Head and Shoulders Pattern Detection

Chenxi Wang | Rongbing Liang | Yena Lee

April 19, 2020

```r
# DJI <- read.csv("DJI.csv")
HO_raw <- read.table("HO-5min.asc", sep = ",", header = TRUE)

HO.df = HO_raw[,1:6]
HO.df$Date = as.Date(HO.df$Date,"%m/%d/%y")
temp = function(x) {x=x*100}
HO.df[,3:6] = apply(HO.df[3:6],2,temp)
HO1 = subset(HO.df,Date>="2010-01-01"&Date<="2014-01-01")
```

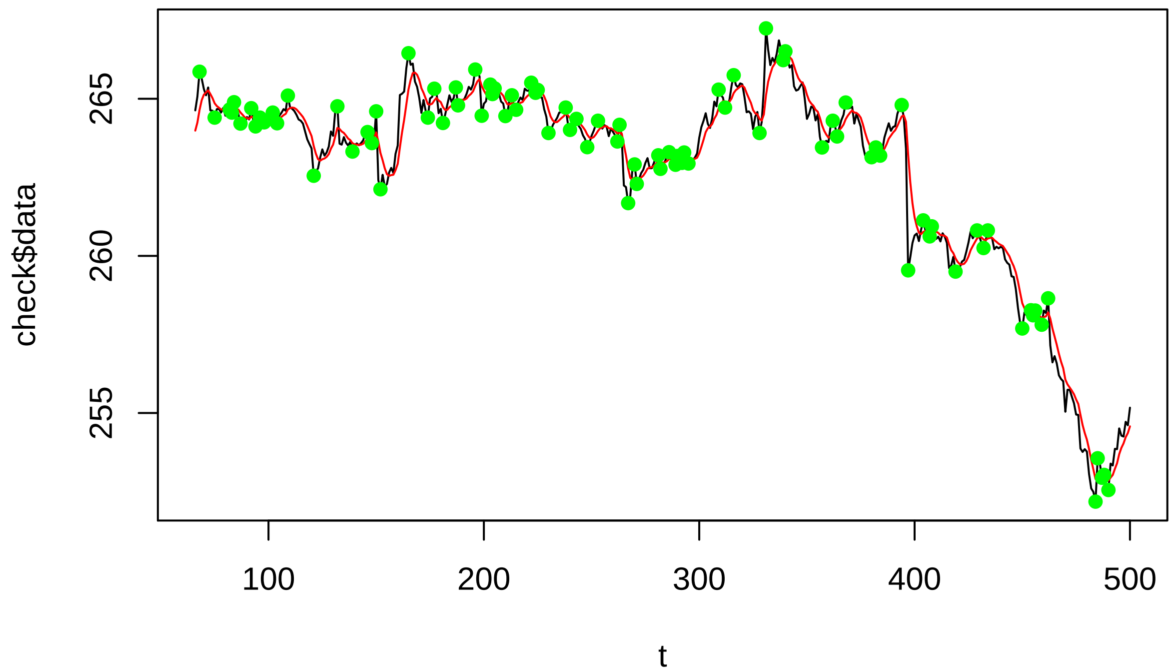
```r
library(sm)
library(quantmod)
```

STEP 1: Smoothing & Find Extrema on Original Data

```r
find.extrema <- function(dat, windowlen) {

  n <- length(dat)
  t <- 1:n

  # fit kernel regression with cross-validation
  h <- h.select(t, dat, method = "cv")
  ks_p <- c()

  for (i in 1:(n-windowlen+1)) {
    ks <- ksmooth(t[i:(i+windowlen-1)], dat[i:(i+windowlen-1)],
                  kernel = c("normal"), bandwidth = h,
                  n.points = windowlen, x.points = i+windowlen-1)
    ks_p <- c(ks_p, ks$y)
  }

  # find estimated fit
  dat_sm <- ks_p

  second_deriv <- diff(sign(diff(dat_sm)))
  temp_loc <- which(second_deriv != 0 ) + 1 # index of extrema in smoothed data
  loc_dir <- -sign(second_deriv[temp_loc-1]) # direction of extrema,
                                             # +1 for max, -1 for min
```

```r
  dat <- dat[windowlen:n] # make original data the same length as smoothed data

  # find index of extrema in original data
  loc <- rep(0, length(temp_loc))
  for (e in 1:length(temp_loc)) {

    if (e == 1) {
      if (loc_dir[e] == 1) {
        # find max from start to E0
        loc[e] <- which.max(dat[1:temp_loc[e]])
      } else {
        # find min from start to E0
        loc[e] <- which.min(dat[1:temp_loc[e]])
      }

    } else {
      if (loc_dir[e] == 1) {
        # find max from E[e-1] to E[e]
        loc[e] <- temp_loc[e-1] + which.max(dat[temp_loc[e-1]:temp_loc[e]]) - 1
      } else {
        # find min from E[e-1] to E[e]
        loc[e] <- temp_loc[e-1] + which.min(dat[temp_loc[e-1]:temp_loc[e]]) - 1
      }
    }
  }

  return(list(data = dat,
              data_sm = dat_sm,
              extrema_loc = loc,
              extrema_dir = loc_dir,
              extrema_sm = temp_loc,
              bandwidth = h))
}
```

```r
windowlen <- 66
dat <- HO1$Close[1:500]
check <- find.extrema(dat, windowlen)

n <- length(dat)
t <- windowlen:n
plot(t, check$data, type = "l", col = "black")
lines(t, check$data_sm, col = "red")
points(t[check$extrema_loc], check$data[check$extrema_loc],
       col = "green", pch = 16)
```




```
check$bandwidth
```

```
## [1] 9.030733
```

STEP 2: Define Patterns Trying to Find

```
# Define HS pattern
HS <- list()
HS$len <- 5
HS$start <- 1 # start with a maximum
HS$formula <- expression({
  avg.top <- (E1 + E5) / 2
  avg.bot <- (E2 + E4) / 2

  # E3 > E1, E3 > E5
  E3 > E1 &
  E3 > E5 &

  # E1 and E5 are within 1.5% of their average
  abs(E1 - avg.top) < 0.015 * avg.top &
  abs(E5 - avg.top) < 0.015 * avg.top &

  # E2 and E4 are within 1.5% of their average
```

```
  abs(E2 - avg.bot) < 0.015 * avg.bot &
  abs(E4 - avg.bot) < 0.015 * avg.bot
})
```

```
# Define half-HS pattern, with only E1, E2, E3
HHS <- list()
HHS$len <- 3
HHS$start <- 1 # start with a maximum
HHS$formula <- expression({
  # E3 > E1, E1 > E2
  E3 > E1 &
  E1 > E2 # This is actually unnecassary since follwing max E1, E2 must be a min
})
```

STEP 3: Find Patterns in Original Data

```
find.pattern <- function(extrema, pattern) {
  # ============
  # extrema: list object, output from find.extrema()
  # pattern: list object, defined for each technical pattern
  # ============

  data_orig <- extrema$data # original data
  data_sm <- extrema$data_sm # smoothed data
  extrema_loc <- extrema$extrema_loc
  extrema_dir <- extrema$extrema_dir
  n <- length(extrema_loc)

  # search for patterns
  pattern_starts <- c()
  for (i in 1:n) {

    # check E1
    if (pattern$start == extrema_dir[i]) {

      # check that there is suffcient number of extrema to complete pattern
      if ((i + pattern$len - 1) <= n) {

        # create enviroment to check pattern
        # Slice the 5 points in ORIGINAL data for evaluation
        envir_data = c(data_orig[extrema_loc][i:(i + pattern$len - 1)],
                       extrema_loc[i:(i + pattern$len - 1)])

        names(envir_data) = c(paste('E', 1:pattern$len, sep=''),
                              paste('t', 1:pattern$len, sep=''))

        envir_data = as.list(envir_data)

        # check if pattern was found
        if (eval(pattern$formula, envir = envir_data)) {
          pattern_starts <- c(pattern_starts, i)
```

```
        }
      }
    }
  }

  return(pattern_starts)
}
```

Splitting Positive and Negative Cases

```
# Find complete HS, i.e. positive cases
check_pattern <- find.pattern(check, HS)
check_pattern # locations of E1
```

```
##  [1]  3 11 19 25 33 49 59 65 69 83
```

```
# Find half HS, i.e. both positive and negative cases
check_all <- find.pattern(check, HHS)
check_all
```

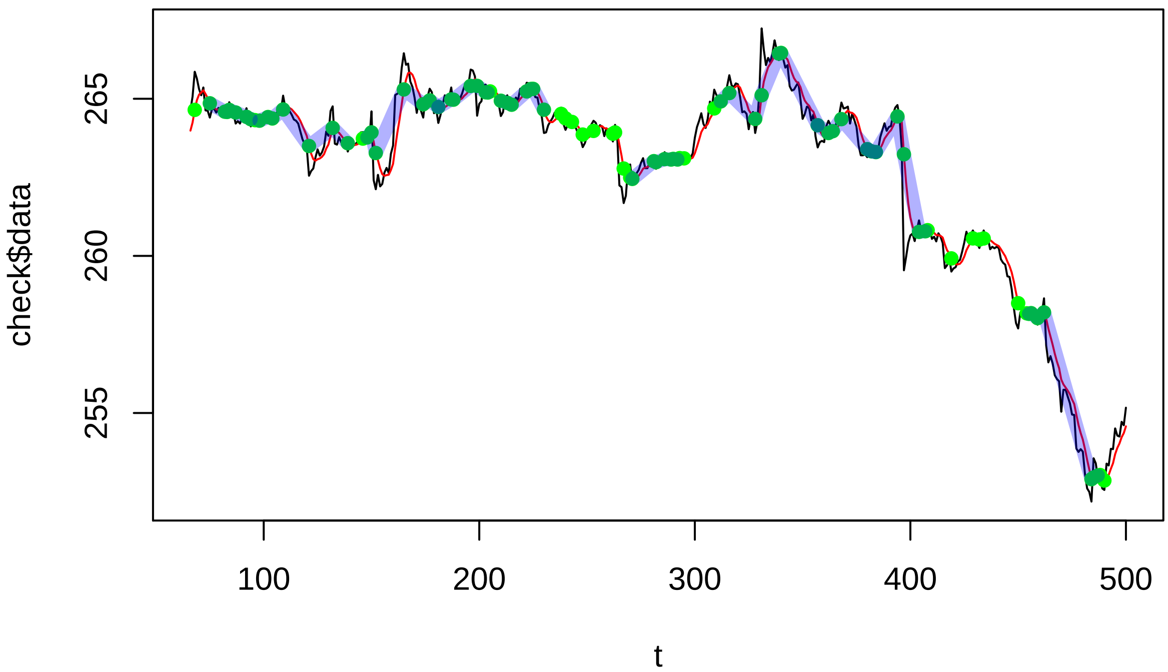
```
##  [1]  3  9 11 17 19 23 25 33 47 49 53 55 57 59 65 69 83
```

```
#intersect(check_pattern, check_all) == check_pattern

# Extract negative cases
check_neg <- setdiff(check_all, check_pattern)
check_neg
```

