Simulation_week5

Tingyu Zhu

10/28/2020

```
library(tidyverse)
library(purrr)
library(ggplot2)
library(here)
devtools::load_all()
set.seed(1222)
```

check for correcteness

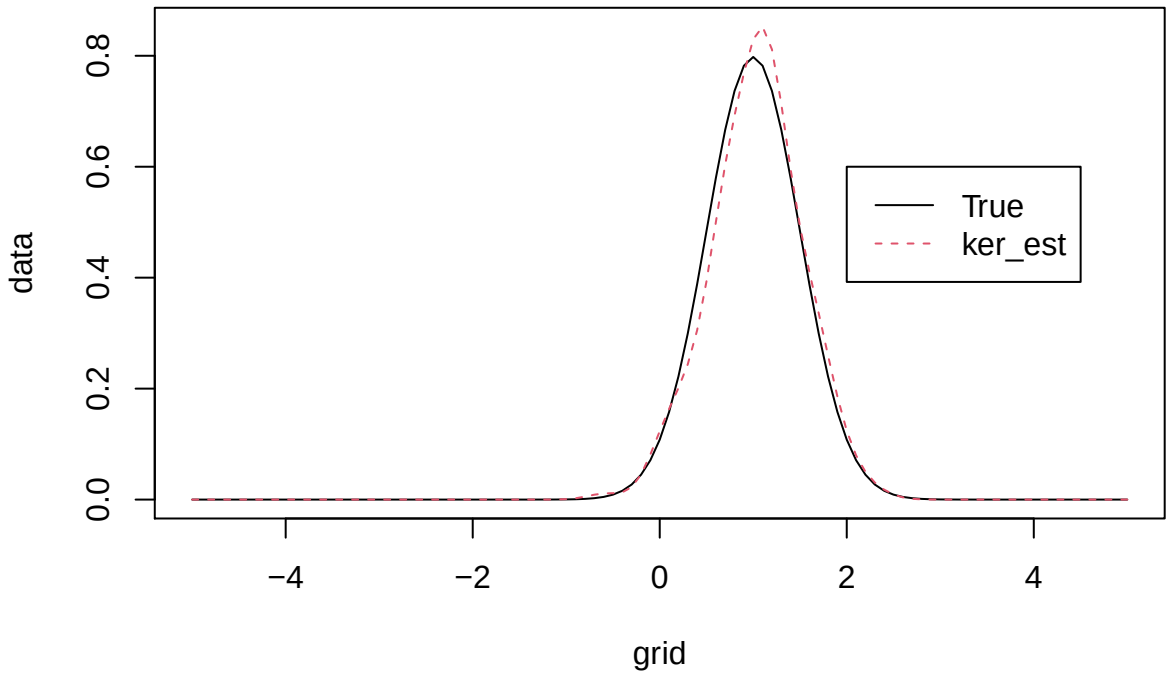
Here we compare the estimated density using the `KDE_est()` with the true density. Samples are drawn from the $N(1, 0.5^2)$. The ture densities are calculated using `dnorm()`.

```
# genearate samples
n <- 200
s <- 0.5
x <- rnorm(n, 1,s)
grid <- seq(-5,5,by=0.1)

# normal reference bandwidth selector
h <- 1.06*s*n^{-0.2}
# estimate using the KDE_est()
f1 <- KDE_est(x,grid,h,"normal")$f_est
```

plot

```
data <- cbind(dnorm(grid, 1,s), f1)
matplot(grid, data, lty=c(1:2),col=c(1:2), type="l")
legend(2,0.6,c("True","ker_est"),lty=c(1:2),col=c(1:2))
```



From the plots we can see that the estimated curve using our `KDE_est()` function is quite close to the ture density curve. This shows that our estimation function can generate correct estimated densities.

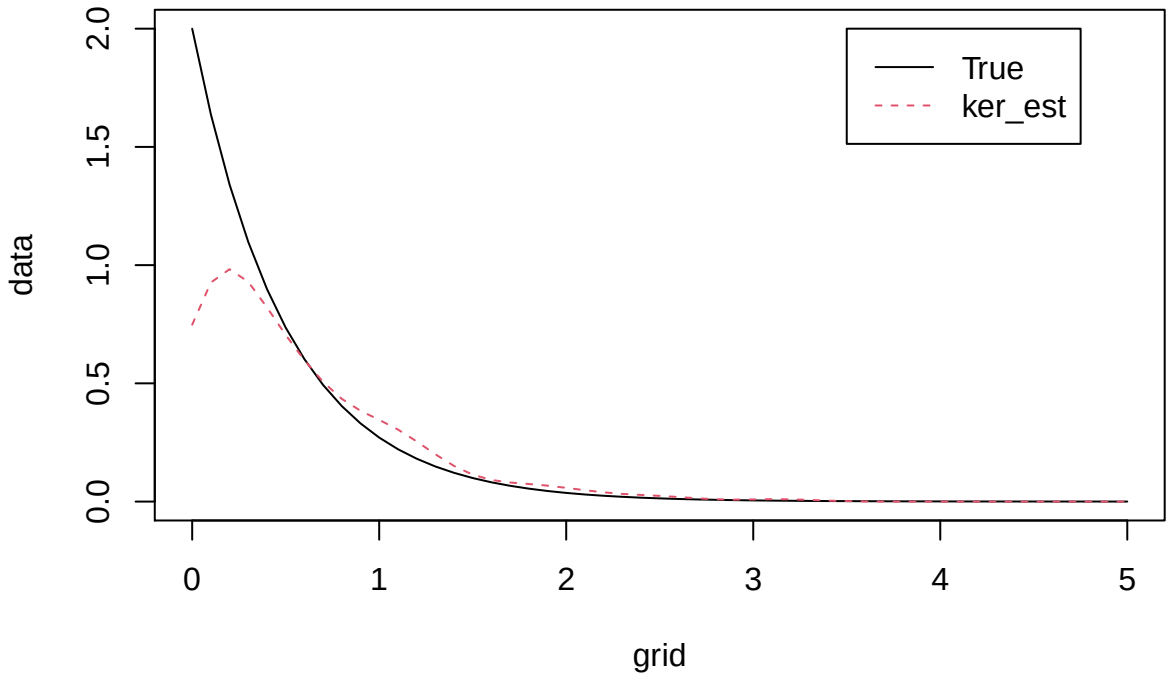
Boundary effect

Here we try to estimate the sample from exp(2) distribution to illustrate one of the biggest challenges of the KDE method- the boundary effect. The ture densities are calculated using `dexp()`.

```
# genearate samples
n <- 200
x <- rexp(n, 2)
grid <- seq(0,5,by=0.1)
# random bandwidth
h <- 0.2
f2 <- KDE_est(x,grid,h,"normal")$f_est
```

plot

```
data <- cbind(dexp(grid, 2), f2)
matplot(grid, data, lty=c(1:2),col=c(1:2), type="l")
legend(3.5,2,c("True","ker_est"),lty=c(1:2),col=c(1:2))
```



As stated in the literature, under some situations, the KDE method can not handle the estimation of the boundary very well. There're methods that can fix the estimation on the boundaries. We will not include this part in this project.