HW7_Lin

zhengzhi lin

2019.10.25

P1

a

The problem is obivious, he uses logapple08, logrm08 instead of the data he generate in bootstrap.

```
library(quantreg)
```

```
## Loading required package: SparseM
```

```
##
## Attaching package: 'SparseM'
```

```
## The following object is masked from 'package:base':
##
##     backsolve
```

```
library(quantmod)
```

```
## Loading required package: xts
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo
```

```
## Loading required package: TTR
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Version 0.4-0 included new data defaults. See ?getSymbols.
```

```r
data(barro)
#1)fetch data from Yahoo
#AAPL prices
apple08 <- getSymbols('AAPL', auto.assign = FALSE, from = '2008-1-1', to =
                        "2008-12-31")[,6]
```

```
## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.
```

```r
#market proxy
rm08<-getSymbols('^ixic', auto.assign = FALSE, from = '2008-1-1', to =
                   "2008-12-31")[,6]

#log returns of AAPL and market
logapple08<- na.omit(ROC(apple08)*100)
logrm08<-na.omit(ROC(rm08)*100)

#OLS for beta estimation
beta_AAPL_08<-summary(lm(logapple08~logrm08))$coefficients[2,1]

#create df from AAPL returns and market returns
df08<-cbind(logapple08,logrm08)
set.seed(666)
Boot_times=1000
sd.boot=rep(0,Boot_times)
beta_1 <- rep(0,Boot_times)
for(i in 1:Boot_times){
  # nonparametric bootstrap
  bootdata=df08[sample(nrow(df08), size = 1000, replace = TRUE),]
  sd.boot[i]= coef(summary(lm(AAPL.Adjusted~IXIC.Adjusted, data = bootdata)))[2,2]
}
head(sd.boot)
```

```
## [1] 0.03198332 0.02977593 0.02750201 0.02845086 0.02917462 0.03111990
```

b

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:xts':
##
##     first, last
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
coe <- matrix(0,nrow = 1000,ncol = 5)
url <- "https://www2.isye.gatech.edu/~jeffwu/wuhamadabook/data/Sensory.dat"
operator <- read.table(url, fill = TRUE)
op_dat <- as.matrix(operator[-c(1:2), ])
for (i in 1:10) {
  t <- op_dat[3 * i - 1, 1:5]
  t <- c(i, t)
  op_dat[3 * i - 1, ] <- t
  m <- op_dat[3 * i, 1 : 5]
  m <- c(i, m)
  op_dat[3 * i, ] <- m
}
op_dat <- op_dat %>% as.data.frame() %>% rename(item = V1,
         operator1 = V2, operator2 = V3, operator3 = V4, operator4 = V5,
         operator5 = V6) %>%
  mutate_if(is.factor, as.character) %>% mutate_if(is.character, as.numeric )
system.time(for(i in 1:Boot_times){
  # nonparametric bootstrap
  bootdata=op_dat[sample(nrow(op_dat), size = 100, replace = TRUE),2:6]
  coe[i,]= coef(summary(lm(operator1 ~ operator2 + operator3 + operator4 + operator5, data = bootdata))
})
```

```
##    user  system elapsed
##    1.05    0.01    1.06
```

```
coe <- coe %>% as.data.frame() %>% rename(operator2 = V2,
                                          operator3 = V3,
                                          operator4 = V4,
                                          operator5 = V5,
                                          intercep = V1)
```

c Speed of r codes are bounded by cpu, therefore, enable multiple tasks to take place and make use of more than one processor will make computation faster.

```
library(parallel)
library(foreach)
library(doParallel)
```

```
## Loading required package: iterators
```

```r
cores=detectCores()
cl <- makeCluster(cores[1]-1)
registerDoParallel(cl)

coe <- matrix(0,nrow = 1000,ncol = 5)


system.time({foreach(i = 1:Boot_times) %dopar% {
  # nonparametric bootstrap
  bootdata=op_dat[sample(nrow(op_dat), size = 100, replace = TRUE),2:6]
  coe[i,]= coef(summary(lm(operator1 ~ operator2 + operator3 + operator4 + operator5, data = bootdata))
}
})
```