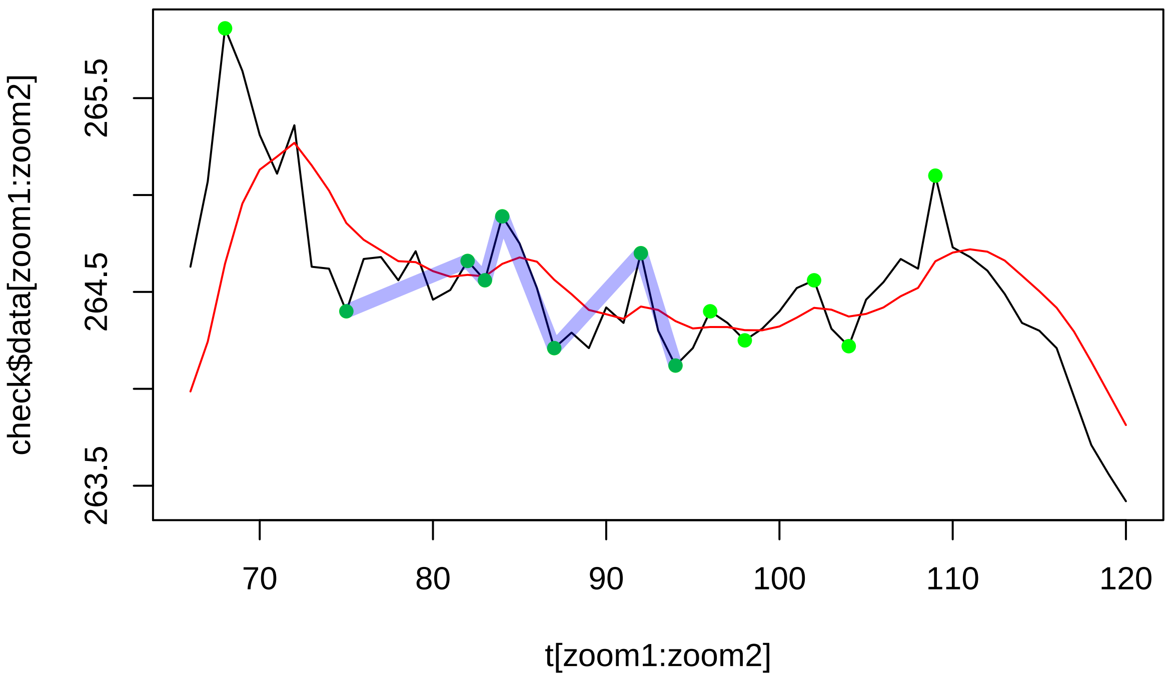
```
## [1]  9 17 23 47 53 55 57
```

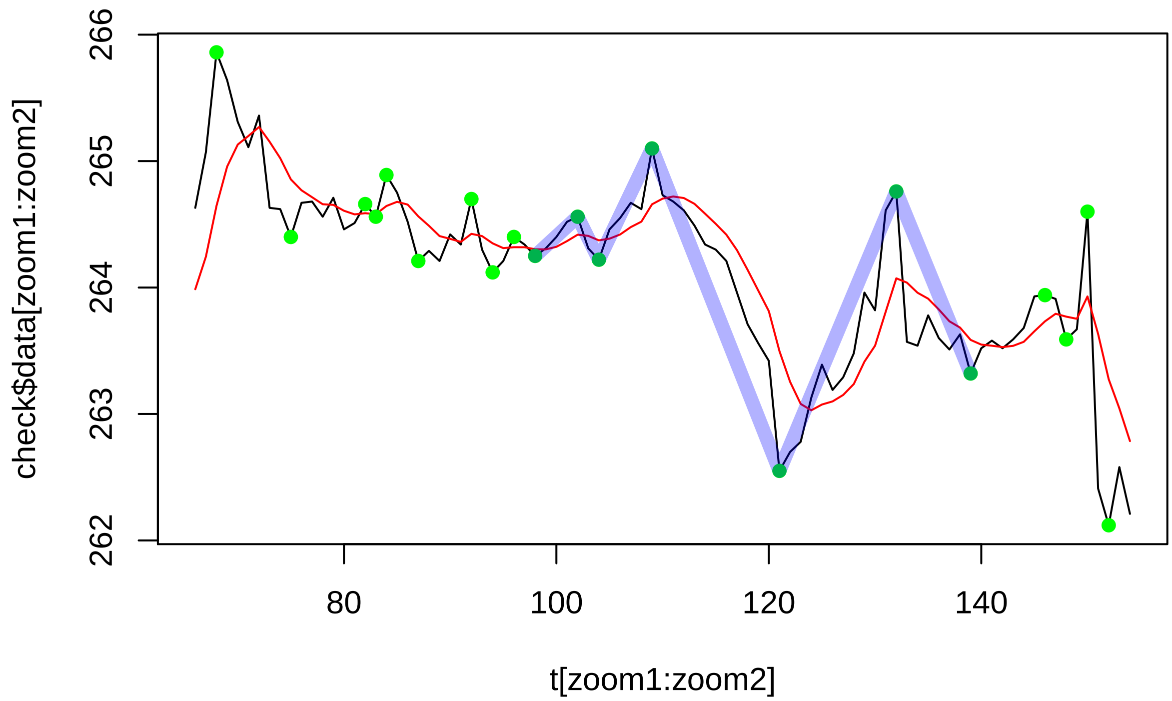
Plotting HS

```
# plot complete HS in one graph
if (!is.null(check_pattern)) {
  plot(t, check$data, type = "l", col = "black")
  lines(t, check$data_sm, col = "red")
  points(t[check$extrema_loc], check$data_sm[check$extrema_loc],
         col = "green", pch = 16)

  for (extrema_idx in check_pattern) {

    extrema_idxes <- (extrema_idx-1):(extrema_idx + HS$len)
    data_idxes <- check$extrema_loc[extrema_idxes]

    lines(t[data_idxes], check$data_sm[data_idxes],
          col = rgb(0, 0, 1, alpha = 0.3), lwd = 7)
  }
}
```

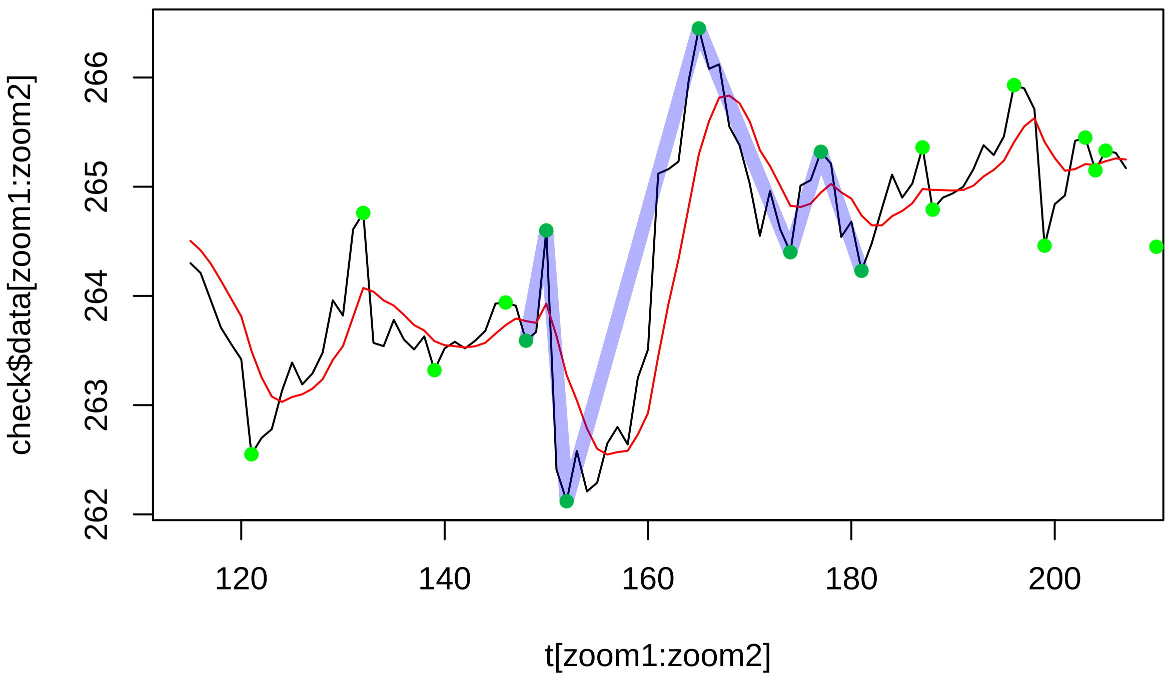



```r
# plot E0 and E6
for (extrema_idx in check_pattern){

  extrema_idxes <- (extrema_idx - 1):(extrema_idx + HS$len)
  data_idxes <- check$extrema_loc[extrema_idxes]
  zoom1 = max(data_idxes[1]-windowlen*0.5,0)
  zoom2 = data_idxes[5]+windowlen*0.5

  plot(t[zoom1:zoom2], check$data[zoom1:zoom2], type = "l", col = "black")
  lines(t[zoom1:zoom2], check$data_sm[zoom1:zoom2], col = "red")
  points(t[check$extrema_loc], check$data[check$extrema_loc],
         col = "green", pch = 16)
  lines(t[data_idxes], check$data[data_idxes],
        col = rgb(0, 0, 1, alpha = 0.3), lwd = 7)
}
```

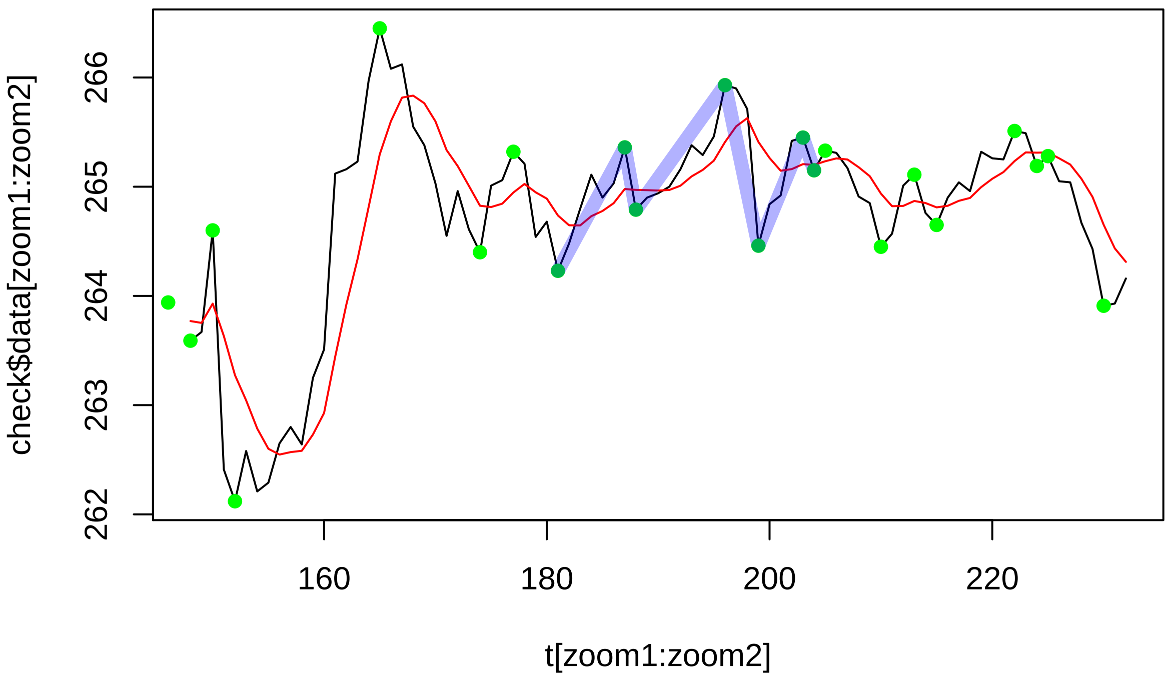

check$data[zoom1:zoom2]
263.5
264.5
265.5
70
80
90
100
110
120
t[zoom1:zoom2]



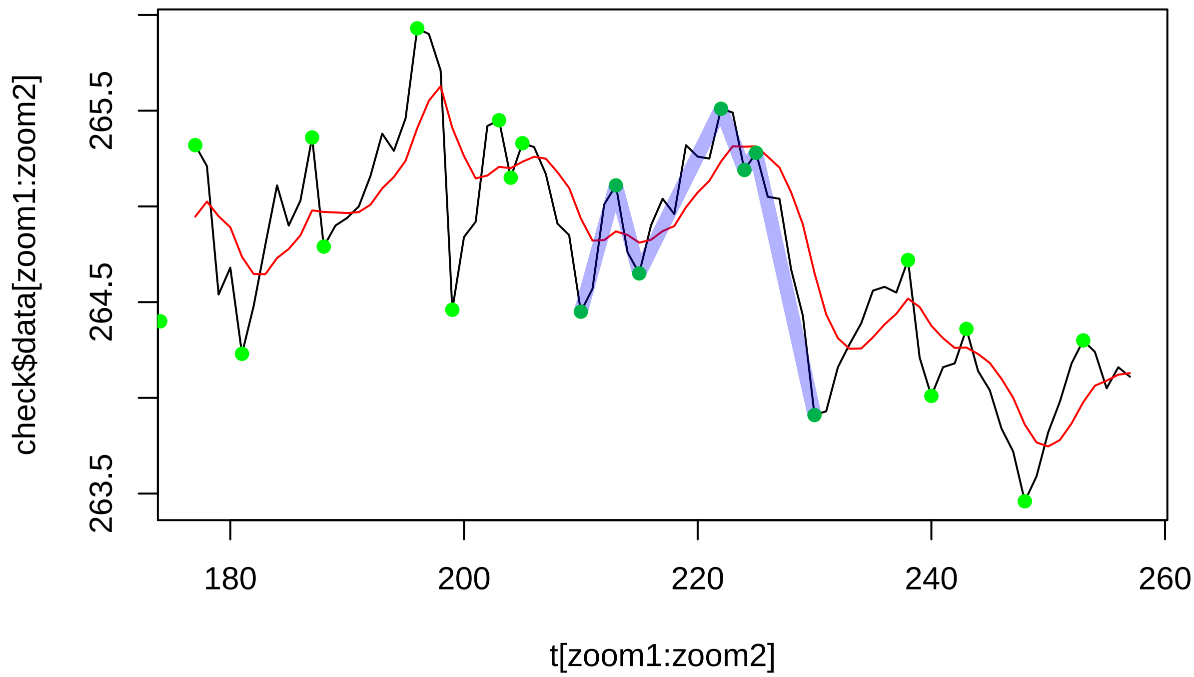
266
265
264
263
262
check$data[zoom1:zoom2]
80
100
120
140
t[zoom1:zoom2]



