

Comparison study 1 - -0.9 Table 4a

2024-07-24

```
#####  
##### LIBRARIES #####  
#####  
options(scipen = 999)  
#options(digits = 22)  
library(sda)  
  
## Loading required package: entropy  
## Loading required package: corpcor  
## Loading required package: fdrtool  
library(statmod)  
library(dlbayes)  
library(GIGrvg)  
library(fitdistrplus)  
  
## Loading required package: MASS  
## Loading required package: survival  
library(invgamma)  
library(extraDistr)  
  
##  
## Attaching package: 'extraDistr'  
  
## The following objects are masked from 'package:invgamma':  
##  
##     dinvchisq, dinvgamma, pinvchisq, pinvgamma, qinvchisq, qinvgamma,  
##     rinvchisq, rinvgamma  
library(rootSolve)  
library(pracma)  
  
##  
## Attaching package: 'pracma'  
  
## The following objects are masked from 'package:rootSolve':  
##  
##     gradient, hessian  
  
## The following objects are masked from 'package:Matrix':  
##  
##     expm, lu, tril, triu  
library(truncnorm)  
library(fGarch)  
  
## NOTE: Packages 'fBasics', 'timeDate', and 'timeSeries' are no longer
```

```

## attached to the search() path when 'fGarch' is attached.
##
## If needed attach them yourself in your R script by e.g.,
##     require("timeSeries")

library(BGLR)
library(spate)

## Loading required package: mvtnorm

library(readr)

##
## Attaching package: 'readr'

## The following object is masked from 'package:spate':
##
##     cols

library(Matrix)
library(sna)

## Loading required package: statnet.common

##
## Attaching package: 'statnet.common'

## The following objects are masked from 'package:base':
##
##     attr, order

## Loading required package: network

##
## 'network' 1.18.2 (2023-12-04), part of the Statnet Project
## * 'news(package="network")' for changes since last version
## * 'citation("network")' for citation information
## * 'https://statnet.org' for help, support, and other information

## sna: Tools for Social Network Analysis
## Version 2.7-2 created on 2023-12-05.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
## For citation information, type citation("sna").
## Type help(package="sna") to get started.

##
## Attaching package: 'sna'

## The following object is masked from 'package:pracma':
##
##     cutpoints

library(e1071)

##
## Attaching package: 'e1071'

## The following object is masked from 'package:pracma':
##
##     sigmoid

```

```

library(bayesplot)

## This is bayesplot version 1.11.1
## - Online documentation and vignettes at mc-stan.org/bayesplot
## - bayesplot theme set to bayesplot::theme_default()
##   * Does _not_ affect other ggplot2 plots
##   * See ?bayesplot_theme_set for details on theme setting
library(ggplot2)
library(rstanarm)

## Loading required package: Rcpp
## This is rstanarm version 2.32.1
## - See https://mc-stan.org/rstanarm/articles/priors for changes to default priors!
## - Default priors may change, so it's safest to specify priors, even if equivalent to the defaults.
## - For execution on a local, multicore CPU with excess RAM we recommend calling
##   options(mc.cores = parallel::detectCores())
##
## Attaching package: 'rstanarm'
## The following object is masked from 'package:pracma':
##
##   logit

#####
##### GLOBAL FUNCTIONS #####
#####
getmode <- function(v) {
  uniqv <- unique(v)
  uniqv[which.max(tabulate(match(v, uniqv)))]
}

#####
##### DATA SIMULATION #####
#####
niter = 5000
burn0 = 2000
nsim = 20
p = 100
n = 50
plots_ind = F

#Store results for sim
rsim_SN <- matrix(NA, nsim, p)
rsim_N <- matrix(NA, nsim, p)

skew0 <- -0.9
delta_0 <- sign(skew0)*sqrt(22/7/2*(abs(skew0))^(2/3)/((abs(skew0))^(2/3)+((4-22/7)/2)^(2/3)))
a <- 0.2#dlhyper(x, y)

```

```

hyper <- a #dlhyper(x, y)
a0 <- 1
b0 <- 0.9

corr_mat <- matrix(NA,nrow = p,ncol = p)

for(i in 1:p){
  for(j in 1:p){

    corr_mat[i,j] <- abs(i-j)

  }
}

c_mat <- 0.5^corr_mat
s_mat <- diag(p)

x <- {}

for(i in 1:n){

  x <- rbind(x,mvrnorm(n=1, mu=rep(0,p), Sigma=c_mat))

}

x <- as.matrix(x)
colnames(x) <- paste0("X", rep(1:p))

#####
##### LATENT STRUCTURE TO SIM Y #####
#####

Su_0<- sqrt(1-((2/pi)*(delta_0^2)))
Eu_0 <- sqrt(2/pi)*delta_0
tau_0 <- stats::rgamma(n = 1, shape = p * hyper, rate = 1/2) #can test varying rate
phi_0 <- c(LaplacesDemon::rdirichlet(n = 1, alpha = rep(hyper,p)))
psi_0 <- stats::rexp(p, rate = 1/2) #can vary the parameter
s2_0 <- 1
#s2b_0 <- mean(phi_0^2 * psi_0 * tau_0^2)

#choosing uniform prior for sigma2 and delta

beta_0 <- c(runif(10,min = 0,max = 0),
            runif(5,min = 4,max = 5),
            runif(20,min = 0,max = 0),
            runif(5,min = 4,max = 5),

```

```

runif(p-40,min = 0,max = 0))
#beta_0 <- c(rep(0,10),runif(5,min = 0.5,max = 1),rep(0,25),runif(5,min = 0.5,max = 1),rep(0,5))

for (isim in 1:nsim){
print(paste0("Simulation ", isim))
t_0 <- truncnorm::rtruncnorm(n,a = 0,b=Inf,mean = 0,sd=1)

mu_y0 <- as.matrix(x)%*%beta_0 + (sqrt(s2_0)/Su_0)*(delta_0*t_0 - Eu_0)
sd_y0 <- (s2_0*sqrt((1-delta_0)^2))/Su_0
y_0 <- rnorm(n, mean = mu_y0, sd = sd_y0)
y <- y_0
plot(density(y_0))
skewness(y)
plot(density(y - as.matrix(x)%*%beta_0))
skewness(y - as.matrix(x)%*%beta_0)

#write.csv(as.data.frame(cbind(x,y)), "~/Documents/Arno_sim1/simdat.csv", row.names = FALSE)

#simdat <- read_csv("~/Documents/Arno_sim1/simdat.csv", col_names = FALSE)

simdat = cbind(x,y)

#y <- as.matrix(simdat[,ncol(simdat)])
#x <- as.matrix(simdat[,1:(ncol(simdat)-1)])

#####
##### INITIAL VALUES AND PREAMBLE #####
#####

### SENSITIVITY ANALYSIS ###

#init val
psi_init = rep(1,p)
phi_init = rep(1,p)/p
beta_init <- rep(2,p)
t_init = rep(mean(truncnorm::rtruncnorm(100*n,a = 0,b=Inf,mean = 0,sd=1)),n)
tau_init <- 0.5*(p*a - 1)
delta_init <- 0.5
#s2e_init <- 5

psi <- psi_init
phi <- phi_init
tau <- tau_init
beta <- beta_init
t <- t_init
delta <- delta_init
s2e <- 1

beta_mat <- matrix(NA,nrow = niter, ncol = p)

```

[illegible]

```
#####
##### GIBBS SAMPLER #####
#####
```

```
for(iter in 1:niter) {
```

```
Su <- sqrt(1-((2/pi)*(delta^2)))
Eu <- sqrt(2/pi)*delta
```

```
c1 <- sum(x[,j]^2) + ((1-delta^2)*s2e)/((Su^2) *(psi[j]*(phi[j]^2)*(tau^2)))
c2 <- sum(x[,j]*y) - sum(x[,j]*x[,-j]%*%beta[-j]) - sum(x[,j]*delta*t*sqrt(s2e)/(Su)) + sum(x[,j]*E
mu_beta <- c2/c1
sig2_beta <- ((1-delta^2)*s2e)/((Su^2) * c1)
```

}

T

```
Eu <- sqrt(2/pi)*delta
```

```
sig2_t <- 1-delta^2
```

```
tm <- rbind(tm,t)
```

Delta

```
Su <- sqrt(1-(2/pi)*delta^2)
```

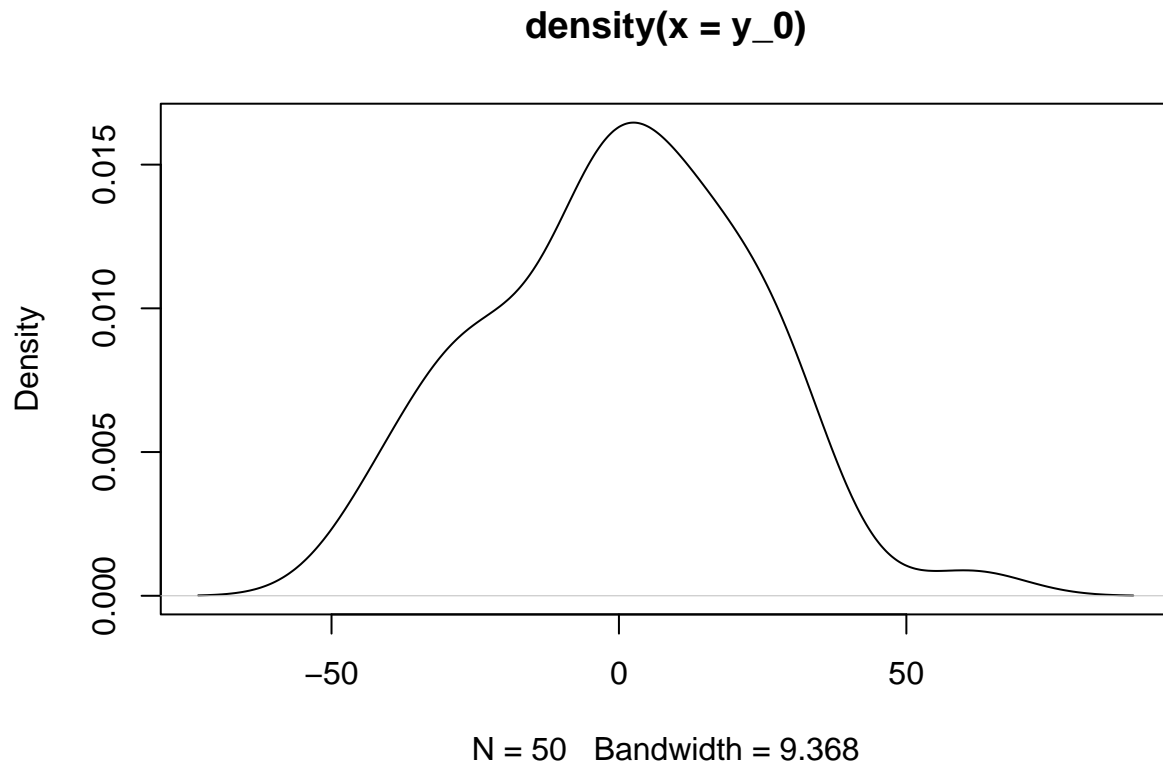


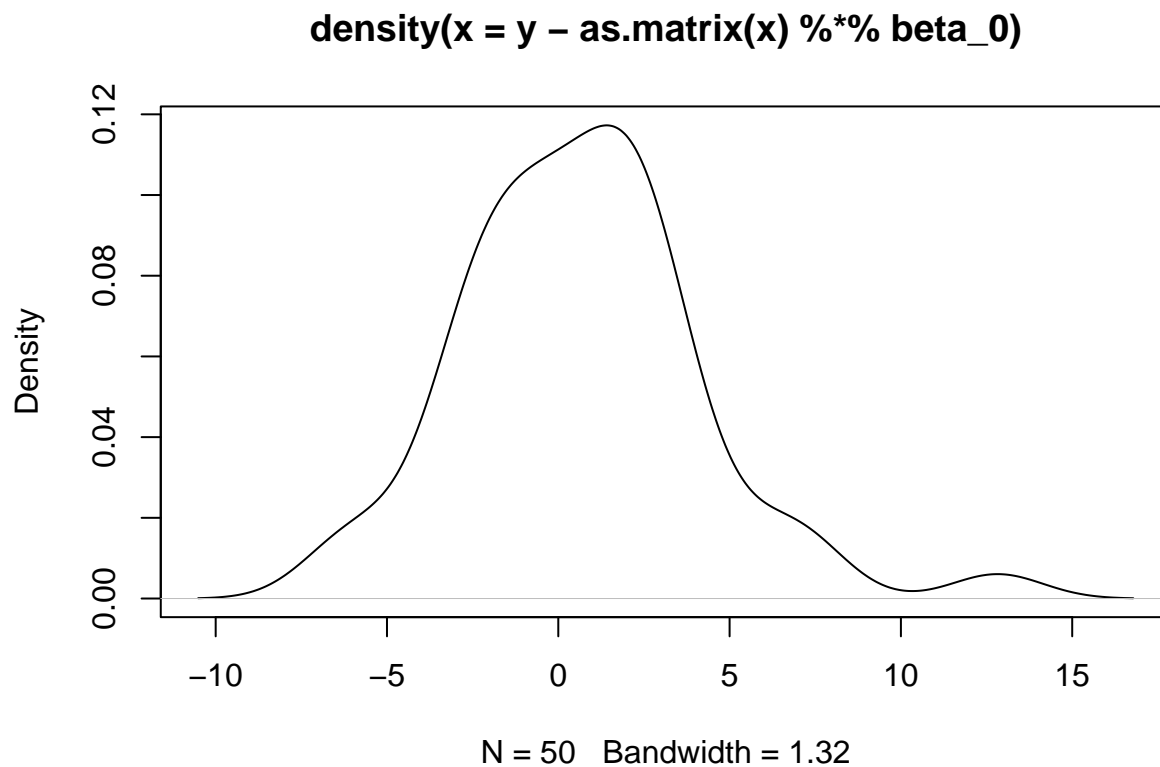
```

print("Assuming asymmetric error")
print(which(abs(postmodeSN)>1))
print("Assuming symmetric error")
print(which(abs(dlbayes::dlvs(dlresult))>0))
}