```
##    user  system elapsed 
##    0.43    0.07    0.54 
```

```r
coe <- coe %>% as.data.frame() %>% rename(operator2 = V2,
                                          operator3 = V3,
                                          operator4 = V4,
                                          operator5 = V5,
                                          intercep = V1)
stopCluster(cl)
```

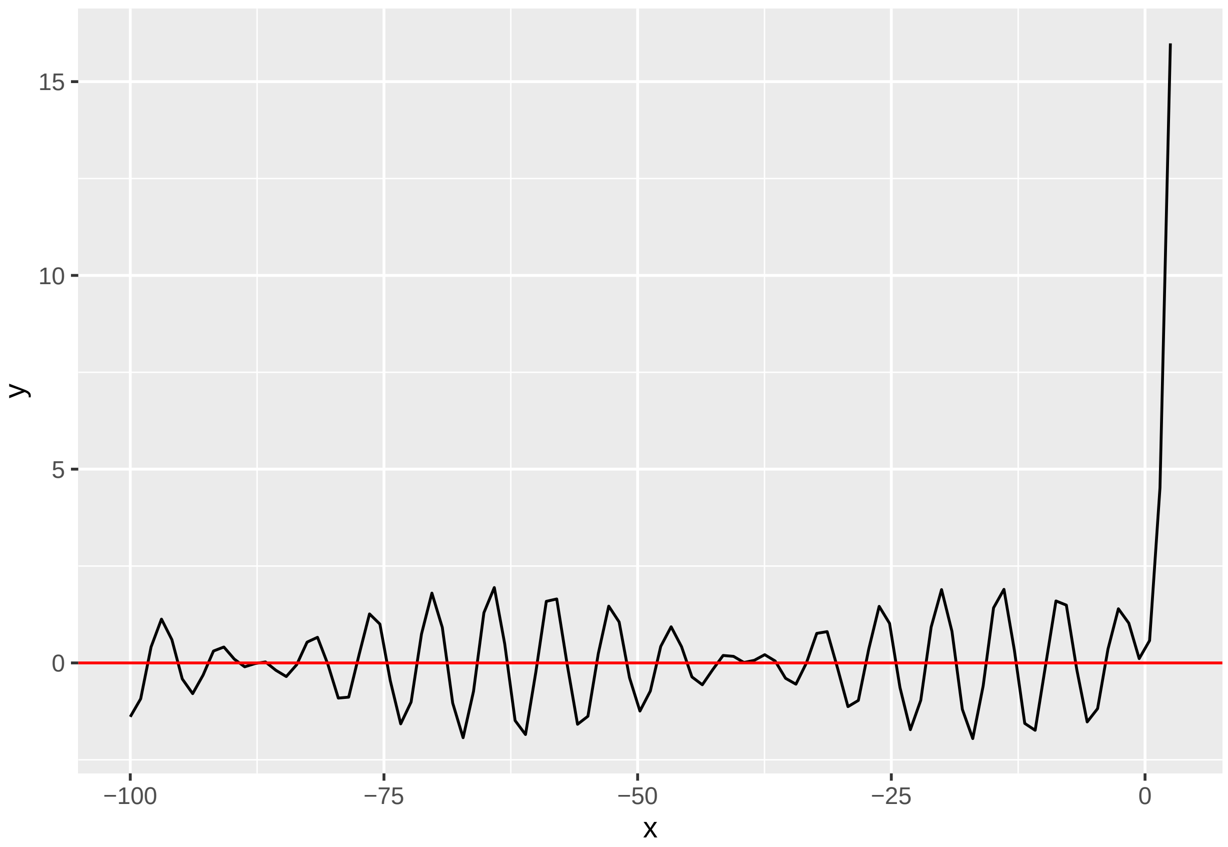
p3

a

```r
library(ggplot2)
func <- function(x){
  y <- 3 ^ x - sin(x) + cos(5 * x)
  return(y)
}

d_func <- function(x){
  y <- 3 ^ x * log(3) - cos(x) - 5 * sin(5 * x)
  return(y)
}

ggplot(data = data.frame(x = 0,y = 0), mapping = aes(x = x)) +
  stat_function(fun = func) +
  xlim(-100, 2.5) +
  geom_abline(intercept = 0, slope = 0, colour = "red")
```



```
x_0 <- -2.5
eps <- 1e-6
x <- x_0
nt <- function(x) {while (abs(func(x)-0) > eps) {
  x <- x - func(x)/d_func(x)
  return(x)
}}
nt(x_0)
```

```
## [1] -5.574795
```

```
system.time(solution <- lapply(-1000:0, nt))
```

```
##    user  system elapsed 
##    0.02    0.00    0.02
```

b

```
system.time(solution <- mclapply(-1000:0, nt, mc.cores = 1))
```

```
##    user  system elapsed 
##       0       0       0
```

```r
mean(solution)
```

```
## Warning in mean.default(solution): argument is not numeric or logical:
## returning NA
```

```
## [1] NA
```