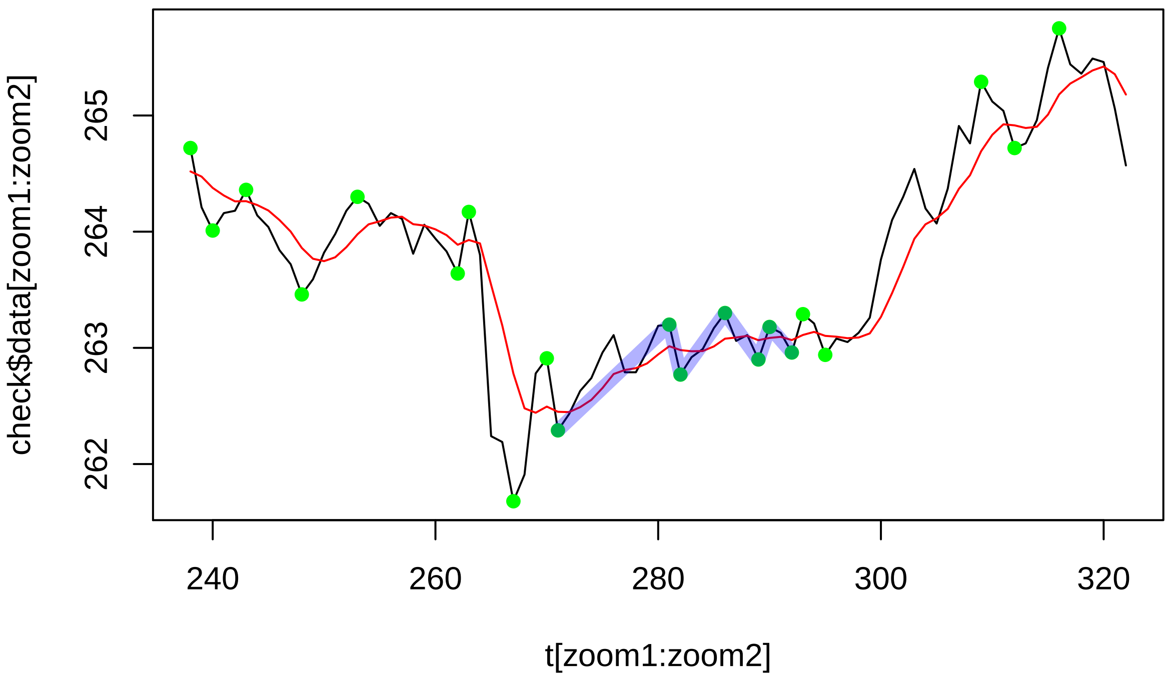
check$data[zoom1:zoom2]
262
263
264
265
266
120
140
160
180
200
t[zoom1:zoom2]



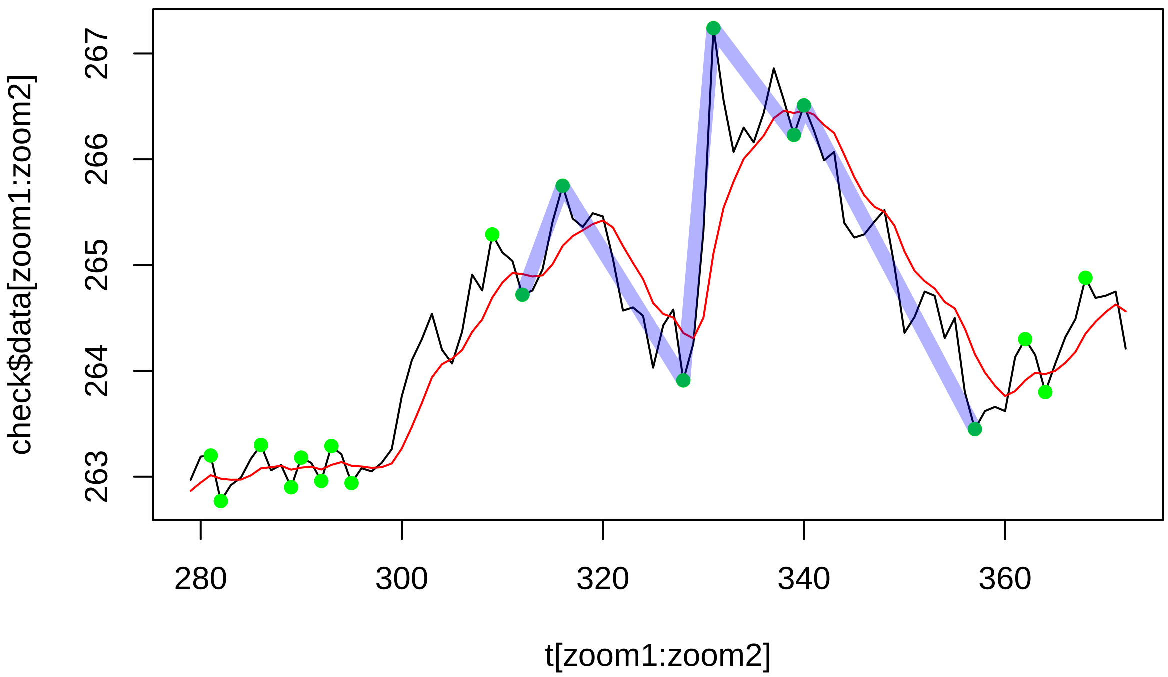
check$data[zoom1:zoom2]
262
263
264
265
266
160
180
200
220
t[zoom1:zoom2]



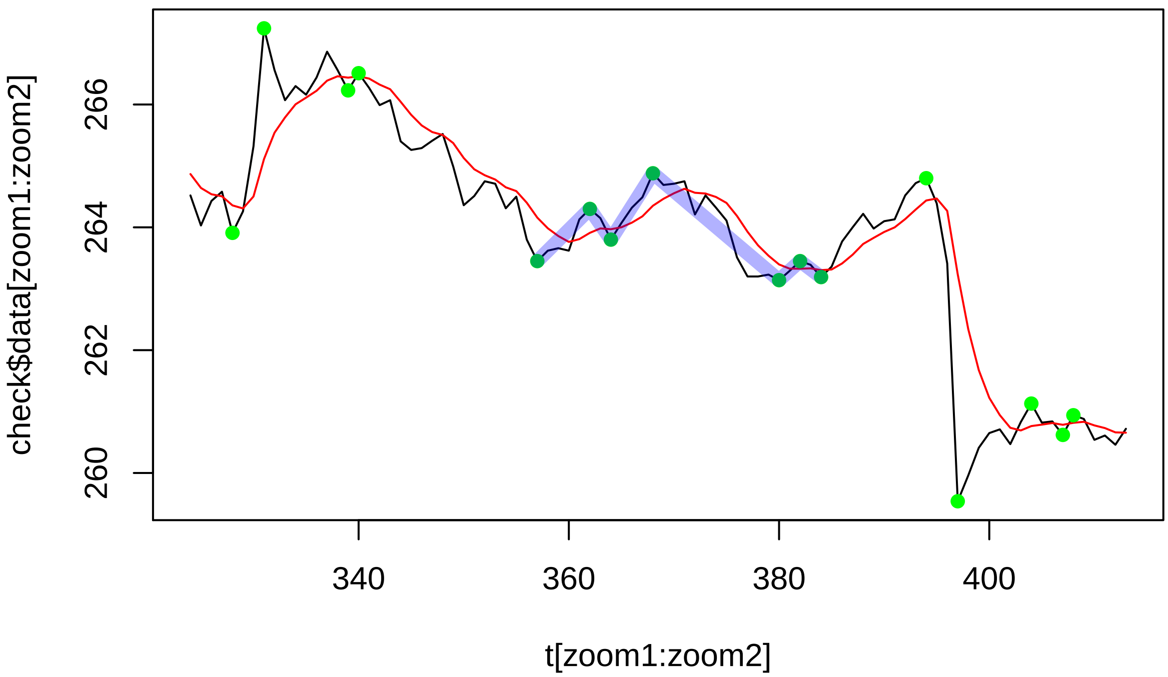
check$data[zoom1:zoom2]
263.5
264.5
265.5
180
200
220
240
260
t[zoom1:zoom2]

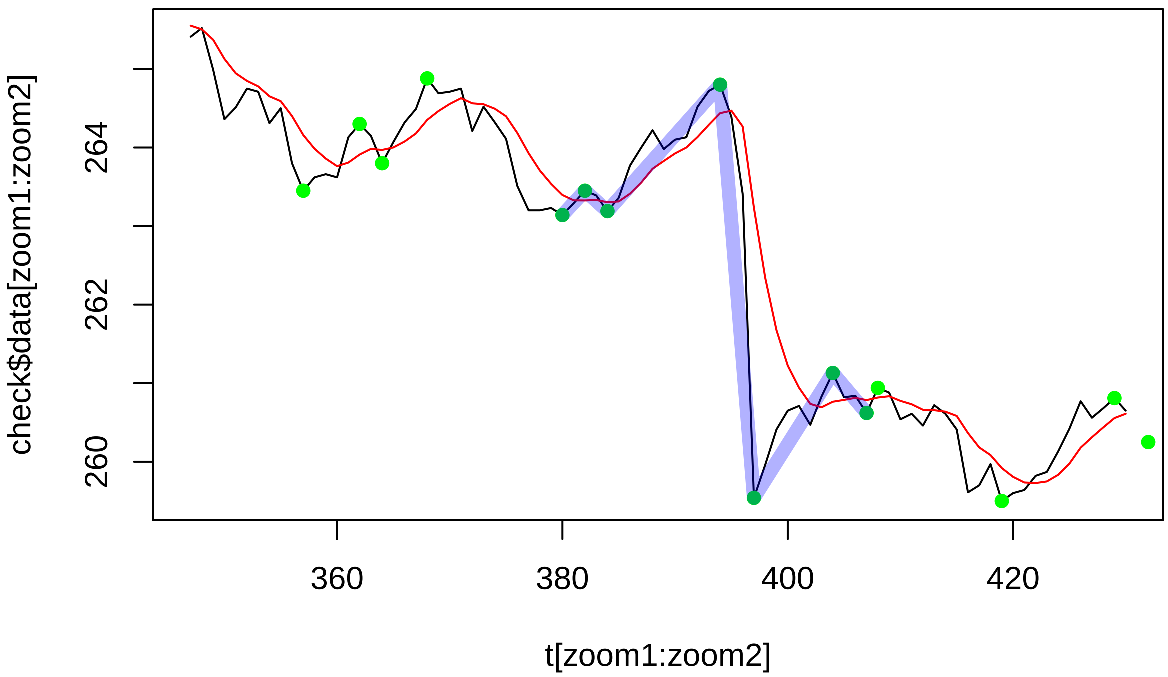


check$data[zoom1:zoom2]
262
263
264
265
240
260
280
300
320
t[zoom1:zoom2]