```

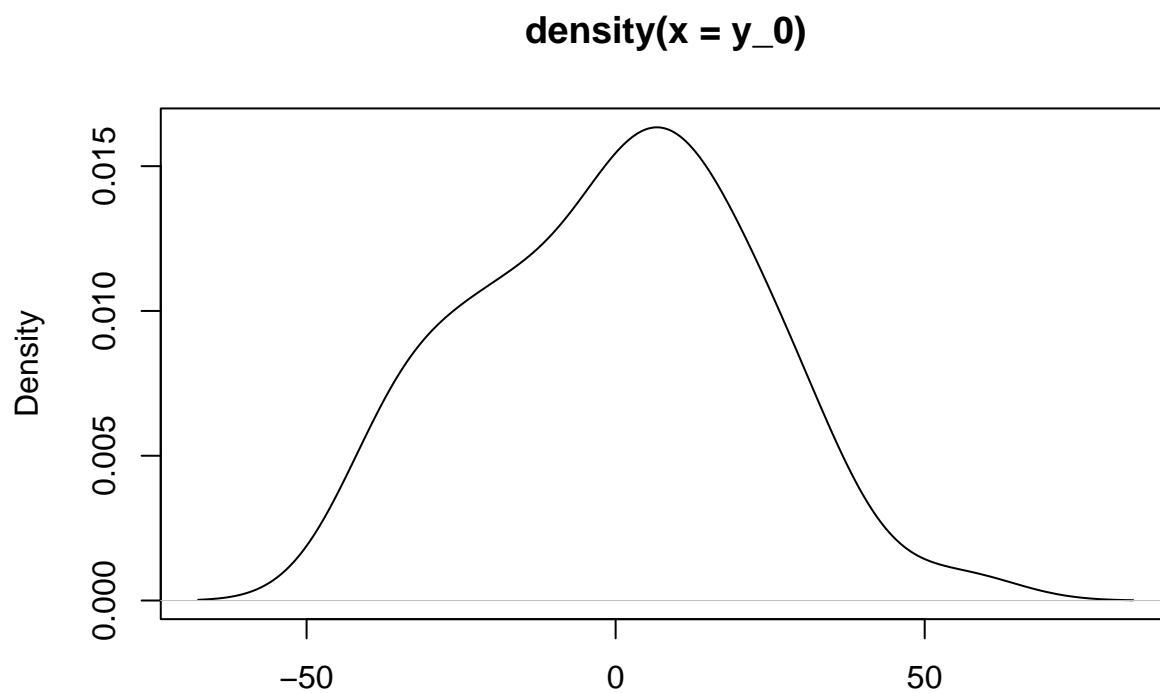
```
## [1] "Simulation 1"
```



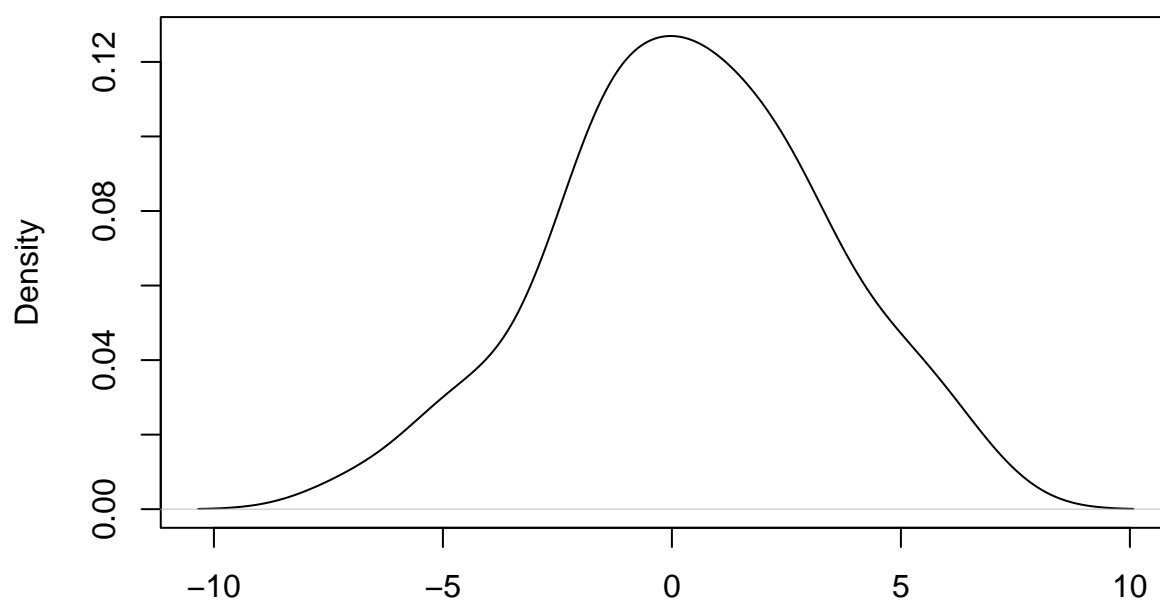


```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
```

```
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 1 11 12 13 14 15 16 17 36 37 38 39 40 48 61 82 87 96
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 2"
```



N = 50 Bandwidth = 9.194
density(x = y - as.matrix(x) %*% beta_0)



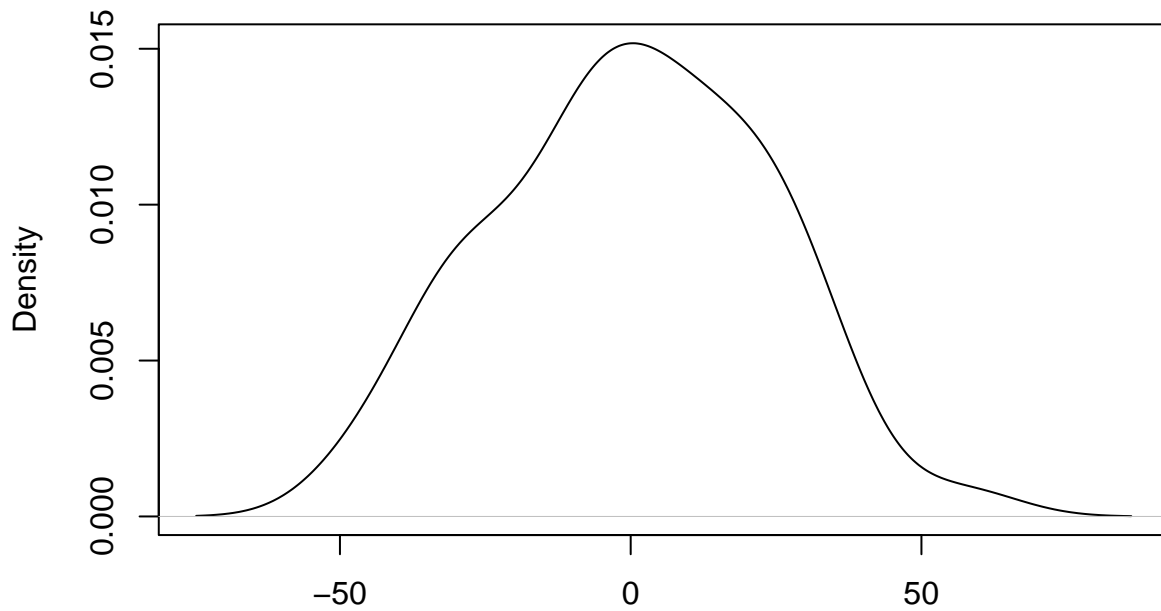
N = 50 Bandwidth = 1.152

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
```

```
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
```

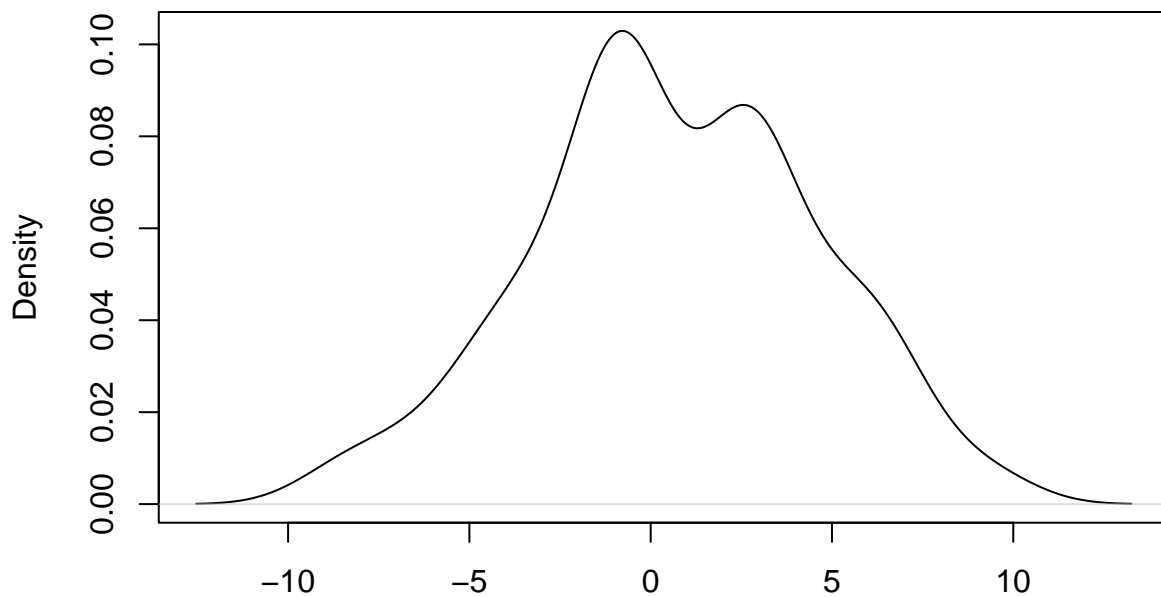
```
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 27 36 37 38 39 40 90 94 100
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 3"
```

density(x = y_0)



N = 50 Bandwidth = 9.582

density(x = y - as.matrix(x) %*% beta_0)

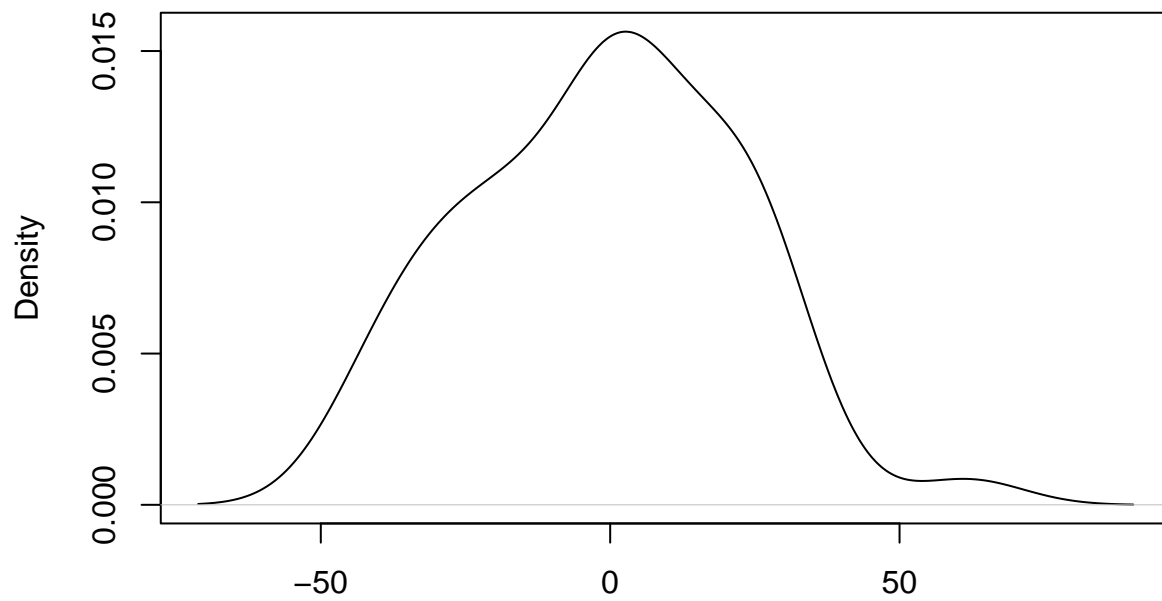


N = 50 Bandwidth = 1.308

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
```

