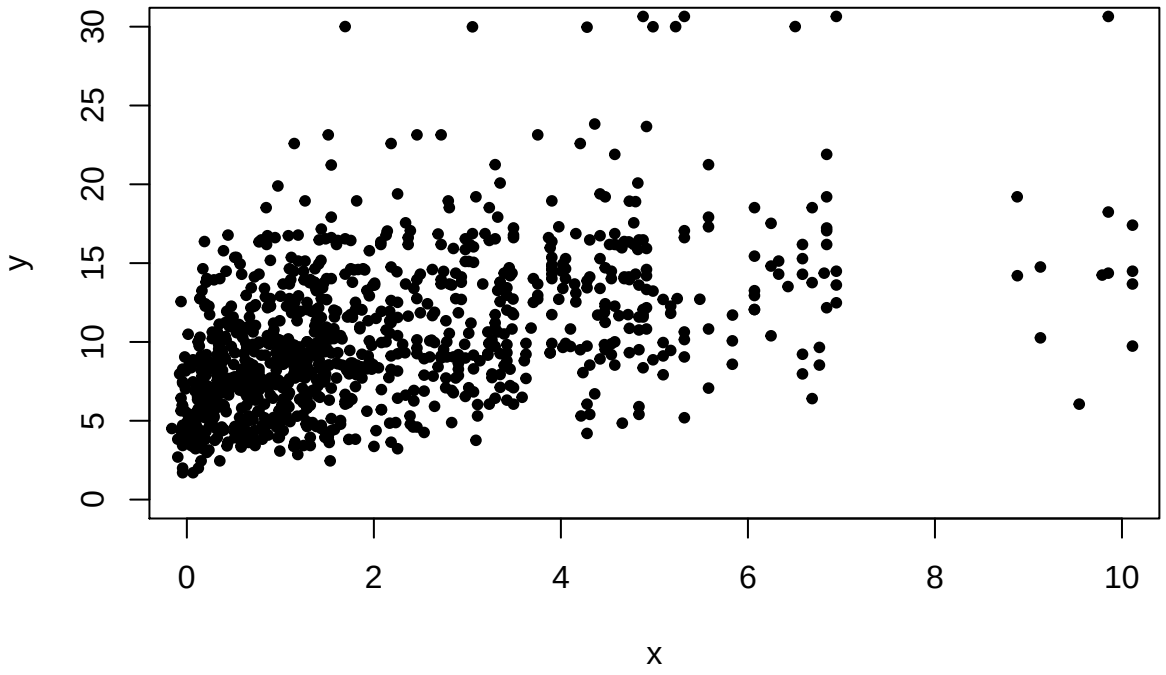
```
n <- 1000
sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
set.seed(456)
dat <- rmvnorm(n, sigma = sigma)
dat <- pnorm(dat)

x_ecdf <- sample.kernel(x, dat[,1], adj = 1)
y_ecdf <- sample.kernel(y, dat[,2], adj = 1)
#pdf("/Users/YaqiongYao/Dropbox/5361proj/Figure/sample_u.pdf")
plot(x_ecdf, y_ecdf, xlim = c(0, 10), ylim = c(0, 30), pch = 20, xlab = "x", ylab = "y")
```




```
#dev.off()
```

Triangular Kernel

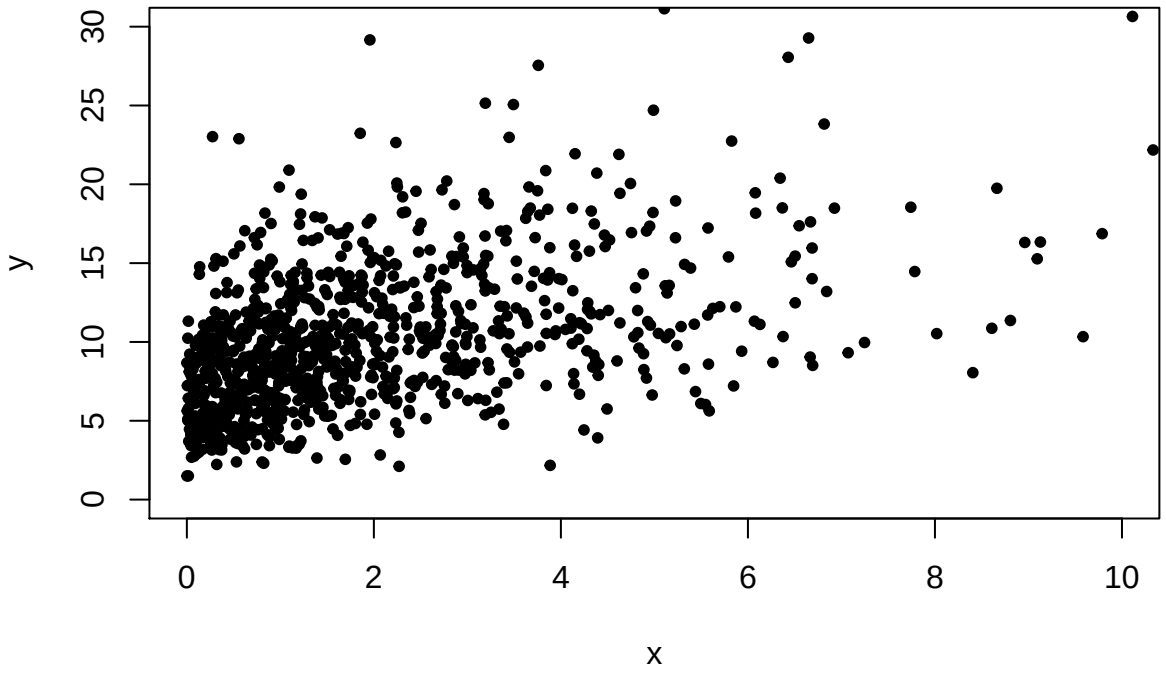
```
n <- 1000
sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
set.seed(456)
dat <- rmvnorm(n, sigma = sigma)
dat <- pnorm(dat)

x_ecdf <- sample.kernel.t(x, dat[,1])
y_ecdf <- sample.kernel.t(y, dat[,2])
#pdf("/Users/YaqiongYao/Dropbox/5361proj/Figure/sample_t.pdf")
plot(x_ecdf, y_ecdf, xlim = c(0, 10), ylim = c(0, 30), pch = 20, xlab = "x", ylab = "y")
```




```
#dev.off()
```

Generate samples from true density

```
require(mvtnorm)
n <- 1000
sigma <- matrix(c(1, 0.5, 0.5, 1), nrow = 2)
set.seed(098)
dat <- rmvnorm(n, sigma = sigma)
dat <- pnorm(dat)
x <- qgamma(dat[,1], shape = 1, scale = 2)
y <- qchisq(dat[,2], df = 10)
#pdf("/Users/YaqiongYao/Dropbox/5361proj/Figure/sample_true.pdf")
plot(x, y, pch = 20, xlim = c(0, 10), ylim = c(0, 30))
```




```
#dev.off()
```