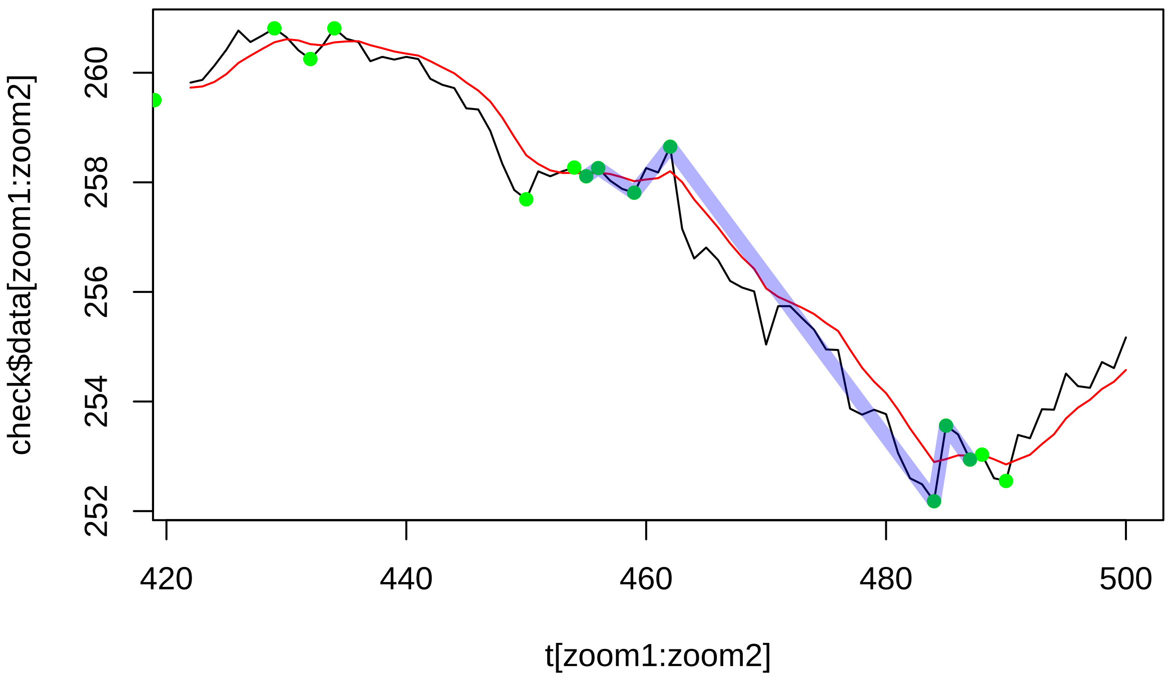


check$data[zoom1:zoom2]
263
264
265
266
267
280
300
320
340
360
t[zoom1:zoom2]


check$data[zoom1:zoom2]
t[zoom1:zoom2]
260
262
264
266
340
360
380
400



check$data[zoom1:zoom2]
260
262
264
360
380
400
420
t[zoom1:zoom2]



economic expansion dataset

```
HO1 = subset(HO.df,Date>="2010-01-01"&Date<="2018-01-01")

dat1=HO1$Close
check1 <- find.extrema(dat1, windowlen)

n <- length(dat1)
t <- windowlen:n
```

average time lag

```
temp1=check1$extrema_loc
temp2=check1$extrema_sm


mean(temp2-temp1)+1 #(5min) this is average time we need to find an extrema when it appears
```

```
## [1] 8.981593
```

```r
mean(diff(temp1)) # (5min) this is average duration between each extrema.
```

```
## [1] 21.44675
```

#attach indexes to dataset

```r
# dataset define
data=as.data.frame(dat1[t])
colnames(data)=c('Price')

data$Price_change=HO1$Close[t]-HO1$Open[t]
data$Return=data$Price_change/HO1$Open[t]

data$HS_Eloc=rep(NA,nrow(data))

data$HHS_Eloc=rep(NA,nrow(data))
```

```r
check_pattern <- find.pattern(check1, HS)
 # locations of E1

# Find half HS, i.e. both positive and negative cases
check_all <- find.pattern(check1, HHS)

#intersect(check_pattern, check_all) == check_pattern

# Extract negative cases
check_neg <- setdiff(check_all, check_pattern)
```

positive cases location

```r
temp=0:6

for (extrema_idx in check_pattern){

  extrema_idxes <- (extrema_idx - 1):(extrema_idx + HS$len)
  data_idxes <- check1$extrema_loc[extrema_idxes]

  temp=rbind(temp,data_idxes)


}

Pos_loc=as.data.frame(temp[-1,])

colnames(Pos_loc)=c('E0','E1','E2','E3','E4','E5','E6')
```

negative case location

```r
temp=0:4

for (extrema_idx in check_neg){

  extrema_idxes <- (extrema_idx - 1):(extrema_idx + HHS$len)
  data_idxes <- check1$extrema_loc[extrema_idxes]

  temp=rbind(temp,data_idxes)


}

Neg_loc=as.data.frame(temp[-1,])

colnames(Neg_loc)=c('tE0','tE1','tE2','tE3','tE4')
```

construct dataset of HS

```r
data$org_index=1:nrow(data)
data$lag1=Lag(data$Price_change,k=1)
data$lag2=Lag(data$Price_change,k=2)
data$lag3=Lag(data$Price_change,k=3)
data$lag4=Lag(data$Price_change,k=4)
data$lag5=Lag(data$Price_change,k=5)

temp=data.frame()
for( i in 1:nrow(Pos_loc) ){

  tempE0=Pos_loc[i,1]
  tempE6=Pos_loc[i,7]

  for(j in 0:6){
    data$HS_Eloc[Pos_loc[i,j+1]]=j


  }

  tempHS=data[tempE0:tempE6,]
  temp=rbind(temp,tempHS)


  data$HS_Eloc=rep(NA,nrow(data))

}
```

```r
data_HS=temp
rownames(data_HS)=1:nrow(data_HS)
```

get normal series dataset (without HS)

```r
temp1=data$org_index
temp2=data_HS$org_index

temp3=setdiff(temp1,temp2)

data_normal=data[data$org_index %in% temp3,]
```

#statistical feature #(measured by price change)

```r
library(PerformanceAnalytics)
```

```
##
## Attaching package: 'PerformanceAnalytics'

## The following object is masked from 'package:graphics':
##
##     legend
```

```r
pchg_norm=data_normal$Price_change
pchg_HS=data_HS$Price_change
#mean
mean(pchg_norm)
```

```
## [1] -0.001268602
```

```r
mean(pchg_HS)
```

```
## [1] -0.002329001
```

```r
#
t.test(pchg_norm,pchg_HS)
```

```
##
##  Welch Two Sample t-test
##
## data:  pchg_norm and pchg_HS
## t = 0.59211, df = 88188, p-value = 0.5538
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.002449703  0.004570500
## sample estimates:
##    mean of x    mean of y
## -0.001268602 -0.002329001
```

```r
# std
sd(pchg_norm)
```

```
## [1] 0.3223711
```

```r
sd(pchg_HS)
```

```
## [1] 0.3039662
```

```r
kurtosis(pchg_norm)
```

```
## [1] 6.842126
```

```r
kurtosis(pchg_HS)
```

```
## [1] 9.569954
```

```r
skewness(pchg_norm)
```

```
## [1] -0.1323418
```

```r
skewness(pchg_HS)
```

```
## [1] -0.1878772
```

#acf

```r
test=cor.test(pchg_norm,data_normal$lag1)

cor(pchg_norm,data_normal$lag1,use = "complete.obs")
```

```
##             Lag.1
## [1,] -0.02450746
```

```r
cor.test(pchg_HS,data_HS$lag1)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pchg_HS and data_HS$lag1
## t = 0.87819, df = 94399, p-value = 0.3798
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.003520911  0.009237226
## sample estimates:
##         cor
## 0.002858274
```

```r
cor(pchg_HS,data_HS$lag1,use = "complete.obs")
```

```
##             Lag.1
## [1,] 0.002858274
```

```r
cor.test(pchg_norm,data_normal$lag2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pchg_norm and data_normal$lag2
## t = -0.039069, df = 46630, p-value = 0.9688
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.009257196  0.008895373
## sample estimates:
##           cor
## -0.0001809269
```

```r
cor(pchg_norm,data_normal$lag2,use = "complete.obs")
```

```
##              Lag.2
## [1,] -0.0001809269
```

```r
cor.test(pchg_HS,data_HS$lag2)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pchg_HS and data_HS$lag2
## t = 0.51002, df = 94399, p-value = 0.61
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.004719176  0.008039030
## sample estimates:
##         cor
## 0.001659994
```

```r
cor(pchg_HS,data_HS$lag2,use = "complete.obs")
```

```
##             Lag.2
## [1,] 0.001659994
```

```r
cor.test(pchg_norm,data_normal$lag3)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pchg_norm and data_normal$lag3
```

```
## t = 3.3711, df = 46629, p-value = 0.0007494
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.00653396 0.02468230
## sample estimates:
##        cor
## 0.01560942
```

```
cor(pchg_norm,data_normal$lag3,use = "complete.obs")
```

```
##           Lag.3
## [1,] 0.01560942
```

```
cor.test(pchg_HS,data_HS$lag3)
```

```
##
##  Pearson's product-moment correlation
##
## data:  pchg_HS and data_HS$lag3
## t = 2.5632, df = 94399, p-value = 0.01037
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.001963232 0.014720585
## sample estimates:
##         cor
## 0.008342248
```

```
cor(pchg_HS,data_HS$lag3,use = "complete.obs")
```

```
##            Lag.3
## [1,] 0.008342248
```

```
# autocorrelation function and test


acf_normal=c()
acf_HS=c()
p_n=c()
p_HS=c()
temp.df=data[,1:6]

for (i in 1:10) {

temp.df$templag=Lag(temp.df$Price_change,k=i)
temp=data.frame()
for( i in 1:nrow(Pos_loc) ){
  tempE0=Pos_loc[i,1]
  tempE6=Pos_loc[i,7]
  tempHS=temp.df[tempE0:tempE6,]
  temp=rbind(temp,tempHS)
}
```

```
temp.df_HS=temp
temp1=temp.df$org_index
temp2=temp.df_HS$org_index
temp3=setdiff(temp1,temp2)
temp_normal=temp.df[temp.df$org_index %in% temp3,]

temp4=temp_normal$Price_change
temp5=temp_normal$templag
rho=cor(temp4,temp5,use = "complete.obs")
acf_normal=c(acf_normal,rho)
test=cor.test(temp4,temp5)
p_n=c(p_n,test$p.value)


temp4=temp.df_HS$Price_change
temp5=temp.df_HS$templag
rho=cor(temp4,temp5,use = "complete.obs")
acf_HS=c(acf_HS,rho)
test=cor.test(temp4,temp5)
p_HS=c(p_HS,test$p.value)



}

rbind(acf_normal,p_n)
```

```
##                    [,1]          [,2]        [,3]         [,4]          [,5]
## acf_normal -2.450746e-02 -0.0001809269 0.015609416 -0.007612013 -0.0174748222
## p_n         1.203529e-07  0.9688352912 0.000749379  0.100234817  0.0001608724
##                   [,6]          [,7]        [,8]        [,9]         [,10]
## acf_normal -0.003058925 -0.0153804887 0.007039897 0.007745732 -0.0001098937
## p_n         0.508924458  0.0008961192 0.128484141 0.094425283  0.9810693654
```