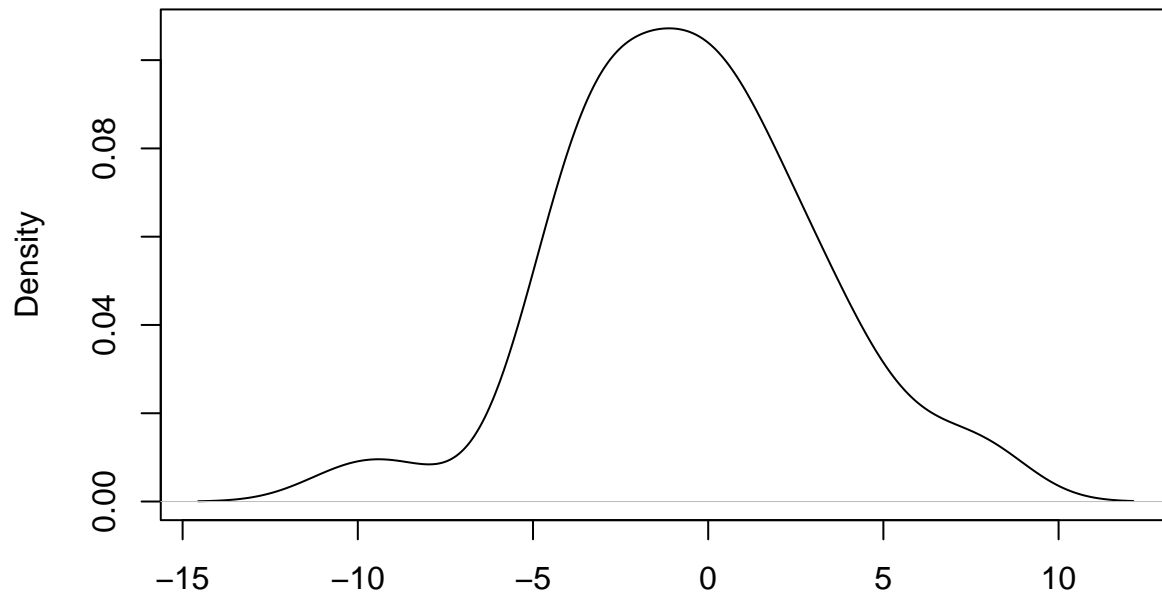
```
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 7 11 12 13 14 15 30 33 36 37 38 39 40 67 85 96 97
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 4"
```

density(x = y_0)



N = 50 Bandwidth = 9.522

density(x = y - as.matrix(x) %*% beta_0)

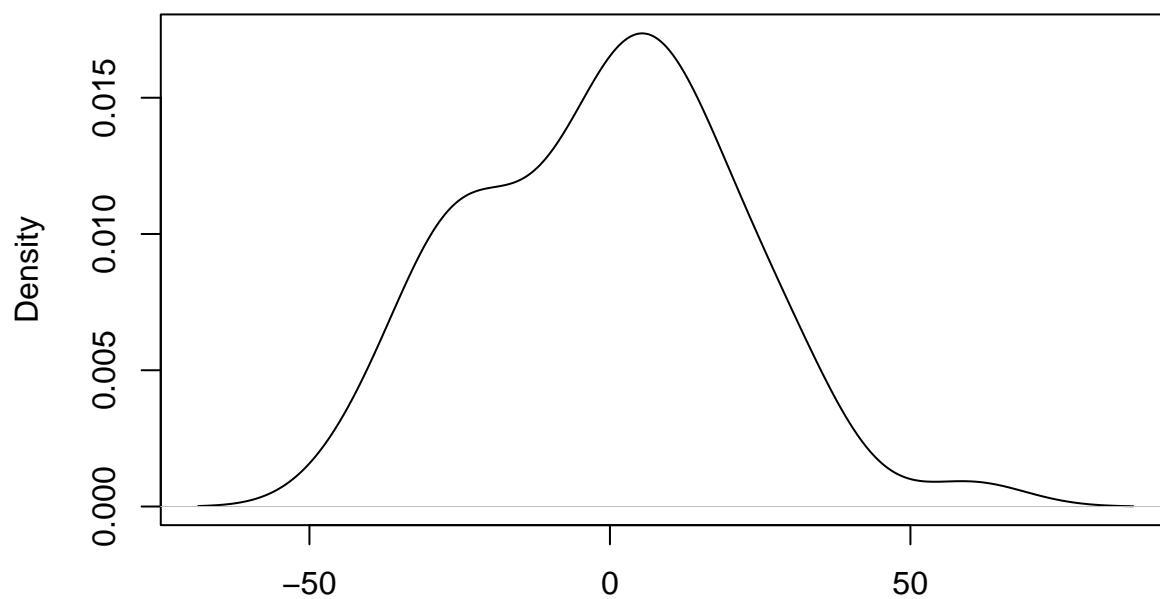


N = 50 Bandwidth = 1.409

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
```

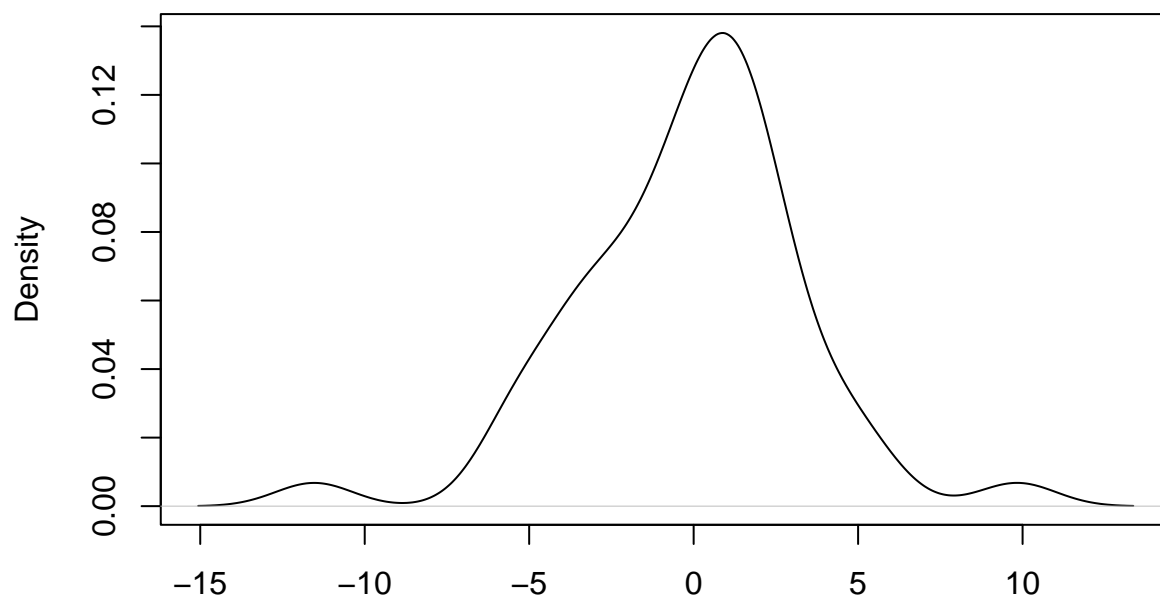
```
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 22 33 36 37 38 39 40 58 62 85
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 5"
```

density(x = y_0)



N = 50 Bandwidth = 8.985

density(x = y - as.matrix(x) %*% beta_0)



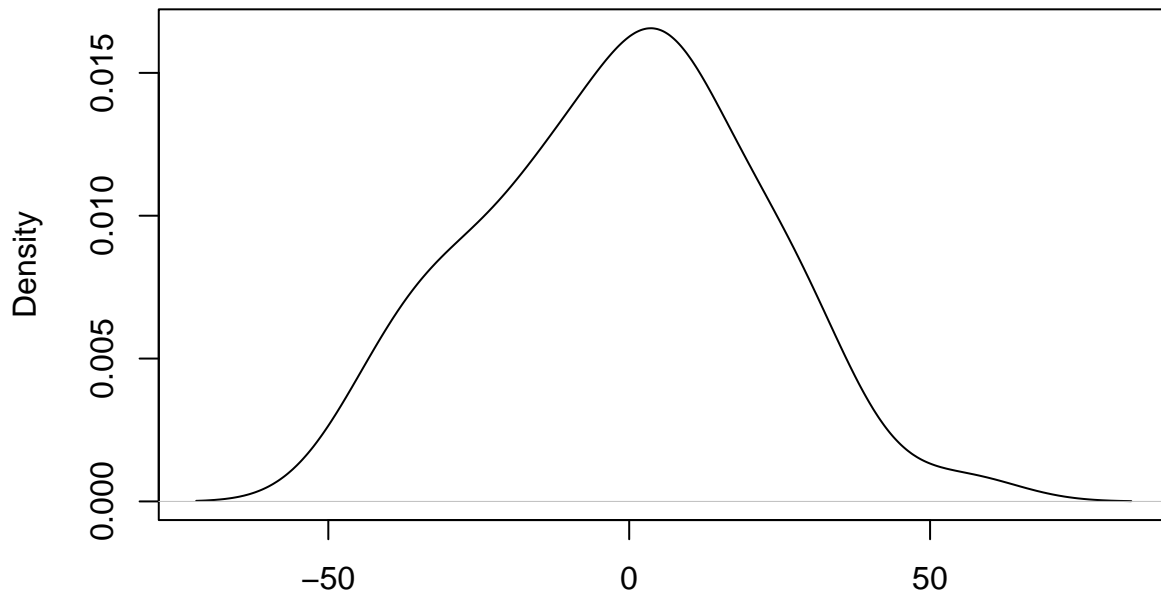
N = 50 Bandwidth = 1.176

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
```

```
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
```

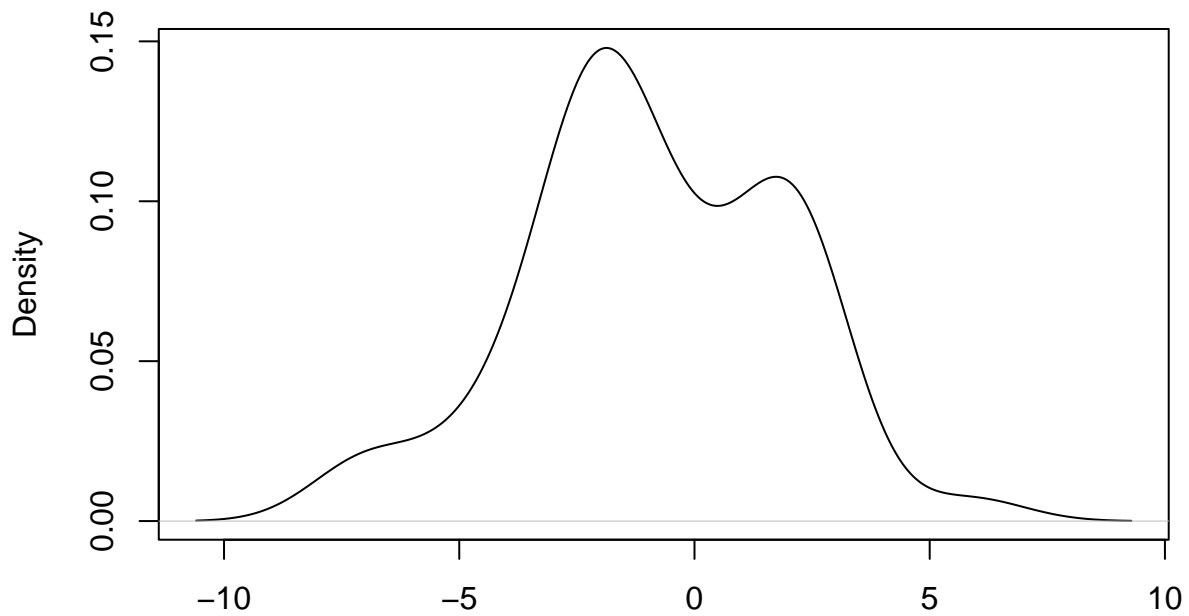
```
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 19 33 36 37 38 39 40 65 67 72 80
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 6"
```

density(x = y_0)



N = 50 Bandwidth = 9.277

density(x = y - as.matrix(x) %*% beta_0)

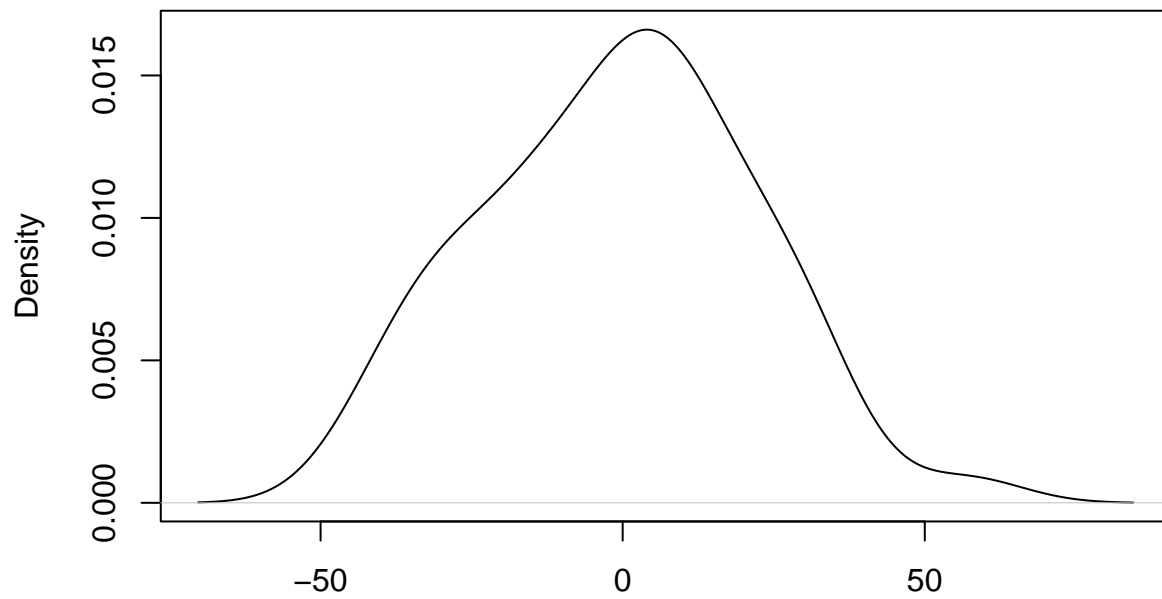


N = 50 Bandwidth = 1.133

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
```

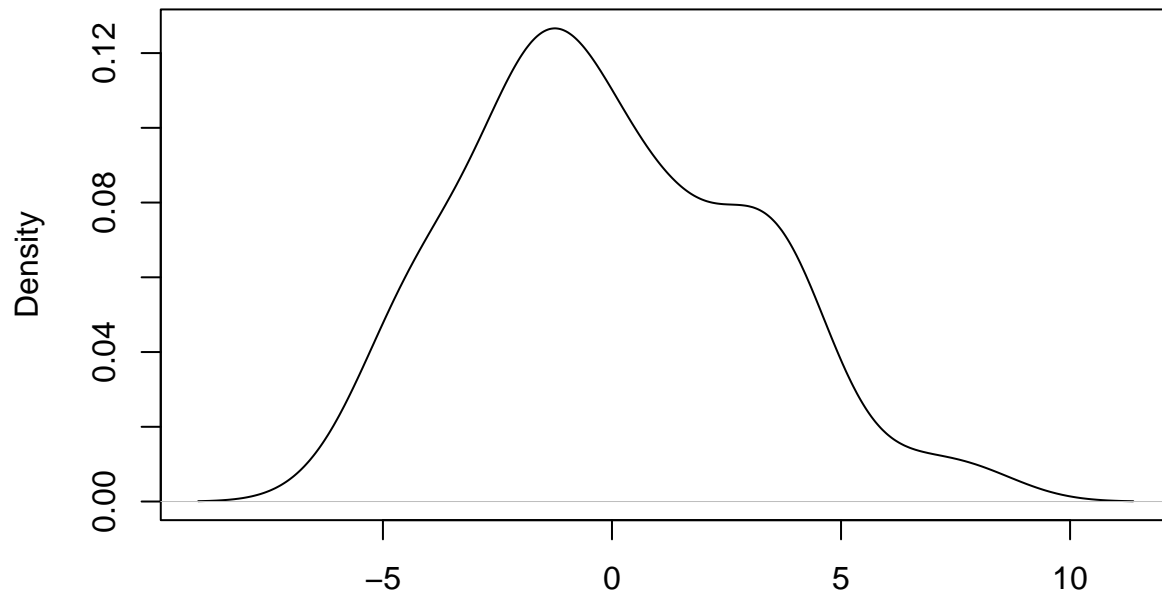
```
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40 44 55 67 71 91
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 7"
```

density(x = y_0)



N = 50 Bandwidth = 9.15

density(x = y - as.matrix(x) %*% beta_0)



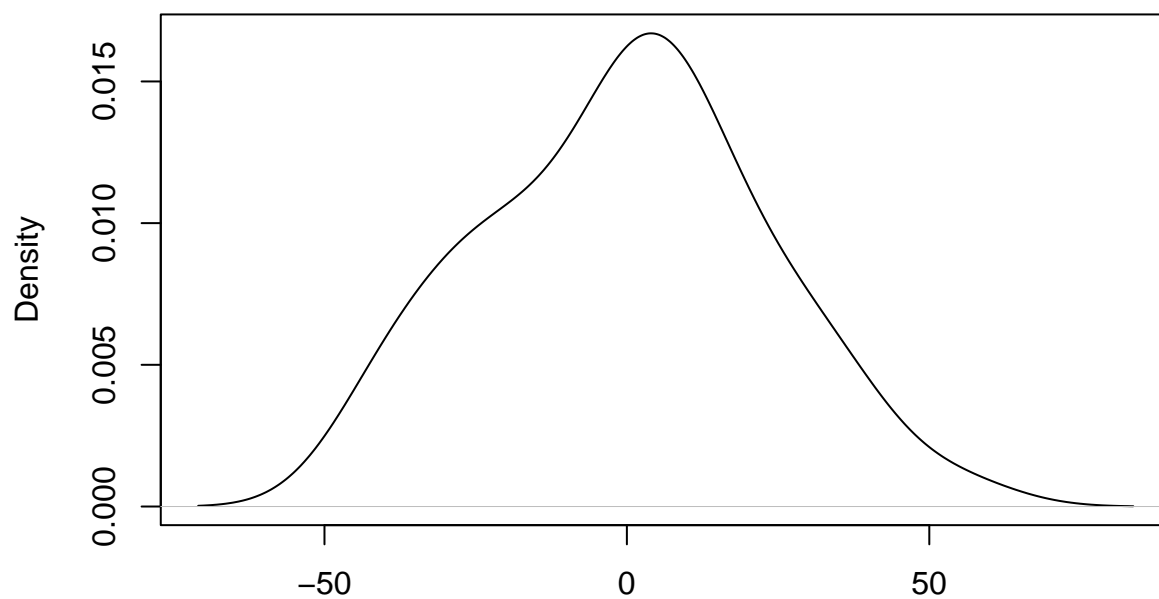
N = 50 Bandwidth = 1.167

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
```



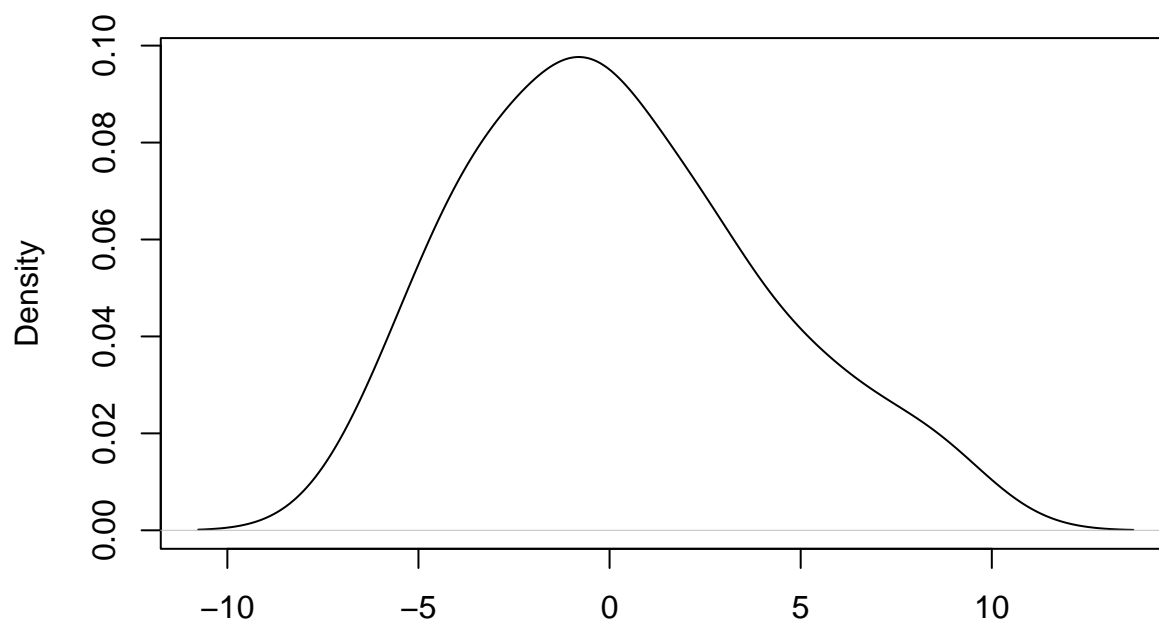
```
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40 46 47 66
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 8"
```

density(x = y_0)



N = 50 Bandwidth = 9.502

density(x = y - as.matrix(x) %*% beta_0)



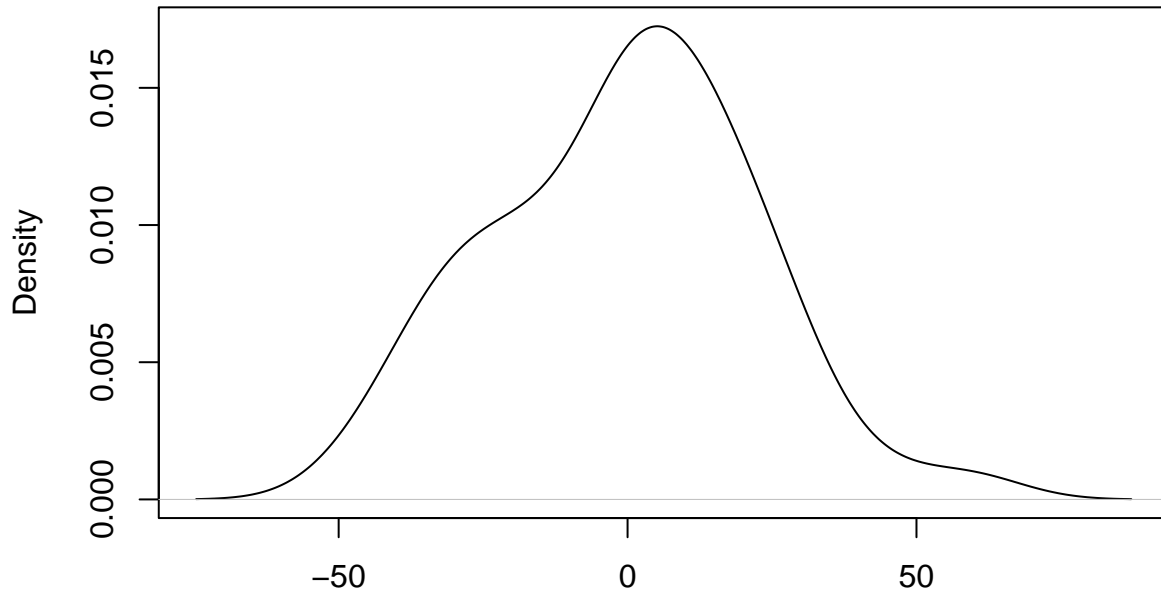
N = 50 Bandwidth = 1.581

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
```

```
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
```

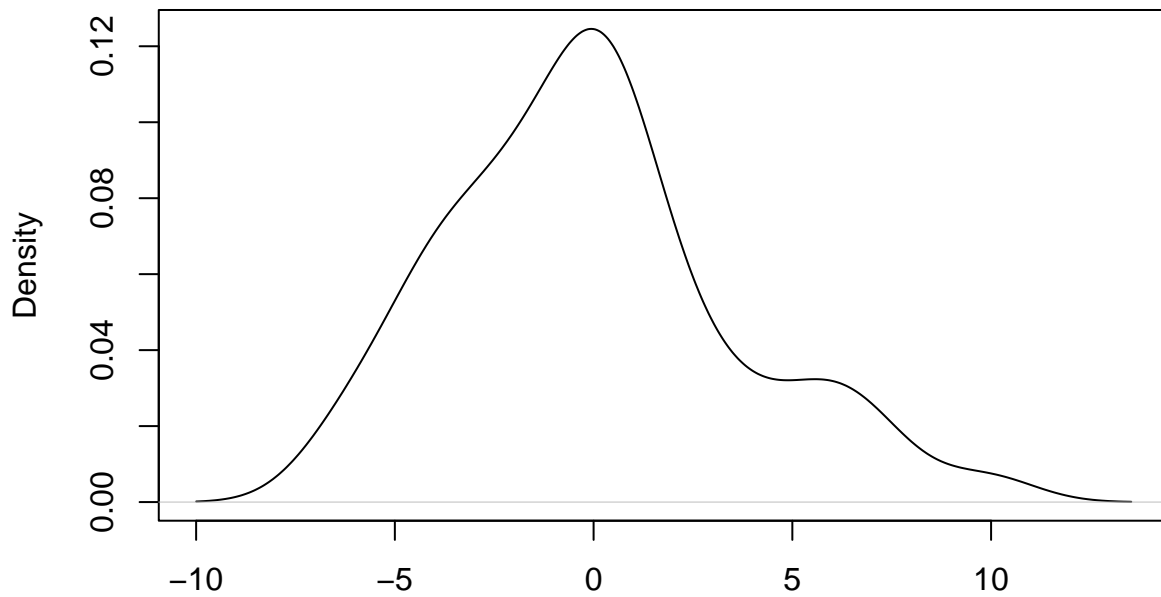
```
## [1] "Assuming asymmetric error"
## [1] 5 9 11 12 14 15 36 37 38 39 40 47 60 69 84 96
## [1] "Assuming symmetric error"
## [1] 11 12 14 15 36 37 38 39 40
## [1] "Simulation 9"
```

density(x = y_0)



N = 50 Bandwidth = 9.297

density(x = y - as.matrix(x) %*% beta_0)

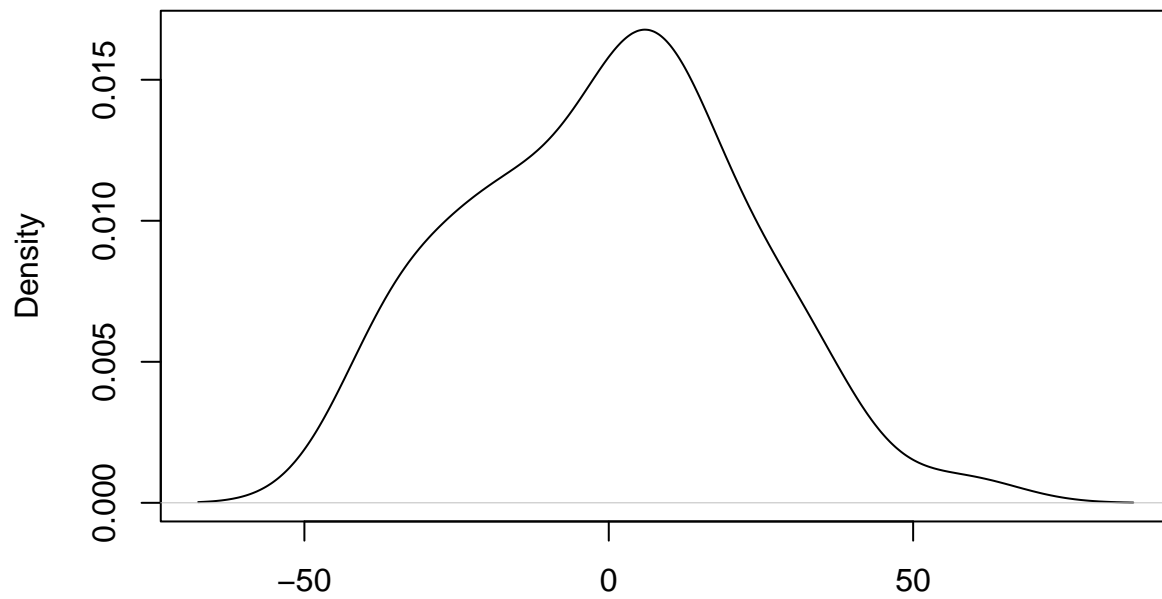


N = 50 Bandwidth = 1.205

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
```