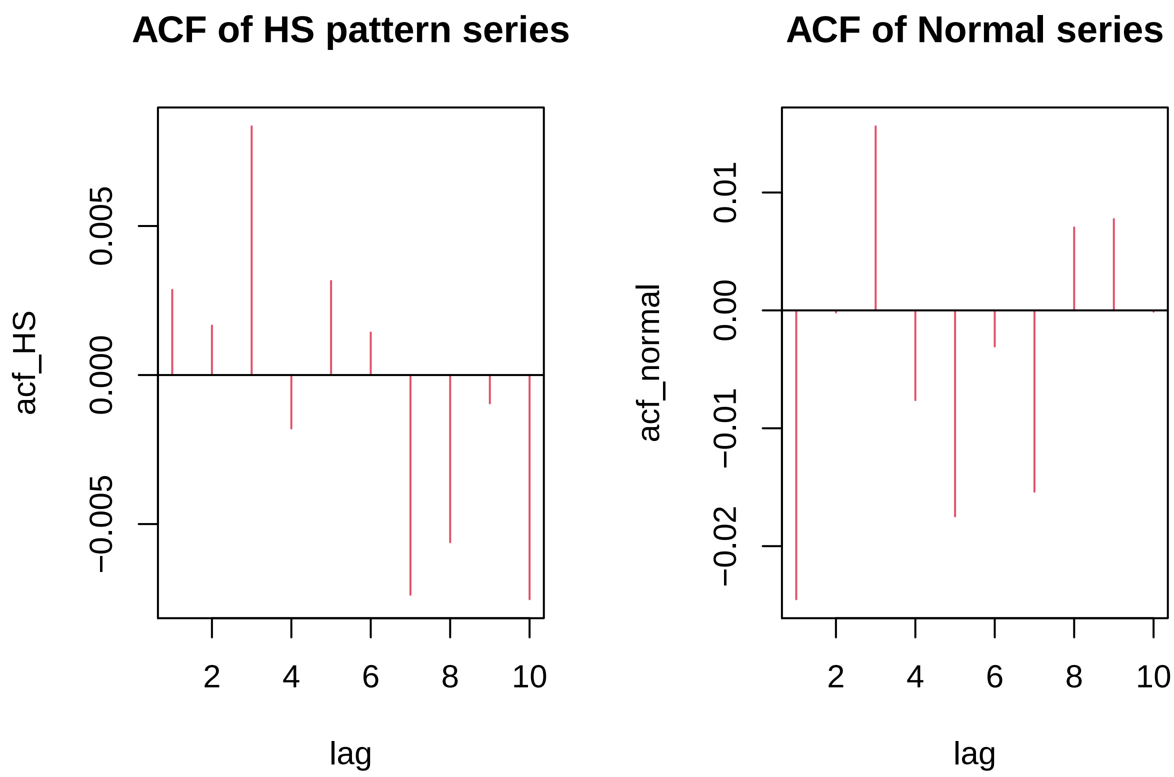
```
rbind(acf_HS,p_HS)
```

```
##              [,1]        [,2]        [,3]         [,4]        [,5]        [,6]
## acf_HS 0.002858274 0.001659994 0.008342248 -0.001793465 0.003157892 0.001427359
## p_HS   0.379842010 0.610035573 0.010372868  0.581612651 0.331925007 0.660989835
##                [,7]         [,8]          [,9]        [,10]
## acf_HS -0.007373657 -0.005610925 -0.0009492451 -0.007523373
## p_HS    0.023479684  0.084719933  0.7705547488  0.020803084
```

#plot acf

```
par(mfrow=c(1,2))
lag=1:10

plot(lag,acf_HS,col="2",type='h',main="ACF of HS pattern series")
abline(h=0)
```

```r
plot(lag,acf_normal,col="2",type='h',main="ACF of Normal series")
abline(h=0)
```



construct learning sample from data_HS

```r
Pos_data=Pos_loc[,1:4]
rownames(Pos_data)=1:nrow(Pos_data)

colnames(Pos_data)=c('tE0','tE1','tE2','tE3')


Pos_data$E0=rep(NA,nrow(Pos_data))
Pos_data$E1=rep(NA,nrow(Pos_data))
Pos_data$E2=rep(NA,nrow(Pos_data))
Pos_data$E3=rep(NA,nrow(Pos_data))
Pos_data$mean=rep(NA,nrow(Pos_data))

Pos_data$std=rep(NA,nrow(Pos_data))
Pos_data$kur=rep(NA,nrow(Pos_data))
Pos_data$skew=rep(NA,nrow(Pos_data))
Pos_data$acf1=rep(NA,nrow(Pos_data))
Pos_data$acf2=rep(NA,nrow(Pos_data))
```

```r
Pos_data$tail=rep(NA,nrow(Pos_data))
Pos_data$HS=rep(1,nrow(Pos_data))

for( i in 1:nrow(Pos_data) ){

  tempE0=Pos_loc[i,1]
  tempE3=Pos_loc[i,4]
  tempE2=Pos_loc[i,3]
  tempE1=Pos_loc[i,2]


  tempHS=data[tempE0:tempE3,]
  Pos_data$E0[i]=data$Price[tempE0]
  Pos_data$E1[i]=data$Price[tempE1]
  Pos_data$E2[i]=data$Price[tempE2]
  Pos_data$E3[i]=data$Price[tempE3]
  Pos_data$mean[i]=mean(tempHS$Price_change)
  Pos_data$std[i]=sd(tempHS$Price_change)
  Pos_data$kur[i]=kurtosis(tempHS$Price_change)
  Pos_data$skew[i]=skewness(tempHS$Price_change)
  Pos_data$acf1[i]=cor(tempHS$Price_change,tempHS$lag1,use = "complete.obs")
  Pos_data$acf2[i]=cor(tempHS$Price_change,tempHS$lag2,use = "complete.obs")
 Pos_data$tail[i]=data$Price[tempE3+2]


}
```

#negative case

```r
temp=data.frame()
for( i in 1:nrow(Neg_loc) ){

  tempE0=Neg_loc[i,1]
  tempE4=Neg_loc[i,5]

  for(j in 0:4){
    data$HHS_Eloc[Neg_loc[i,j+1]]=j


  }

  tempHHS=data[tempE0:tempE4,]
  temp=rbind(temp,tempHHS)


  data$HHS_Eloc=rep(NA,nrow(data))

}




data_HHS=temp
```

```
rownames(data_HHS)=1:nrow(data_HHS)
```

construct learning sample from data_HS

```
Neg_data=Neg_loc[,1:4]
rownames(Neg_data)=1:nrow(Neg_data)


Neg_data$E0=rep(NA,nrow(Neg_data))
Neg_data$E1=rep(NA,nrow(Neg_data))
Neg_data$E2=rep(NA,nrow(Neg_data))
Neg_data$E3=rep(NA,nrow(Neg_data))

Neg_data$mean=rep(NA,nrow(Neg_data))

Neg_data$std=rep(NA,nrow(Neg_data))
Neg_data$kur=rep(NA,nrow(Neg_data))
Neg_data$skew=rep(NA,nrow(Neg_data))
Neg_data$acf1=rep(NA,nrow(Neg_data))
Neg_data$acf2=rep(NA,nrow(Neg_data))
Neg_data$tail=rep(NA,nrow(Neg_data))
Neg_data$HS=rep(0,nrow(Neg_data))



for( i in 1:nrow(Neg_data) ){

  tempE0=Neg_loc[i,1]
  tempE3=Neg_loc[i,4]
  tempE2=Neg_loc[i,3]
  tempE1=Neg_loc[i,2]


  tempHHS=data[tempE0:tempE3,]
  Neg_data$E0[i]=data$Price[tempE0]
  Neg_data$E1[i]=data$Price[tempE1]
  Neg_data$E2[i]=data$Price[tempE2]
  Neg_data$E3[i]=data$Price[tempE3]


  Neg_data$mean[i]=mean(tempHHS$Price_change)
  Neg_data$std[i]=sd(tempHHS$Price_change)
  Neg_data$kur[i]=kurtosis(tempHHS$Price_change)
  Neg_data$skew[i]=skewness(tempHHS$Price_change)
  Neg_data$acf1[i]=cor(tempHHS$Price_change,tempHHS$lag1,use = "complete.obs")
  Neg_data$acf2[i]=cor(tempHHS$Price_change,tempHHS$lag2,use = "complete.obs")
 Neg_data$tail[i]=data$Price[tempE3+2]


}
```

```r
write.csv(Pos_data,file="Positive Case.csv")

write.csv(Neg_data,file="Negative Case.csv")
```

```r
Pos_data <- read.csv("Positive Case.csv")[,-1]
Neg_data <- read.csv("Negative Case.csv")[,-1]
data <- read.csv("data.csv")[-1]
Pos_loc <- read.csv("Pos_loc.csv")[-1]
Neg_loc <- read.csv("Neg_loc.csv")[-1]
```

```r
# slope for the least square line
#Pos data slope
test=Pos_loc[,1:4]
rownames(test)=1:nrow(test)
colnames(test)=c('tE0','tE1','tE2','tE3')
test$xmean=rep(NA,nrow(test))
test$temp_mean = rep(NA,nrow(test))
test$beta1 = rep(NA,nrow(test))
test$beta0 = rep(NA, nrow(test))

for( i in 1:nrow(test) ){
  x = data$Price_change[test[i,1] : test[i,4]]
  test$xmean[i]=mean(x)
  temp = test[i,1] : test[i,4] - test[i,1] + 1
  test$temp_mean[i] = mean(temp)
  test$beta1[i] = sum((temp - test$temp_mean[i])*(x - test$xmean[i]))/ sum((temp - test$temp_mean[i])^2)
  test$beta0[i] = test$xmean[i] - test$beta1[i]*test$temp_mean[i]
  xbar = test$beta0[i] + test$beta1[i] * temp
}


#Neg data slope
test_neg=Neg_loc[,1:4]
rownames(test_neg)=1:nrow(test_neg)
colnames(test_neg)=c('tE0','tE1','tE2','tE3')
test_neg$xmean=rep(NA,nrow(test_neg))
test_neg$temp_mean = rep(NA,nrow(test_neg))
test_neg$beta1 = rep(NA,nrow(test_neg))
test_neg$beta0 = rep(NA, nrow(test_neg))

for( i in 1:nrow(test_neg) ){
  x = data$Price_change[test_neg[i,1] : test_neg[i,4]]
  test_neg$xmean[i]=mean(x)
  temp = test_neg[i,1] : test_neg[i,4] - test_neg[i,1] + 1
  test_neg$temp_mean[i] = mean(temp)
  test_neg$beta1[i] = sum((temp - test_neg$temp_mean[i])*(x - test_neg$xmean[i]))/ sum((temp - test_neg
  test_neg$beta0[i] = test_neg$xmean[i] - test_neg$beta1[i]*test_neg$temp_mean[i]
  xbar = test_neg$beta0[i] + test_neg$beta1[i] * temp
}
# beta1 is a slope
# beta0 is an intercept
```

```r
# N1: the number of mean crossing
N1_val <- rep(NA, nrow(test))
for(j in 1:nrow(test)){
      N1 <- function(i){
      x  = data$Price_change[test[i,1] : test[i,4]]
      n1 = rep(NA, length(x)-1)
                for (j in 1 : length(x)-1){
                 n1[j] <- ifelse ((x[j] -  mean(x))*(x[j+1] -  mean(x)) < 0, 1, 0)
                 N1 <- sum(n1)}
      return(N1)}
  N1_val[j] <- N1(j)
}

N1_val_neg <- rep(NA, nrow(test_neg))
for(j in 1:nrow(test_neg)){
      N1 <- function(i){
      x  = data$Price_change[test_neg[i,1] : test_neg[i,4]]
      n1 = rep(NA, length(x)-1)
                for (j in 1 : length(x)-1){
                 n1[j] <- ifelse ((x[j] -  mean(x))*(x[j+1] -  mean(x)) < 0, 1, 0)
                 N1 <- sum(n1)}
      return(N1)}
  N1_val_neg[j] <- N1(j)
}
#N1 values
```

```r
#N2: the number of least square line crossing
N2_val <- rep(NA, nrow(test))
for(j in 1:nrow(test)){
        N2 <- function(i){
          x  = data$Price_change[test[i,1] : test[i,4]]
          temp = test[i,1] : test[i,4] - test[i,1] + 1
          xbar = test$beta0[i] + test$beta1[i] * temp
          n2 = rep(NA, length(x)-1)
          for (j in 1 : length(x)-1){
                    n2[j] <- ifelse ((x[j] -  xbar[j])*(x[j+1] -  xbar[j+1]) < 0, 1, 0)
                    N2 <- sum(n2)}
          return(N2)}
        N2_val[j] <- N2(j)
}

N2_val_neg <- rep(NA, nrow(test_neg))
for(j in 1:nrow(test_neg)){
        N2 <- function(i){
          x  = data$Price_change[test_neg[i,1] : test_neg[i,4]]
          temp = test_neg[i,1] : test_neg[i,4] - test_neg[i,1] + 1
          xbar = test_neg$beta0[i] + test_neg$beta1[i] * temp
          n2 = rep(NA, length(x)-1)
          for (j in 1 : length(x)-1){
                    n2[j] <- ifelse ((x[j] -  xbar[j])*(x[j+1] -  xbar[j+1]) < 0, 1, 0)
                    N2 <- sum(n2)}
          return(N2)}
        N2_val_neg[j] <- N2(j)
```

```r
}
#N2 Values
```

```r
#APML: the area between the pattern and its mean line
#sum(abs(x - xmean))
APML_val <- rep(NA, nrow(test))
for(j in 1:nrow(test)){
        APML <- function(i){
          x  = data$Price_change[test[i,1] : test[i,4]]
          APML <- sum(abs(x - mean(x)))
          return(APML)}
        APML_val[j] <- APML(j)
}

APML_val_neg <- rep(NA, nrow(test_neg))
for(j in 1:nrow(test_neg)){
        APML <- function(i){
          x  = data$Price_change[test_neg[i,1] : test_neg[i,4]]
          APML <- sum(abs(x - mean(x)))
          return(APML)}
        APML_val_neg[j] <- APML(j)
}
#APML Values
```

```r
#APSL: the area between the pattern and its least squares line
#sum(abs(x - xbar2))
APSL_val <- rep(NA, nrow(test))
for(j in 1:nrow(test)){
        APSL <- function(i){
          x  = data$Price_change[test[i,1] : test[i,4]]
          temp = test[i,1]:test[i,4] - test[i,1] + 1
          xbar = test$beta0[i] + test$beta1[i] * temp
          APSL = sum(abs(x - xbar))
          return(APSL)}
        APSL_val[j] <- APSL(j)
}

APSL_val_neg <- rep(NA, nrow(test_neg))
for(j in 1:nrow(test_neg)){
        APSL <- function(i){
          x  = data$Price_change[test_neg[i,1] : test_neg[i,4]]
          temp = test_neg[i,1] : test_neg[i,4] - test_neg[i,1] + 1
          xbar = test_neg$beta0[i] + test_neg$beta1[i] * temp
          APSL = sum(abs(x - xbar))
          return(APSL)}
        APSL_val_neg[j] <- APSL(j)
}
#APSL Values
APSL(1)
```

```
## [1] 3.450909
```

```
APSL(2)
```

```
## [1] 13.41637
```

```
#AAS: the area between the pattern and the line segments
#sum(abs(xmean - xbar))
AAS_val <- rep(NA, nrow(test))
for(j in 1:nrow(test)){
          AAS <- function(i){
            x  = data$Price_change[test[i,1] : test[i,4]]
            temp = test[i,1]:test[i,4] - test[i,1] + 1
            xbar = test$beta0[i] + test$beta1[i] * temp
            AAS <- sum(abs(mean(x) - xbar))
            return(AAS)}
          AAS_val[j] <- AAS(j)
}

AAS_val_neg <- rep(NA, nrow(test_neg))
for(j in 1:nrow(test_neg)){
          AAS <- function(i){
            x  = data$Price_change[test_neg[i,1] : test_neg[i,4]]
            temp = test_neg[i,1] : test_neg[i,4] - test_neg[i,1] + 1
            xbar = test_neg$beta0[i] + test_neg$beta1[i] * temp
            AAS <- sum(abs(mean(x) - xbar))
            return(AAS)}
          AAS_val_neg[j] <- AAS(j)
}
#AAS Values
AAS(1)
```

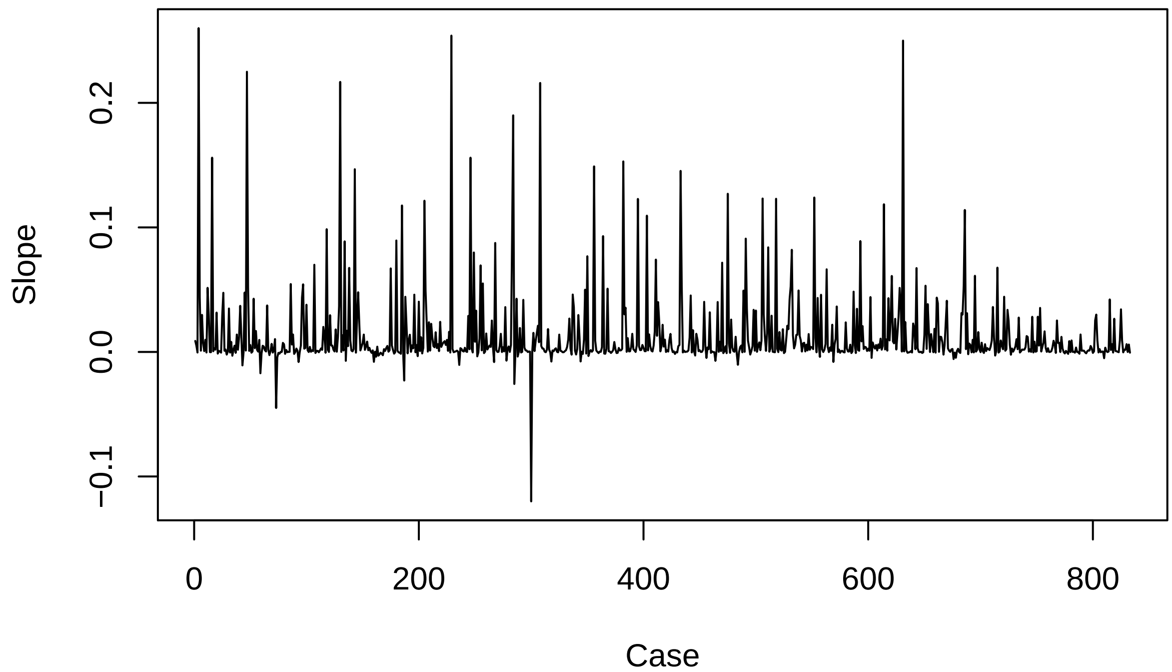
```
## [1] 2.127273
```

```
AAS(2)
```

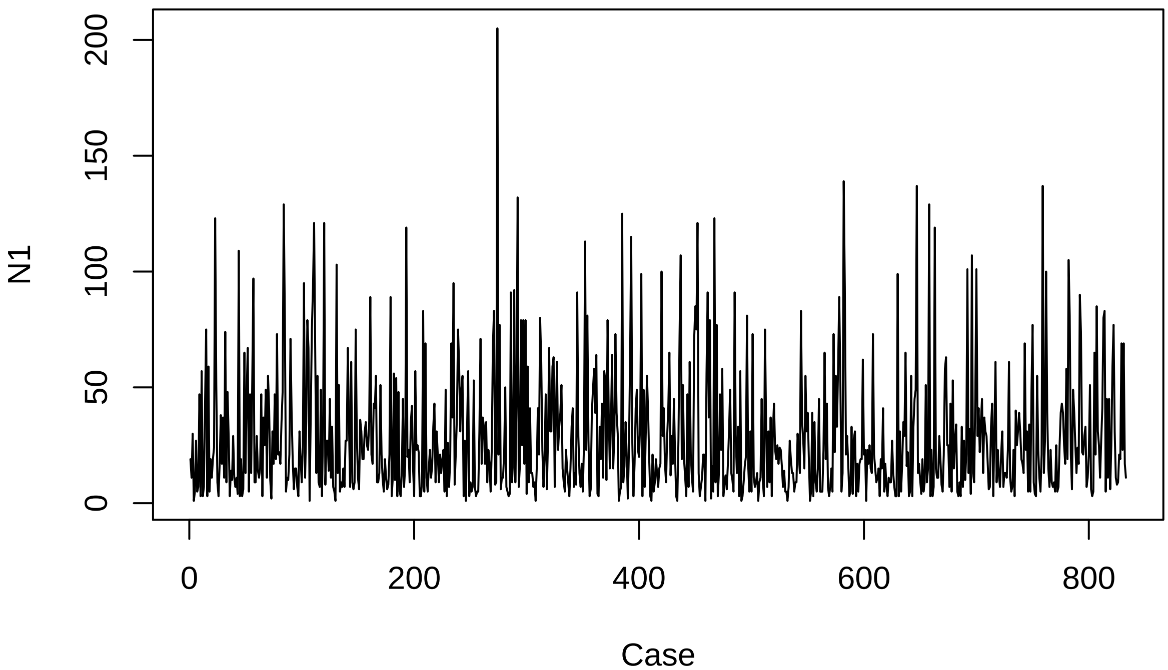
```
## [1] 1.049291
```

```
Pos_data$beta1 <- test$beta1
Pos_data$beta0 <- test$beta0
Pos_data$N1_val <- N1_val
Pos_data$N2_val <- N2_val
Pos_data$APML_val <- APML_val
Pos_data$APSL_val <- APSL_val
Pos_data$AAS_val <- AAS_val

Neg_data$beta1 <- test_neg$beta1
Neg_data$beta0 <- test_neg$beta0
Neg_data$N1_val <- N1_val_neg
Neg_data$N2_val <- N2_val_neg
Neg_data$APML_val <- APML_val_neg
Neg_data$APSL_val <- APSL_val_neg
Neg_data$AAS_val <- AAS_val_neg
View(Pos_data)
```

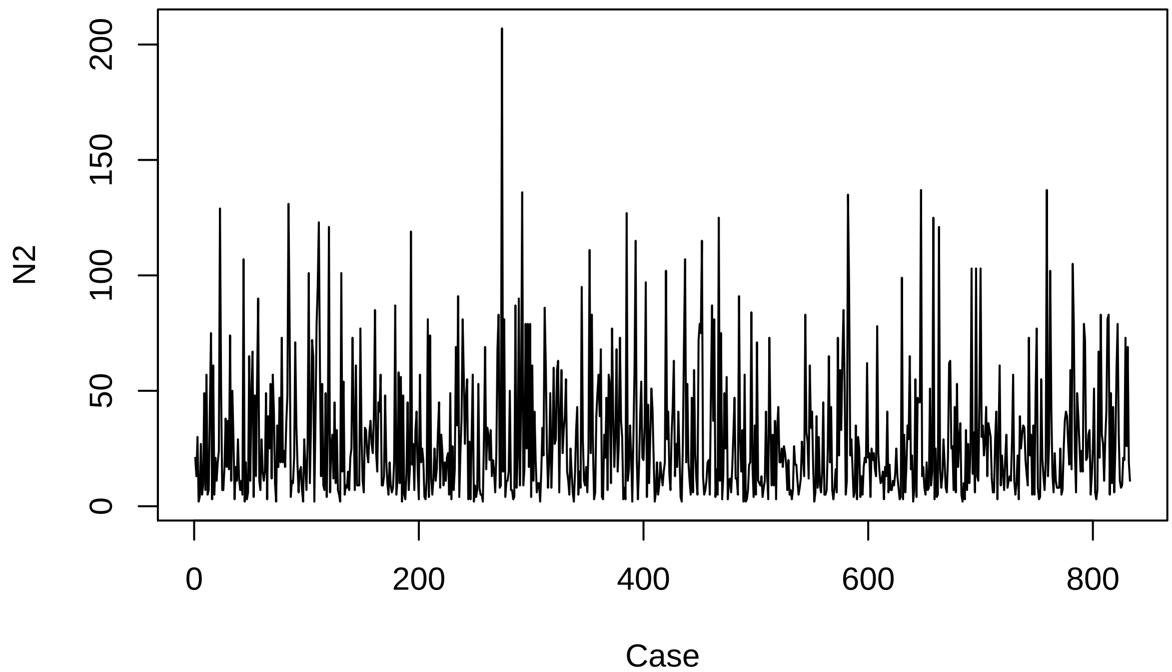
```r
plot(Pos_data$beta1, type = "l", ylab = "Slope", xlab = "Case")
```



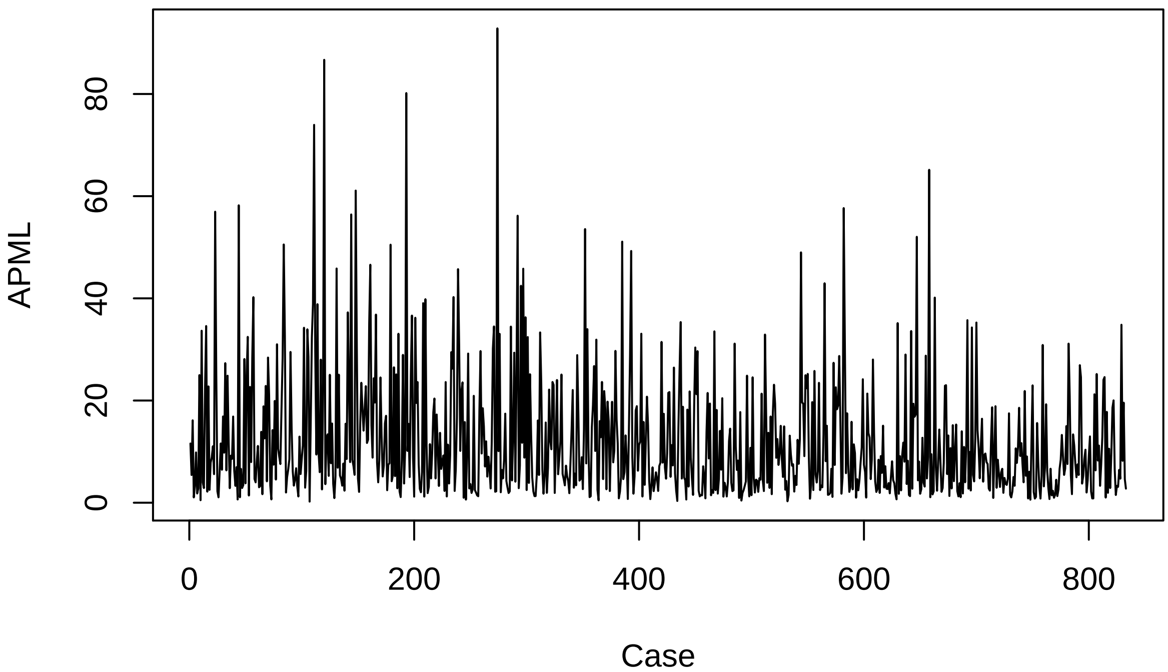

```r
plot(Pos_data$N1_val, type = "l", ylab = "N1", xlab = "Case")
```