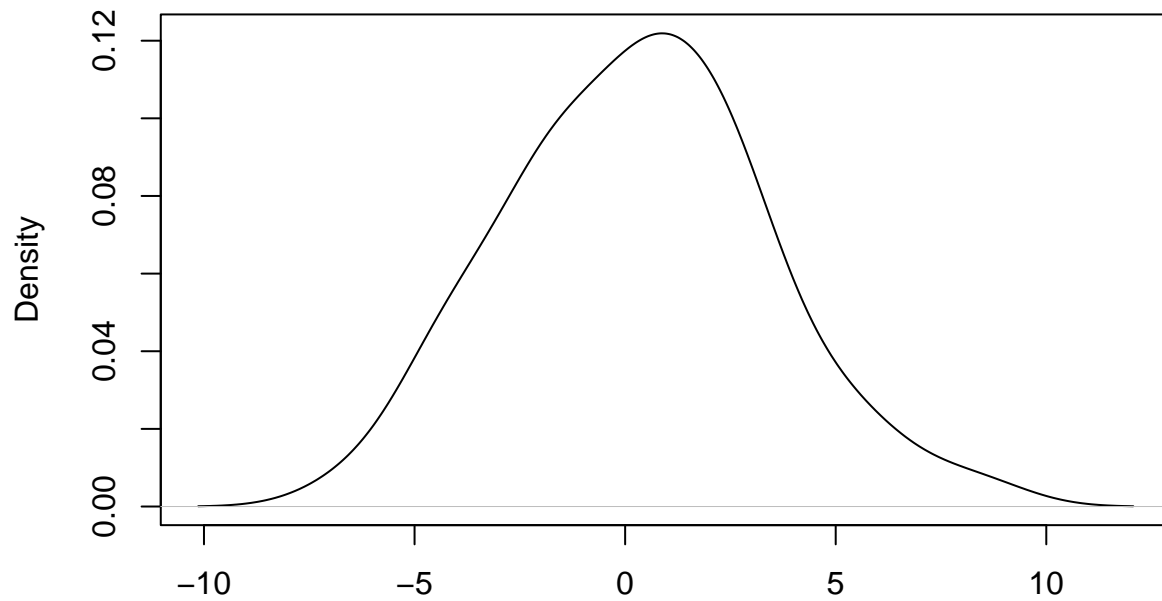
```
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 16 36 37 38 39 40 44 89
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 36 37 38 39 40
## [1] "Simulation 10"
```

density(x = y_0)



N = 50 Bandwidth = 9.279

density(x = y - as.matrix(x) %*% beta_0)



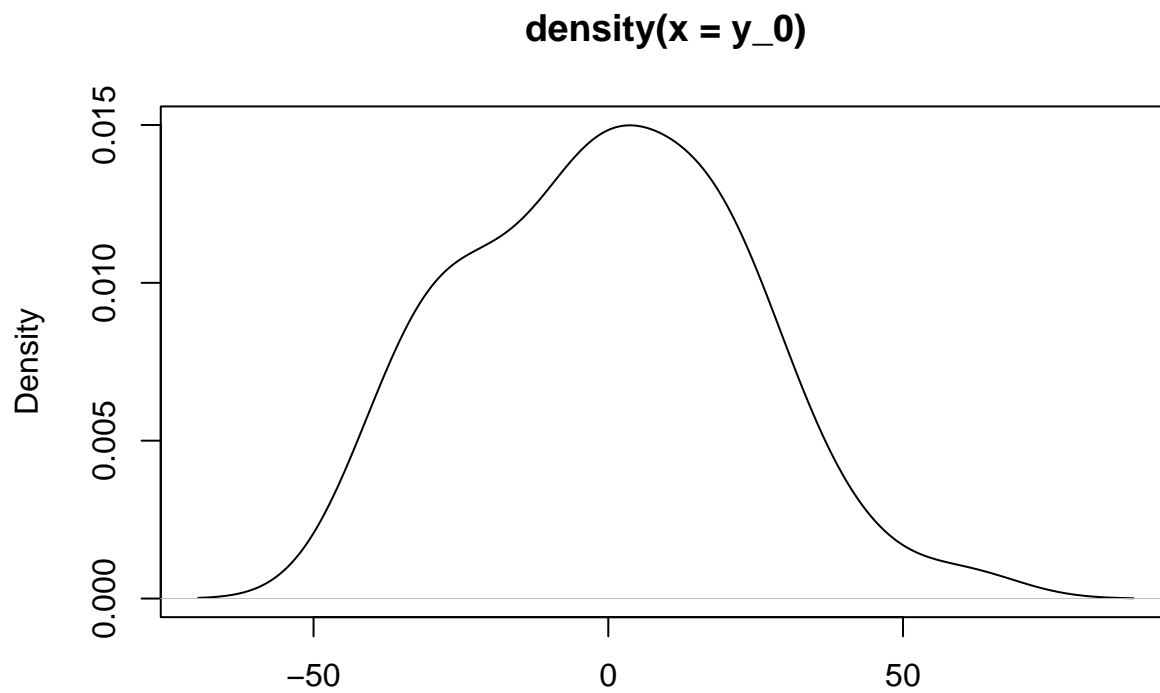
N = 50 Bandwidth = 1.245

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
```

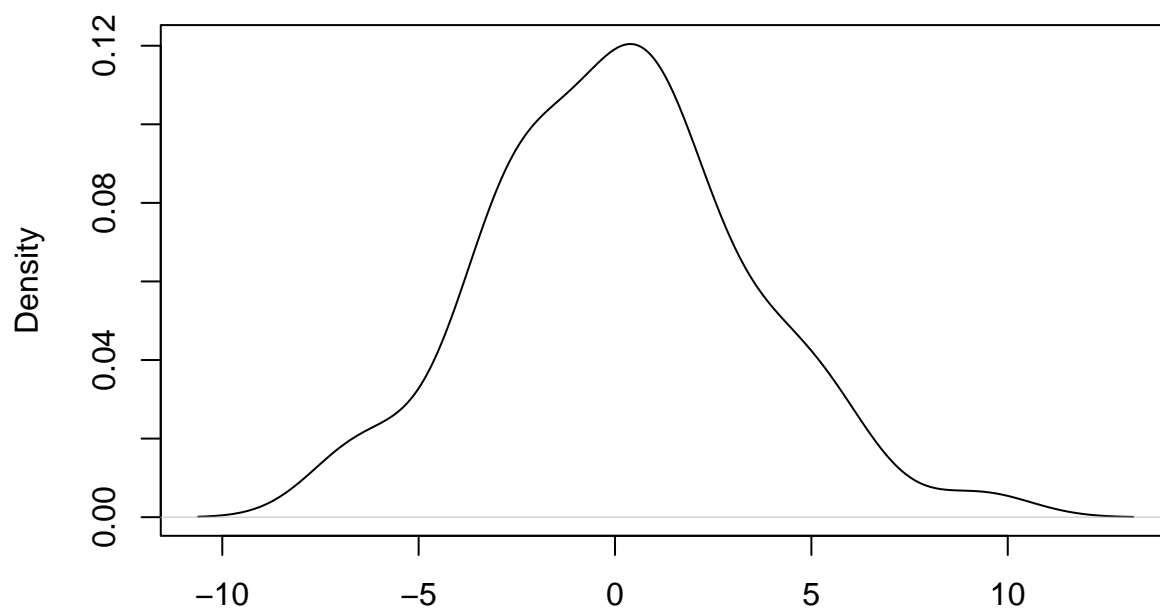
```

## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 2 11 12 13 14 15 30 36 37 38 39 40 56 69 72 73
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 11"

```

N = 50 Bandwidth = 9.562
density(x = y - as.matrix(x) %*% beta_0)



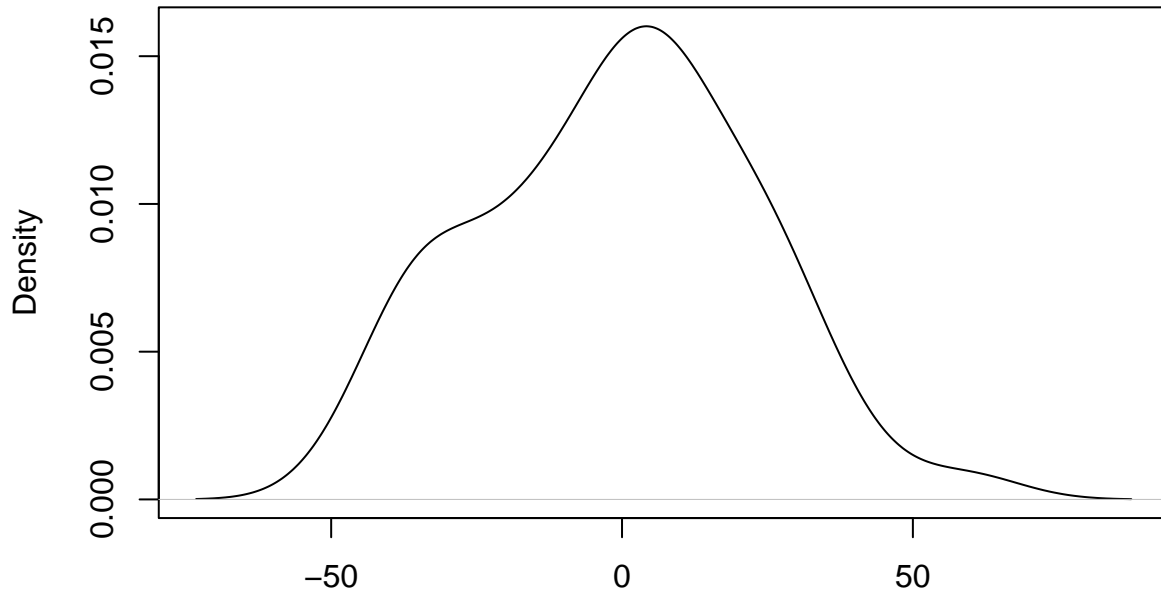
N = 50 Bandwidth = 1.296

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
```

```
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
```

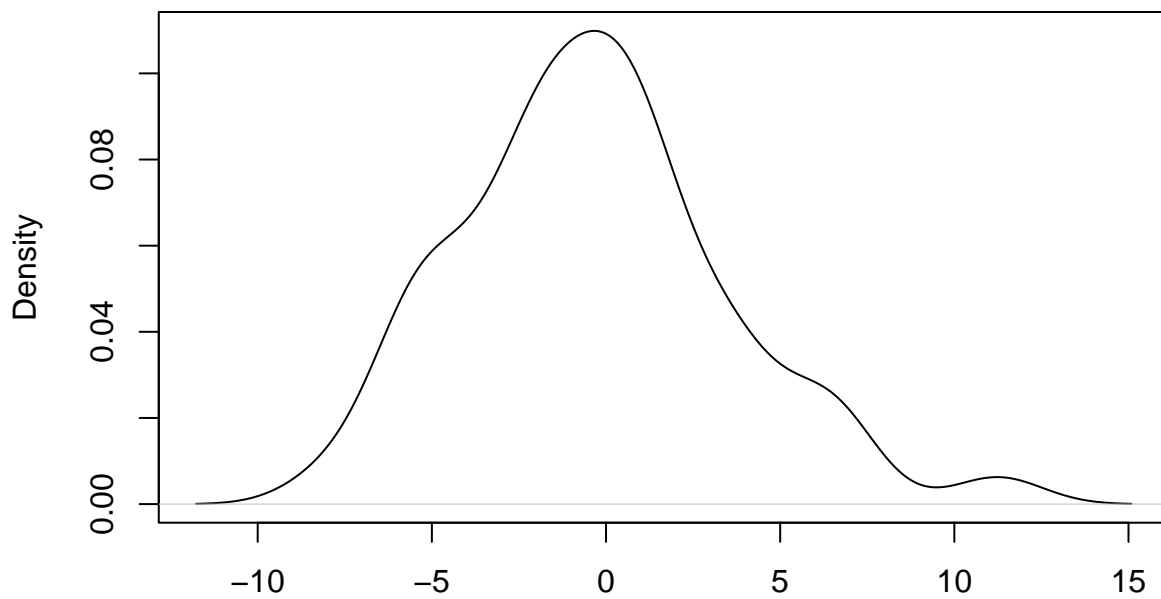
```
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40 49 79 98
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 12"
```

density(x = y_0)



N = 50 Bandwidth = 9.501

density(x = y - as.matrix(x) %*% beta_0)