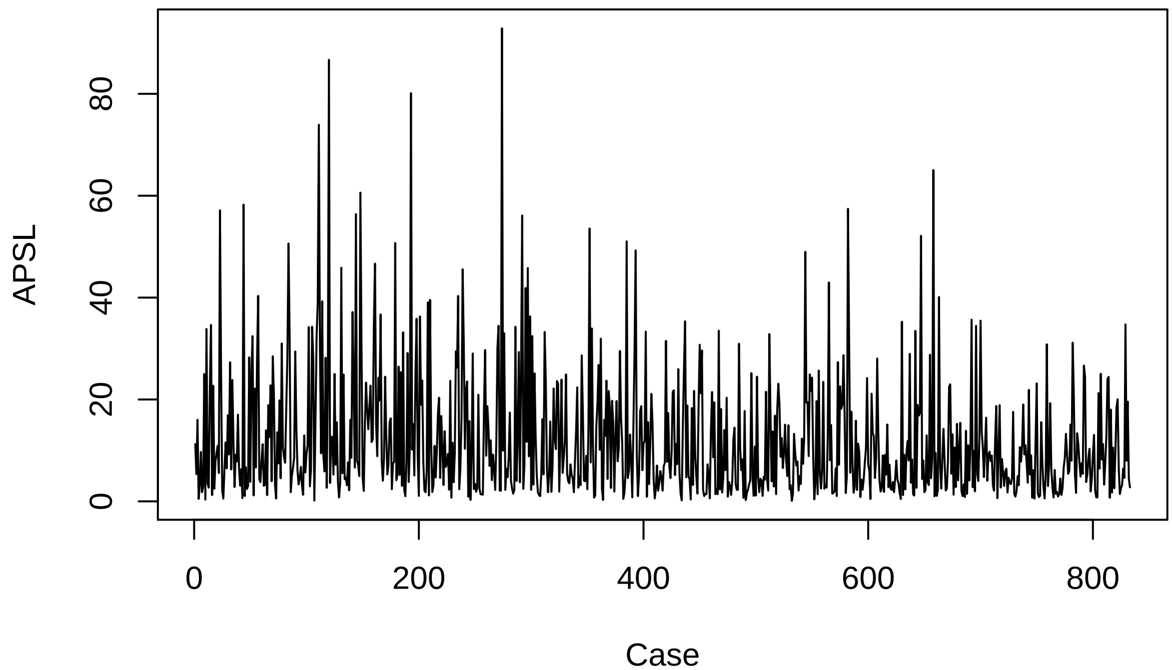

```
plot(Pos_data$N2_val, type = "l", ylab = "N2", xlab = "Case")
```



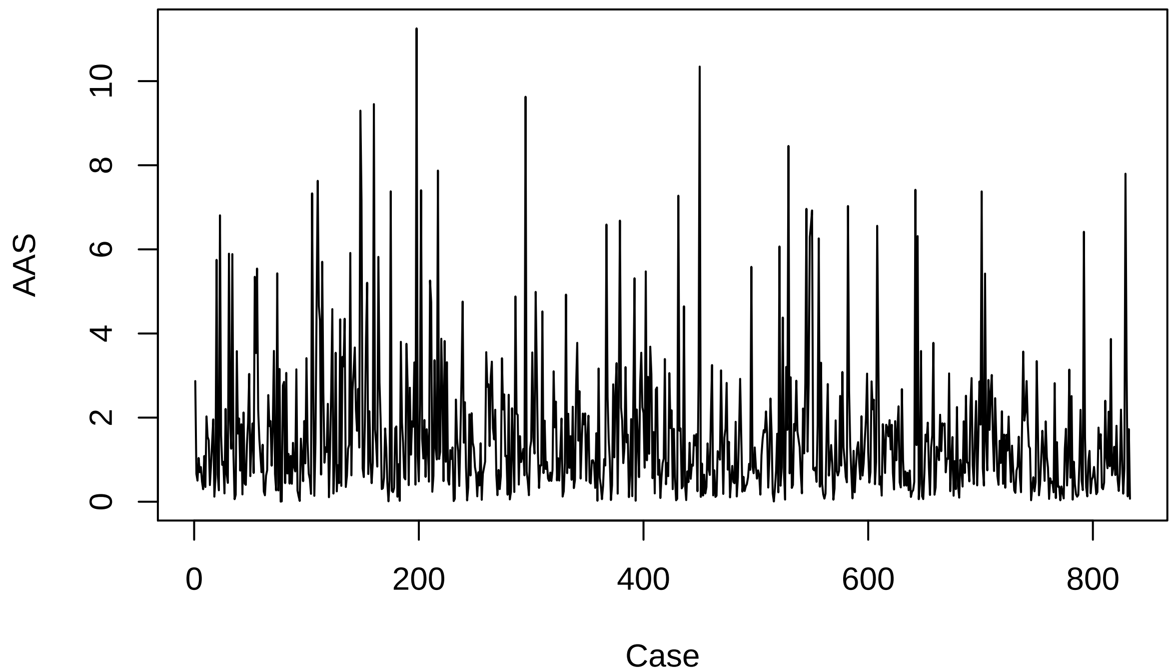

```r
plot(Pos_data$APML_val, type = "l", ylab = "APML", xlab = "Case")
```



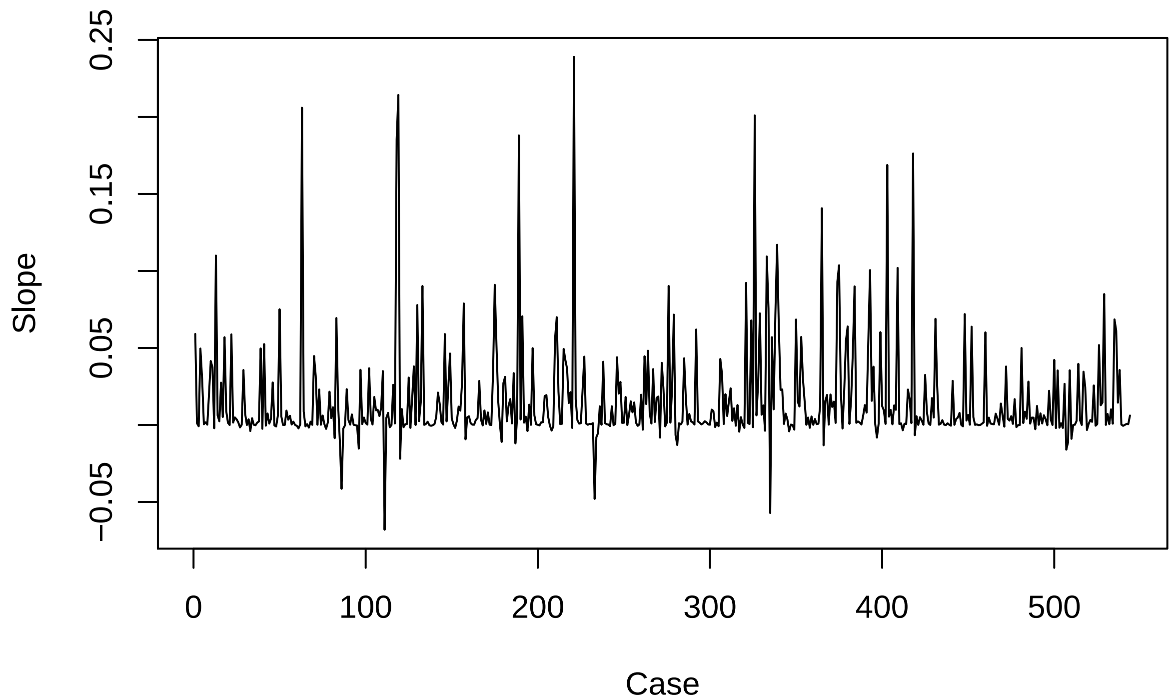

```r
plot(Pos_data$APSL_val, type = "l", ylab = "APSL", xlab = "Case")
```



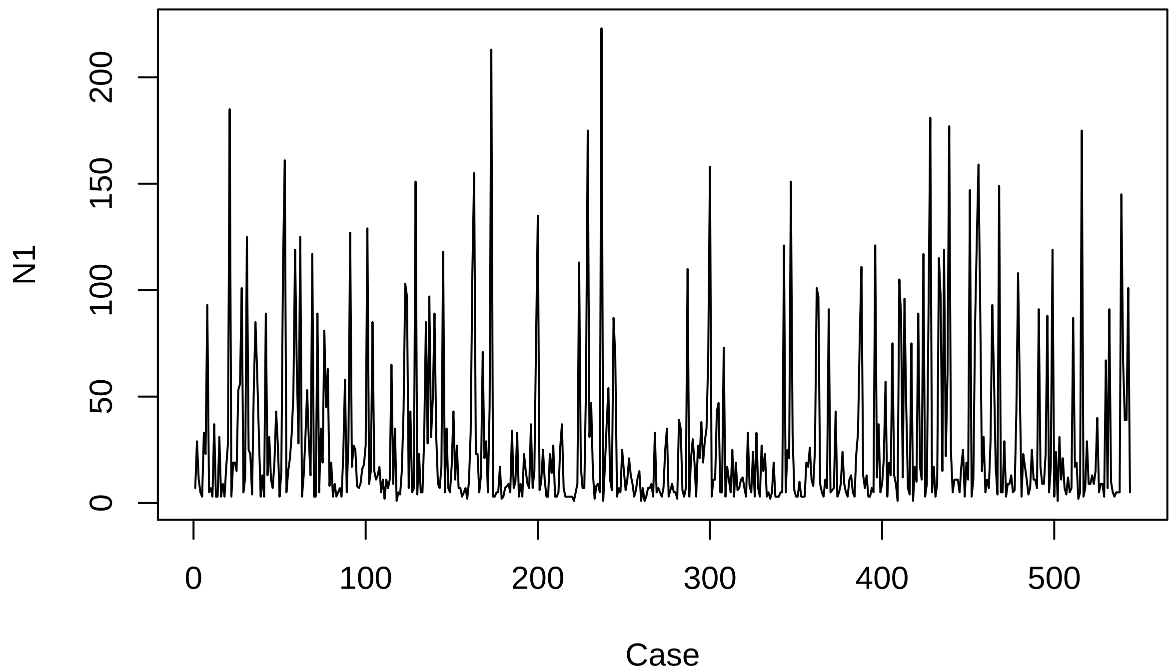

```r
plot(Pos_data$AAS_val, type = "l", ylab = "AAS", xlab = "Case")
```