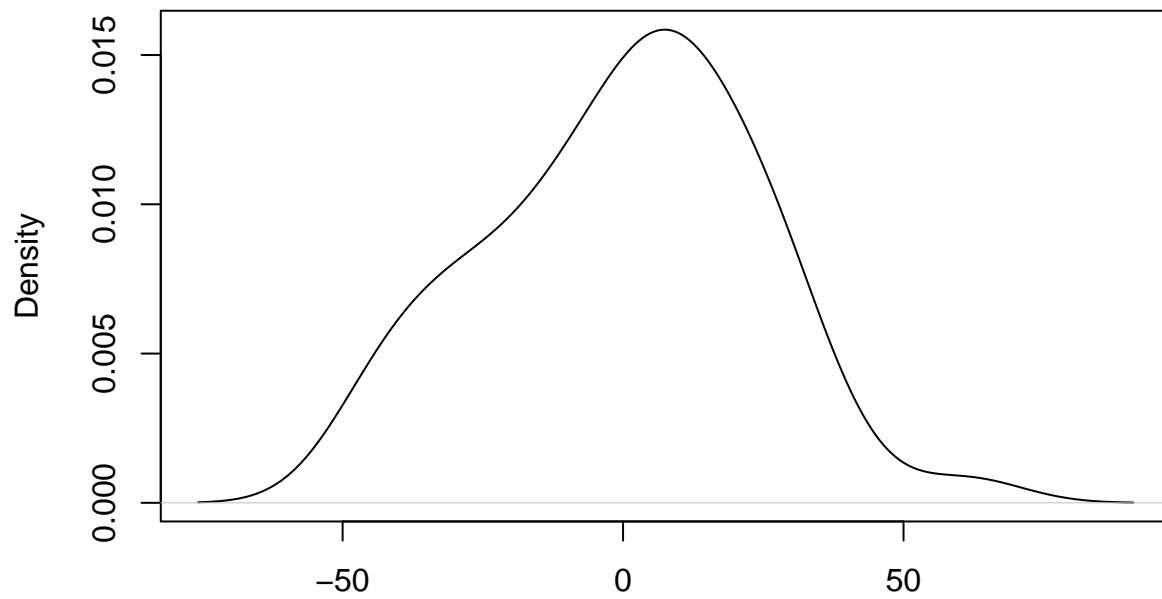


N = 50 Bandwidth = 1.278

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
```

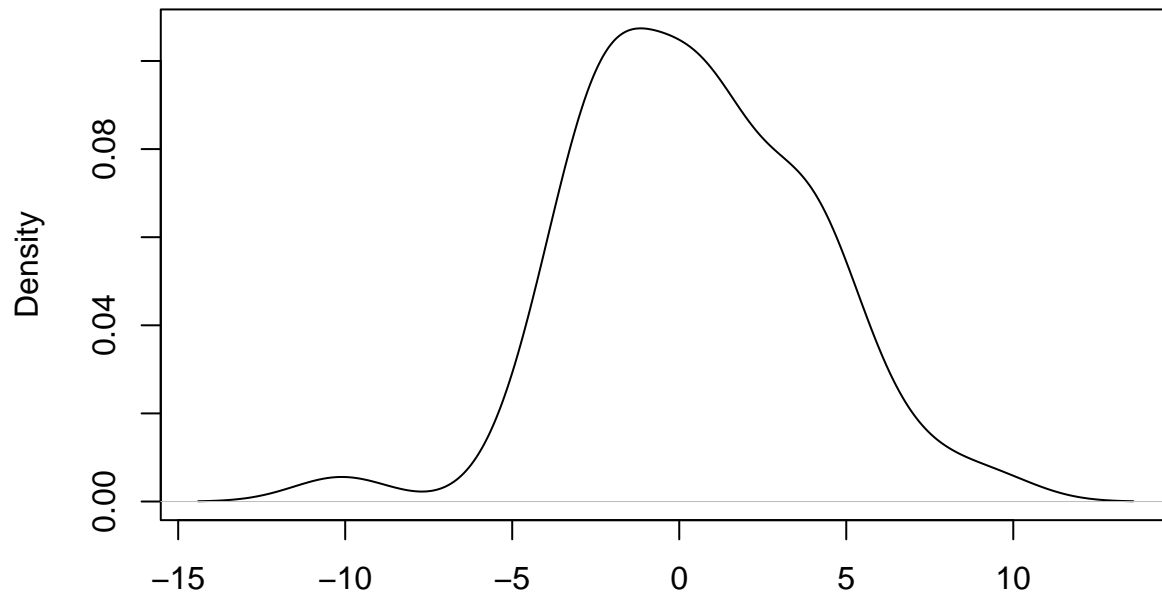
```
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 1 11 12 13 14 15 36 37 38 39 40 42 51 66 79 83 85
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 36 37 38 39 40
## [1] "Simulation 13"
```

density(x = y_0)



N = 50 Bandwidth = 9.837

density(x = y - as.matrix(x) %*% beta_0)

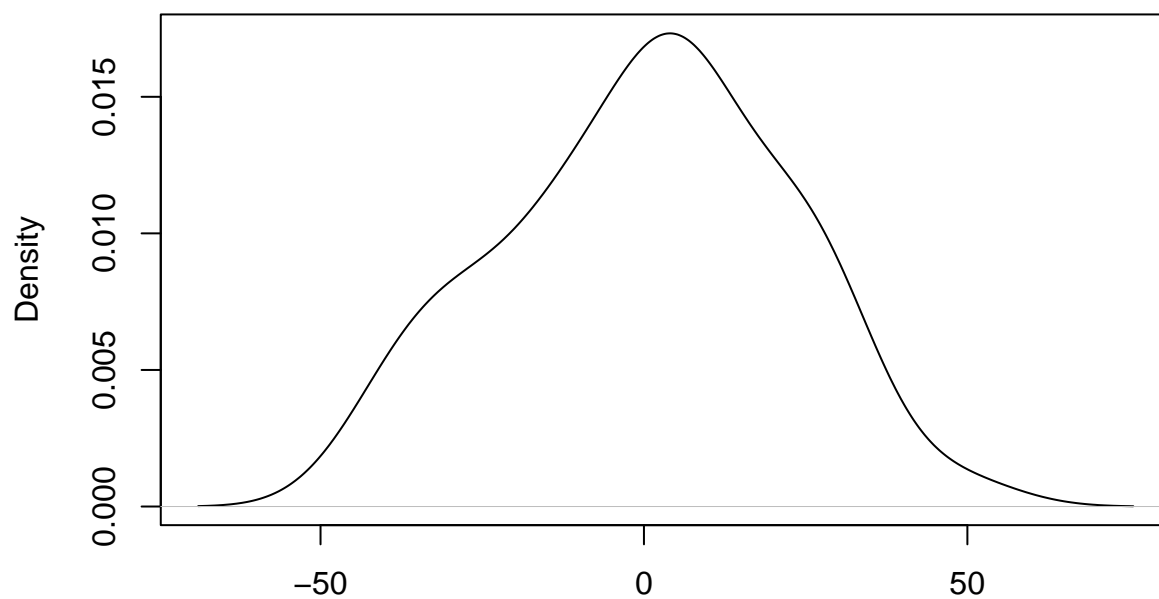


N = 50 Bandwidth = 1.432

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
```

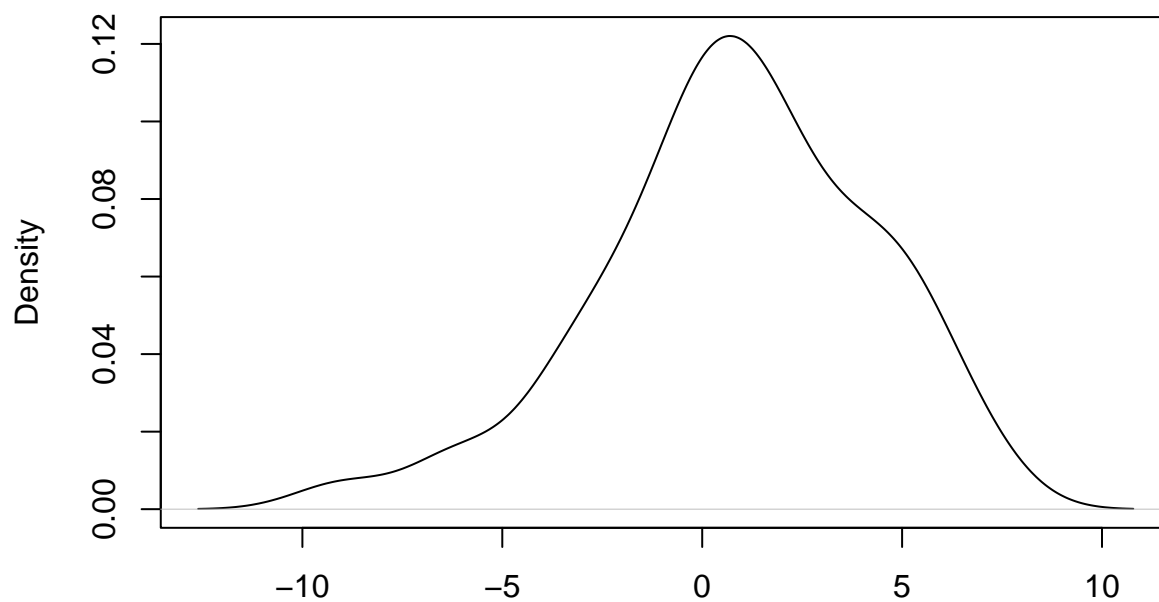
```
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 7 11 12 13 14 15 19 21 23 36 37 38 39 40 45 50 68
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 14"
```

density(x = y_0)



N = 50 Bandwidth = 8.75

density(x = y - as.matrix(x) %*% beta_0)



N = 50 Bandwidth = 1.204

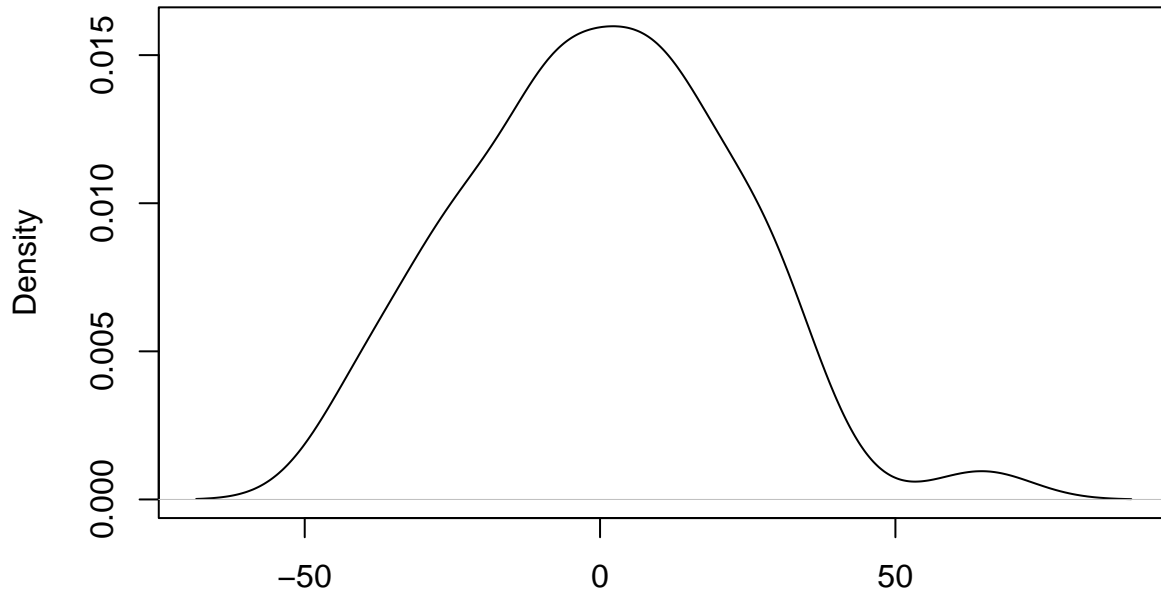
```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
```



```
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
```

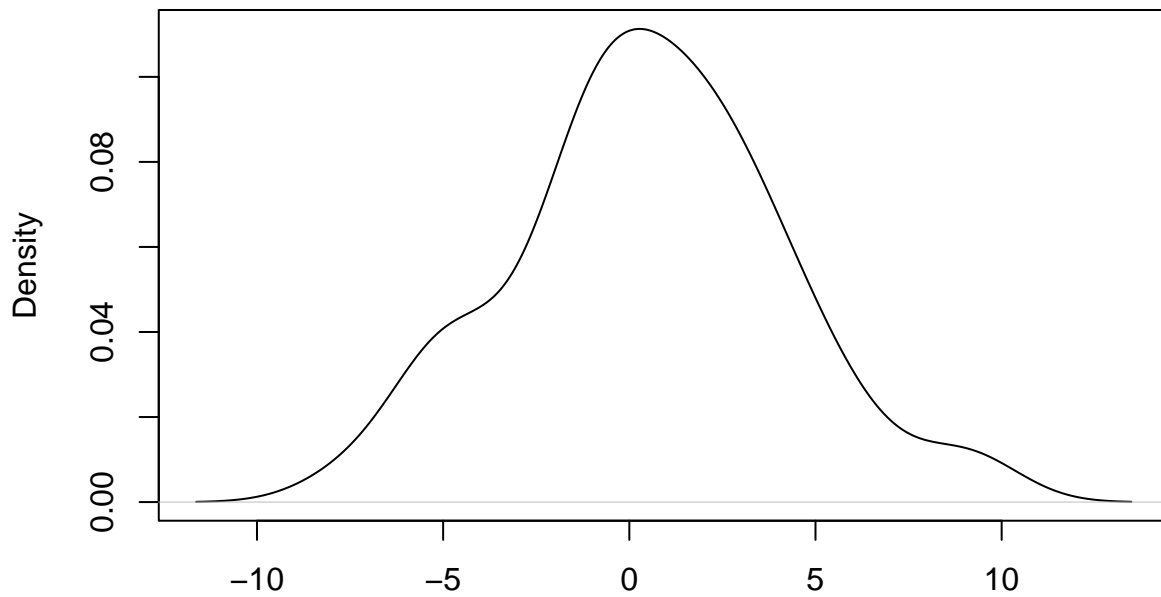
```
## [1] "Assuming asymmetric error"
## [1] 1 4 11 12 13 14 15 36 37 38 39 40 45 46 56 57 58 85 91
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 15"
```

density(x = y_0)