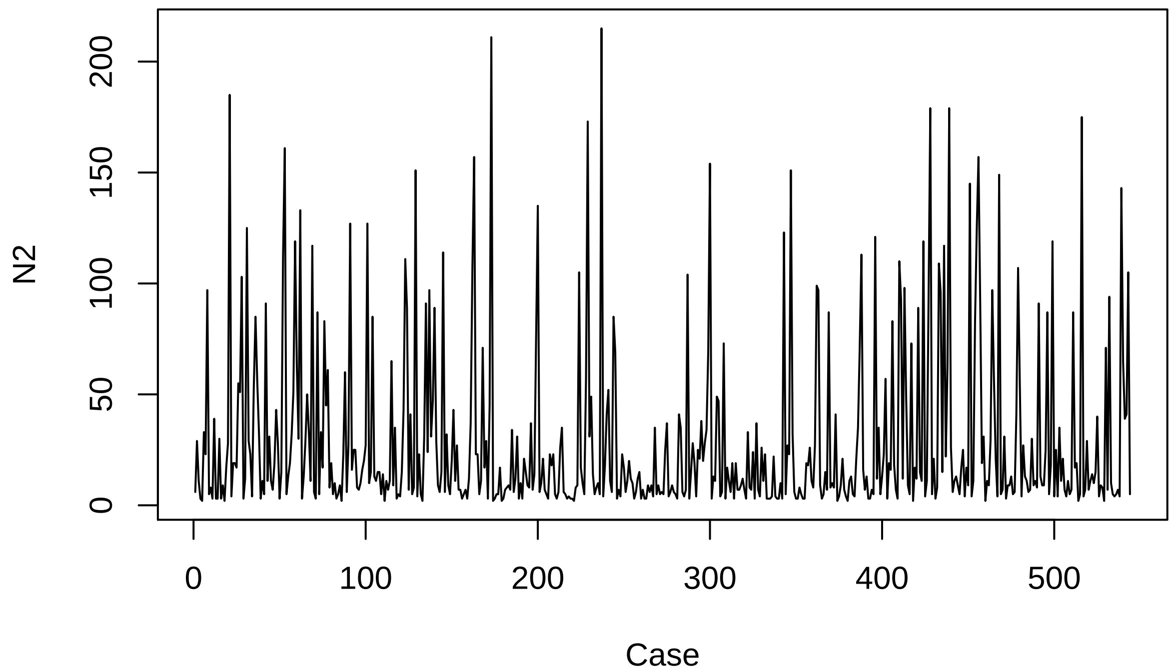

```
plot(Neg_data$beta1, type = "l", ylab = "Slope", xlab = "Case")
```



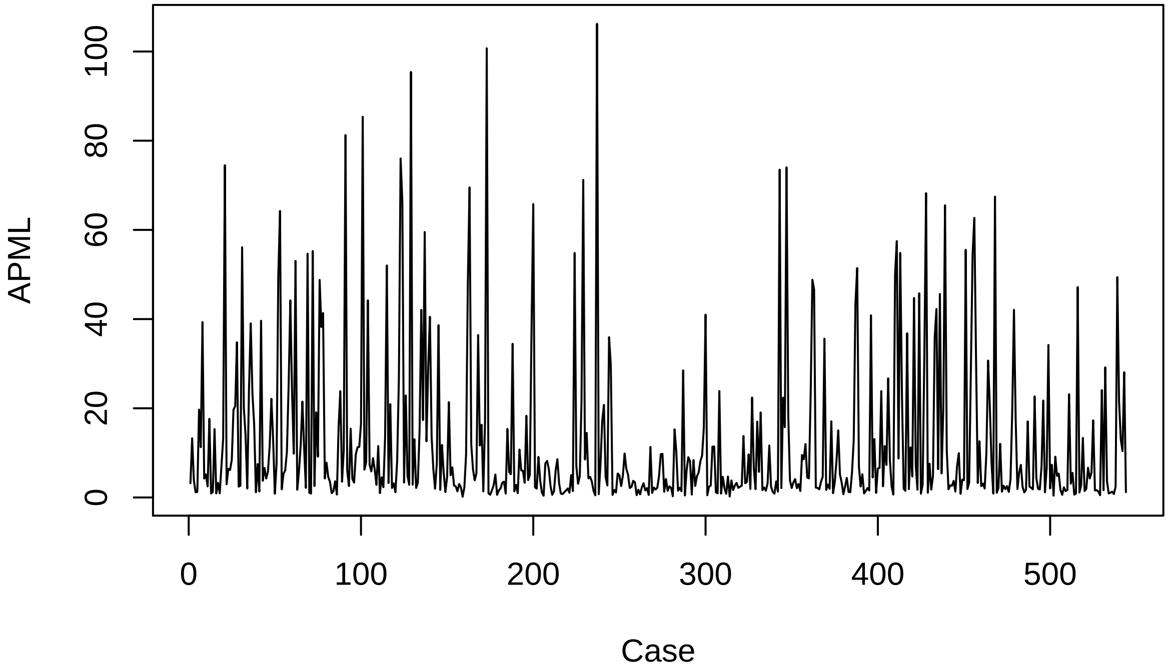

```
plot(Neg_data$N1_val, type = "l", ylab = "N1", xlab = "Case")
```



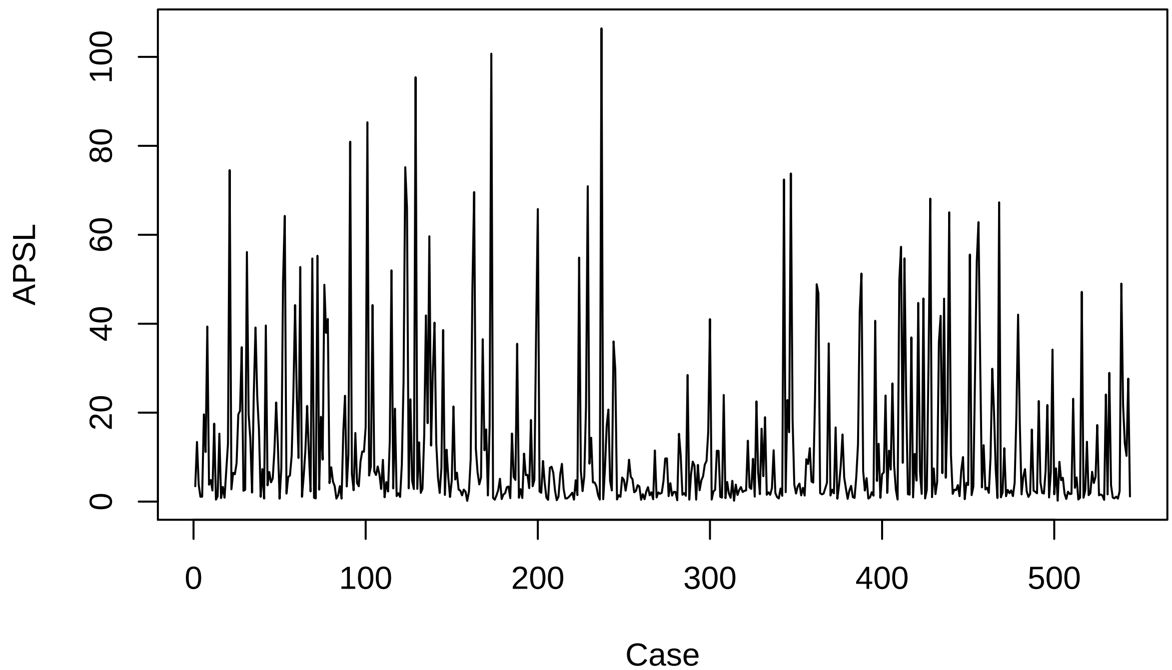

```r
plot(Neg_data$N2_val, type = "l", ylab = "N2", xlab = "Case")
```



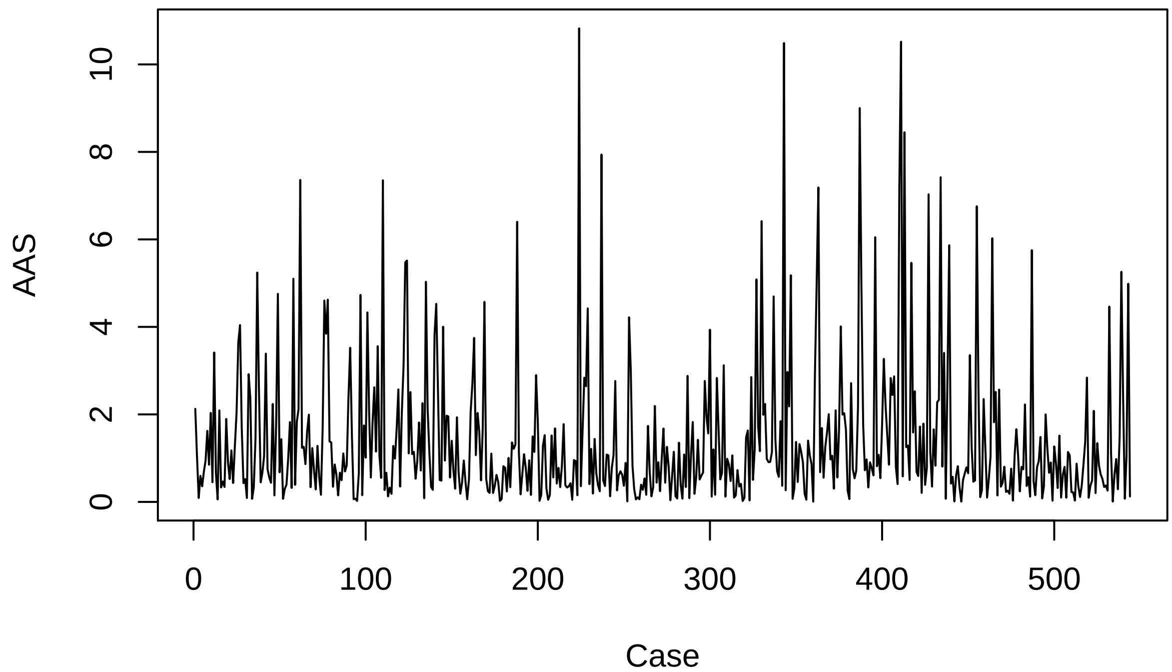

```
plot(Neg_data$APML_val, type = "l", ylab = "APML", xlab = "Case")
```




```
plot(Neg_data$APSL_val, type = "l", ylab = "APSL", xlab = "Case")
```




```
plot(Neg_data$AAS_val, type = "l", ylab = "AAS", xlab = "Case")
```




```
##
Pos_data$duration <- Pos_data$tE3 - Pos_data$tE0 +1
Pos_data$Height <- data$Price[Pos_data$tE3+1] - Pos_data$E1
```

```
train_pos <- read.csv("Positive Case 2.csv")
train_neg <- read.csv("Negative Case 2.csv")
data <- read.csv("data.csv")[-1]
train_neg$duration <- train_neg$tE3 - train_neg$tE0 +1
train_neg$Height <- data$Price[train_neg$tE3+1] - train_neg$E1

train_pos <- train_pos[, 10:26]
train_neg <- train_neg[, 10:26]
train <- as.data.frame(rbind(train_pos, train_neg))
train$HS <- as.factor(train$HS)

set.seed(23)
test_idx <- sample(1:nrow(train), 0.2*nrow(train))
test <- train[test_idx, ]
train <- train[-test_idx, ]
```

Fit Random Forest

```r
library(randomForest)
```

```
## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.
```

```r
p <- ncol(train) - 1
rf <- randomForest(HS ~ ., data = train, importance = T)
pred <- predict(rf, test[, -8]) # exclude response "HS"
mean(pred != test[, 8])
```

```
## [1] 0.3418182
```

```r
importance(rf)
```

```
##                    0          1 MeanDecreaseAccuracy MeanDecreaseGini
## mean      2.11048668  7.4352733             7.886423         38.44622
## std       1.69743050  1.6627266             2.594048         30.71568
## kur      -4.83904735 13.1573007             9.533088         31.21869
## skew     -0.94585239  2.7502245             1.555971         33.13779
## acf1     -4.11964067  8.1923224             4.084280         32.38885
## acf2     -5.12071886  7.5686805             2.714881         32.90756
## tail      1.27210158  4.7506397             4.512675         38.09895
## beta1    -6.02796956 10.8657623             6.288035         30.74915
## beta0    -1.86688698  7.1378215             5.575700         29.87971
## N1_val    0.02888171  8.9100810            10.500776         21.44085
## N2_val    3.74362050  7.5608231            11.717020         24.40200
## APML_val -0.66012015 12.0080665            14.028191         36.45904
## APSL_val  0.14040868 10.0926618            12.844809         34.33382
## AAS_val   8.47712546  0.6552722             7.364972         30.91101
## duration -1.21653250 11.1983417            11.844243         28.14715
## Height   27.50924653 -3.8014206            18.895889         56.71906
```

```r
rf$confusion
```

```
##     0   1 class.error
## 0 181 264   0.5932584
## 1 149 508   0.2267884
```

Fit Decision Tree

```r
library(rpart)
library(rpart.plot)
dt <- rpart(HS ~., data = train, method = "class",
            parms = list(split='information'))
pred.dt <- predict(dt, test[, -8], type = "class")
mean(pred.dt != test[, 8])
```

```
## [1] 0.3381818
```

```r
rpart.plot(dt)
```