N = 50 Bandwidth = 8.401

density(x = y - as.matrix(x) %*% beta_0)

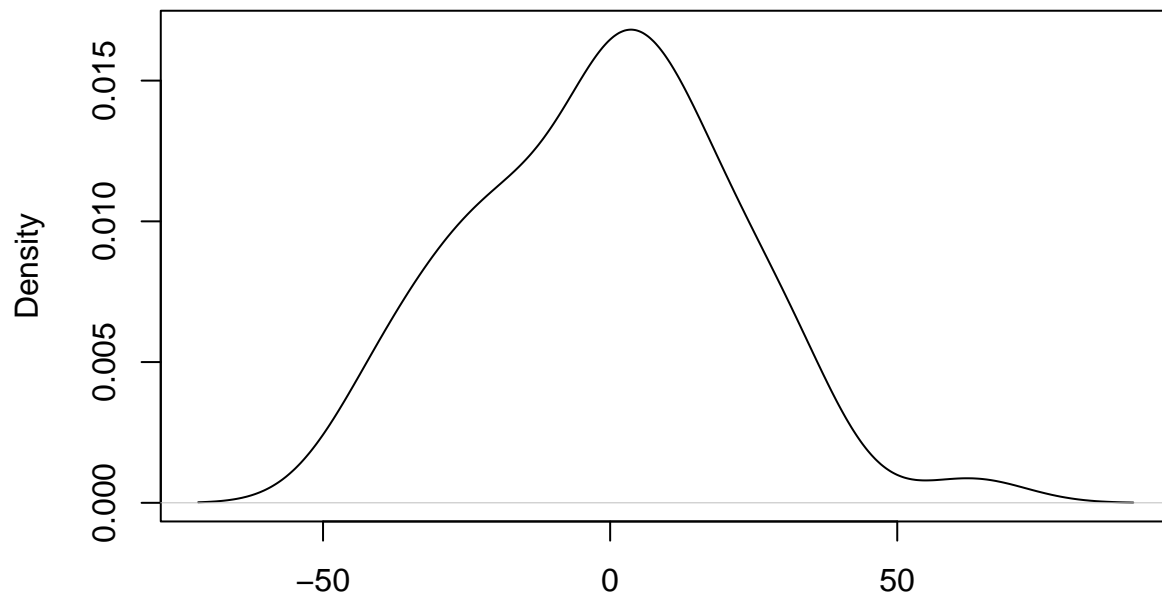


N = 50 Bandwidth = 1.358

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
```

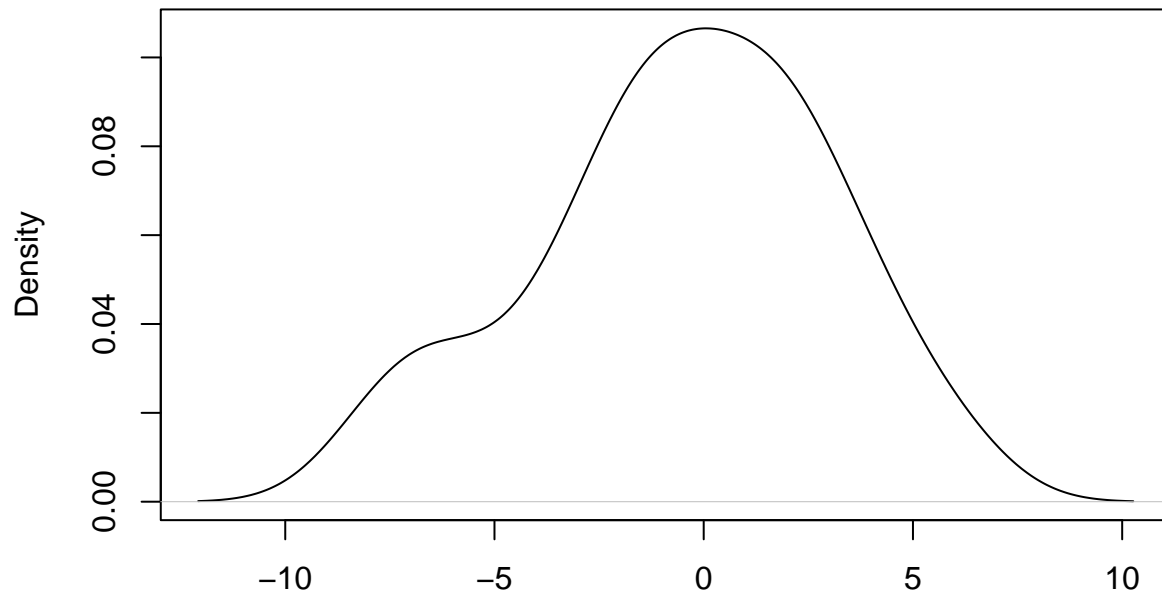
```
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 18 19 32 36 37 38 39 40 71 83
## [1] "Assuming symmetric error"
## [1] 11 12 13 15 36 37 38 39 40
## [1] "Simulation 16"
```

density(x = y_0)



N = 50 Bandwidth = 9.365

density(x = y - as.matrix(x) %*% beta_0)

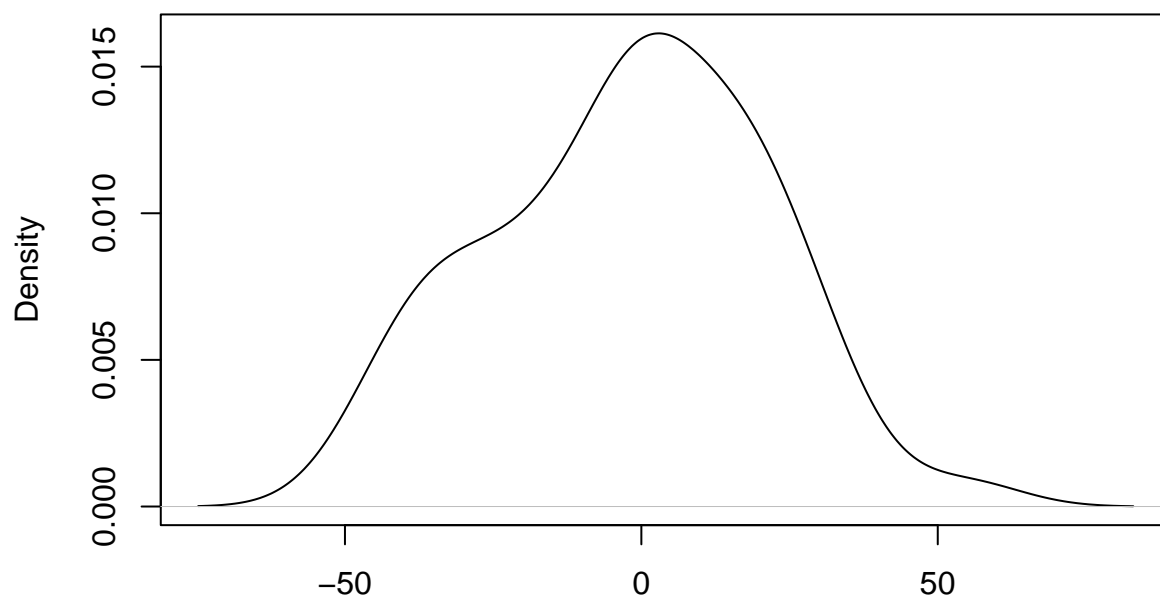


N = 50 Bandwidth = 1.368

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
```

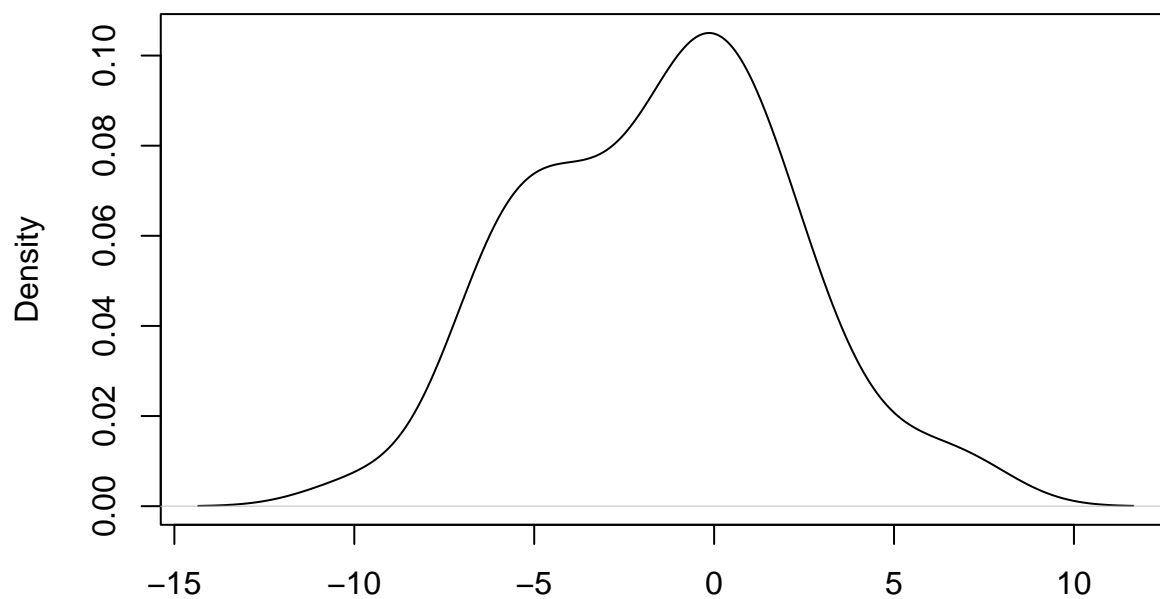
```
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 1 11 12 13 14 15 36 37 38 39 40 44 51 73
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 17"
```

density(x = y_0)



N = 50 Bandwidth = 9.467

density(x = y - as.matrix(x) %*% beta_0)



N = 50 Bandwidth = 1.491

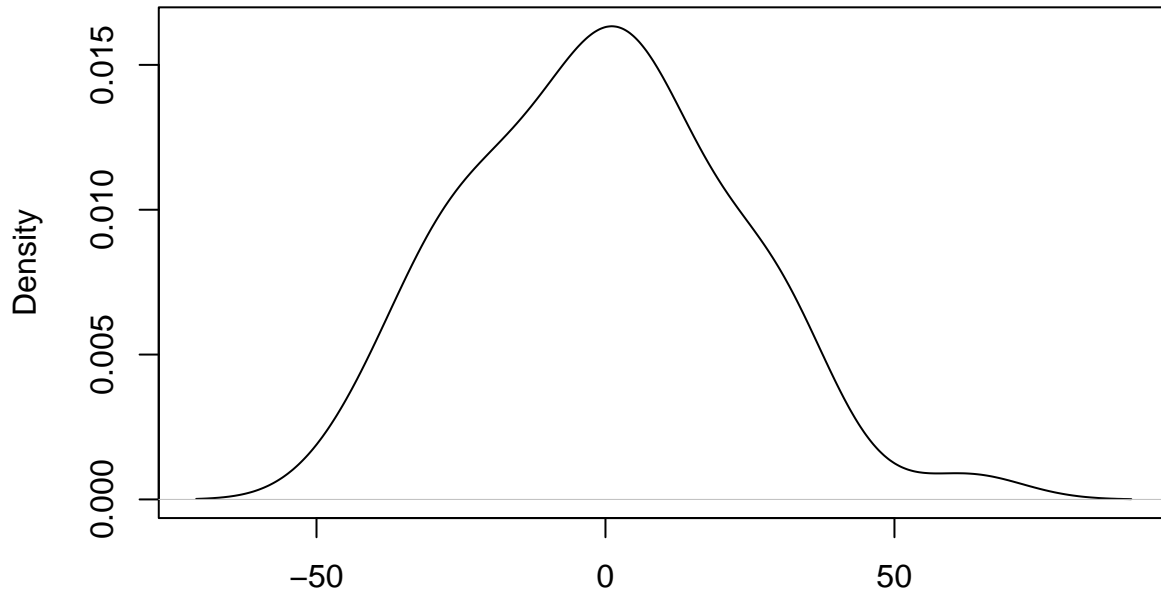
```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
```

```
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
```



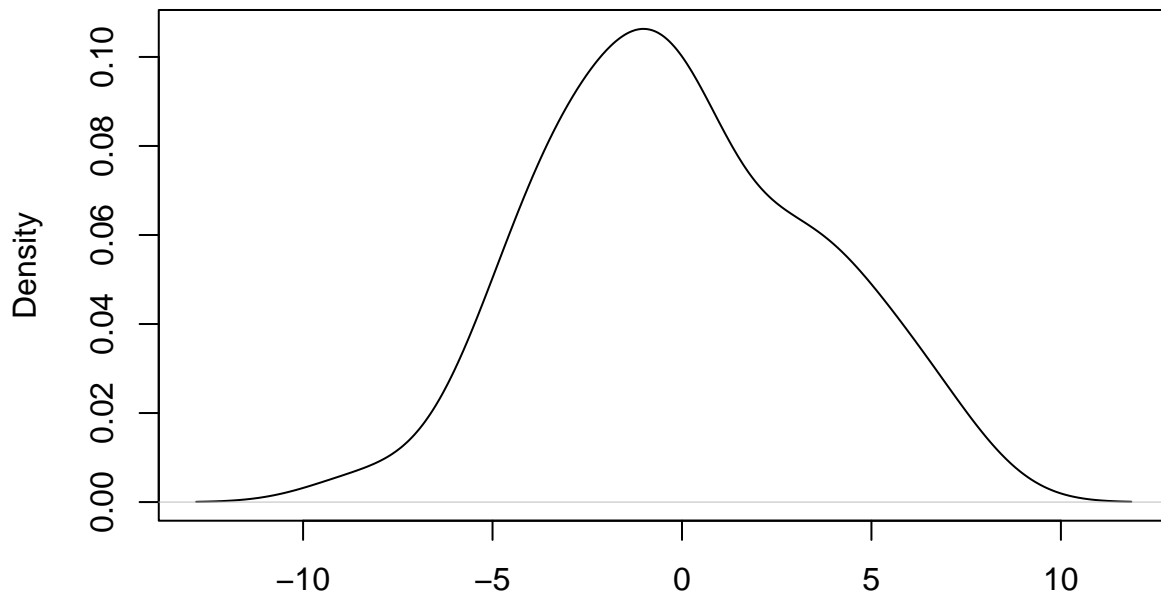
```
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 23 36 37 38 39 40 42 48 86 95 99
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 18"
```

density(x = y_0)



N = 50 Bandwidth = 9.35

density(x = y - as.matrix(x) %*% beta_0)

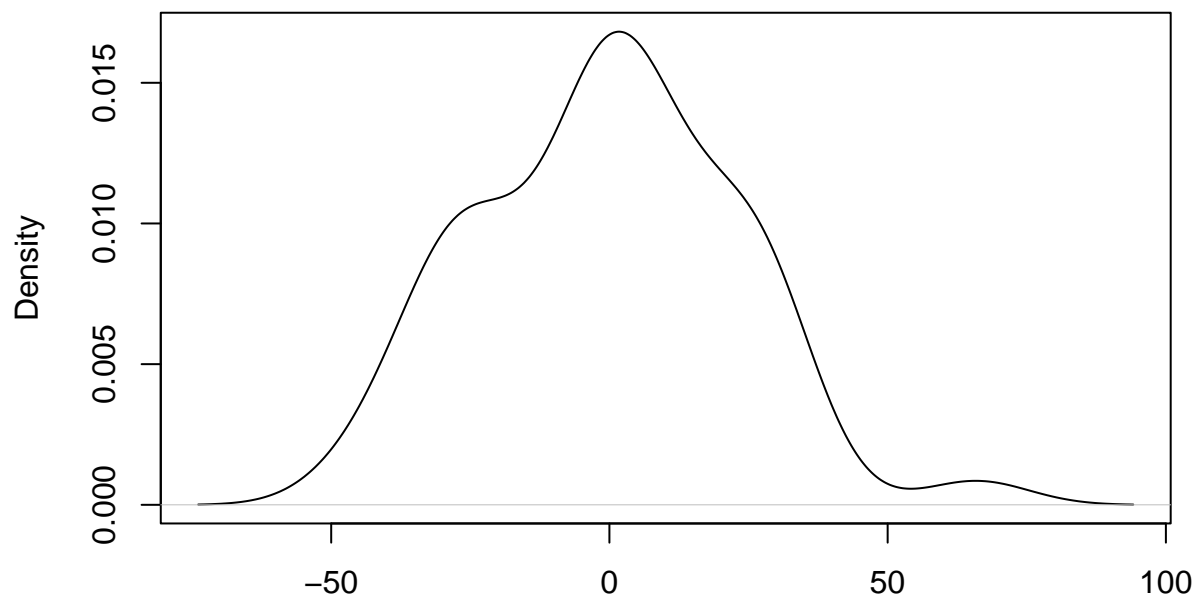


N = 50 Bandwidth = 1.467

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
```

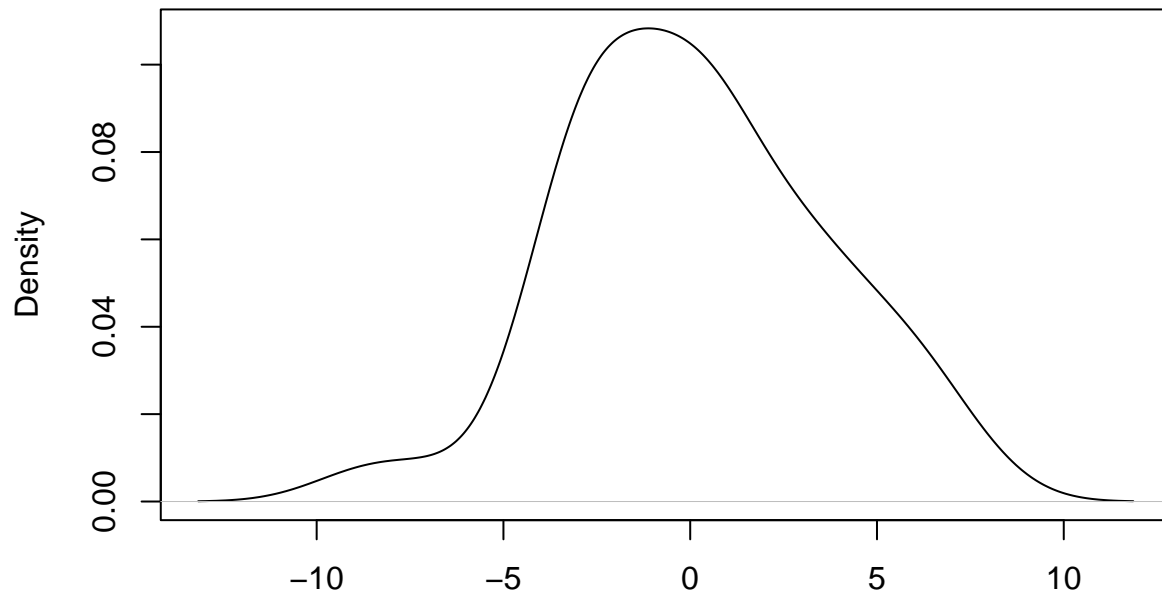
```
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 3 5 7 9 11 12 14 15 20 23 36 37 38 39 40 49 69 79 85 86
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 19"
```

density(x = y_0)



N = 50 Bandwidth = 9.395

density(x = y - as.matrix(x) %*% beta_0)

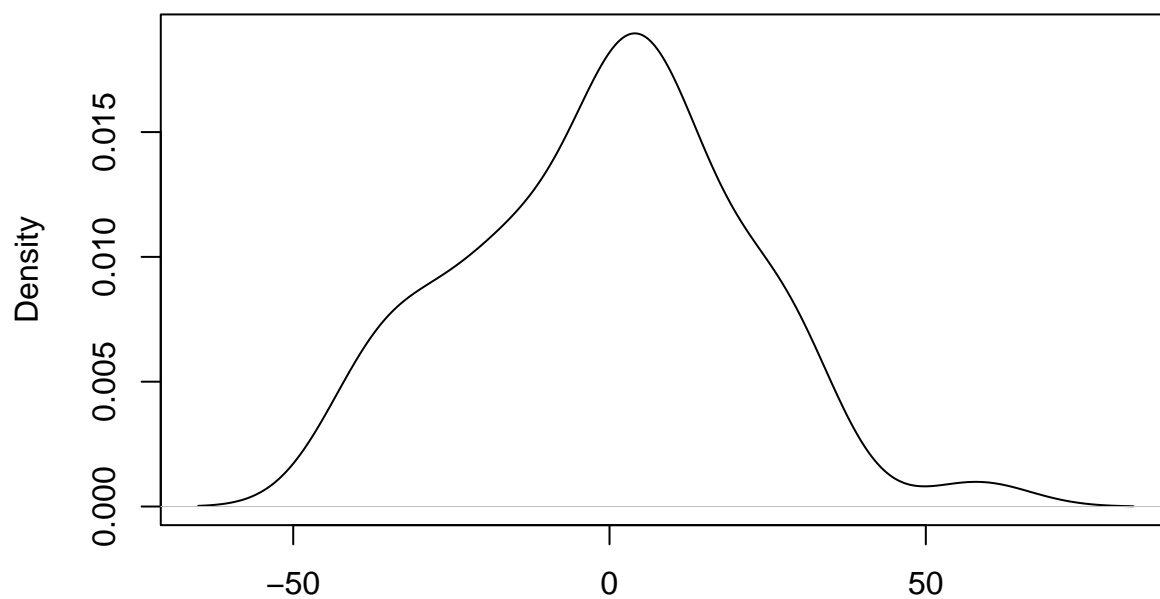


N = 50 Bandwidth = 1.433

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
```

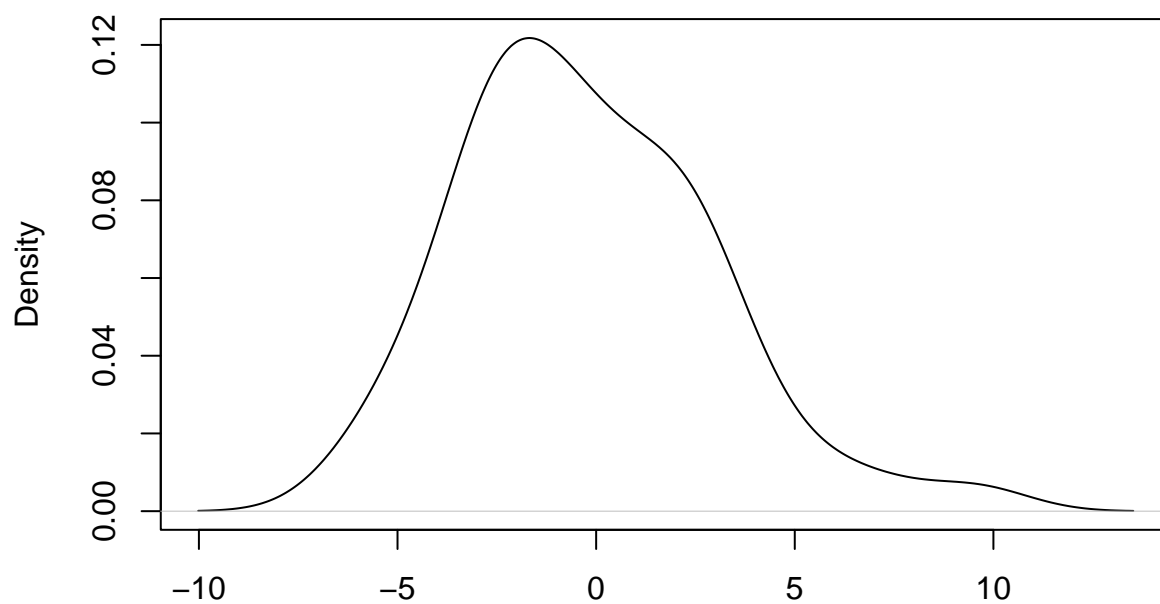
```
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 20 21 28 36 37 38 39 40 50 54 59 70 74
## [1] "Assuming symmetric error"
## [1] 11 12 13 14 15 36 37 38 39 40
## [1] "Simulation 20"
```

density(x = y_0)



N = 50 Bandwidth = 8.175

density(x = y - as.matrix(x) %*% beta_0)



N = 50 Bandwidth = 1.311

```
## iteration: 100
## iteration: 200
## iteration: 300
## iteration: 400
## iteration: 500
```

```
## iteration: 600
## iteration: 700
## iteration: 800
## iteration: 900
## iteration: 1000
## iteration: 1100
## iteration: 1200
## iteration: 1300
## iteration: 1400
## iteration: 1500
## iteration: 1600
## iteration: 1700
## iteration: 1800
## iteration: 1900
## iteration: 2000
## iteration: 2100
## iteration: 2200
## iteration: 2300
## iteration: 2400
## iteration: 2500
## iteration: 2600
## iteration: 2700
## iteration: 2800
## iteration: 2900
## iteration: 3000
## iteration: 3100
## iteration: 3200
## iteration: 3300
## iteration: 3400
## iteration: 3500
## iteration: 3600
## iteration: 3700
## iteration: 3800
## iteration: 3900
## iteration: 4000
## iteration: 4100
## iteration: 4200
## iteration: 4300
## iteration: 4400
## iteration: 4500
## iteration: 4600
## iteration: 4700
## iteration: 4800
## iteration: 4900
## iteration: 5000
## [1] 1000
## [1] 2000
## [1] 3000
## [1] 4000
## [1] 5000
## [1] 6000
## [1] 7000
## [1] "Real beta's"
## [1] 11 12 13 14 15 36 37 38 39 40
```

```
## [1] "Assuming asymmetric error"
## [1] 11 12 13 14 15 25 34 36 37 38 39 40 96 97
## [1] "Assuming symmetric error"
## [1] 11 13 14 15 36 38 39 40
```

```
VS_SN <- colSums(abs(rsim_SN)>1)
VS_SN
```

```
## [1] 4 1 1 1 2 0 3 0 2 0 20 20 18 20 20 2 1 1 3 2 2 1 3 0 1
## [26] 0 1 1 0 2 0 1 3 1 0 20 20 20 20 20 0 2 0 3 2 2 2 2 2 2
## [51] 2 0 0 1 1 2 1 2 1 1 1 1 0 0 1 2 3 1 3 1 2 2 2 1 0
## [76] 0 0 0 3 1 0 1 2 1 5 2 1 0 1 1 2 0 0 1 1 4 2 1 1 1
```

```
VS_N <- colSums(abs(rsim_N)>0)
VS_N
```

```
## [1] 0 0 0 0 0 0 0 0 0 0 0 20 19 19 19 18 0 0 0 0 0 0 0 0 0
## [26] 0 0 0 0 0 0 0 0 0 0 0 20 20 20 19 20 0 0 0 0 0 0 0 0 0
## [51] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
## [76] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```