Homework #1 - Solution and Grading Guideline

Larry Lei Hua

(All rights are reserved; please use it for your reference only. No copies or distributions in any form are allowed without my written permission)

Jan 27, 2019

In total there are 30 pts:

(1) 5 pts for homework policy and presentation of the homework.

- If the files are not complete or not zipped into one file (should be one zipped file with one pdf and one Rmd), deduct 2 pts
- If the overall presentation is messy and not organized, deduct 3 pts

(2) 5 pts for Question 1

- The critical place to check is whether the three decimal of milliseconds have been correctly displayed

(3) 20 pts for Question 2

- 4 pts for each bar, and 4 pts for the comments
- Do not need to check the detailed R codes, but the basic patterns, in particular the sparse and dense patterns should match what have been provided here; however the plots do not need to be exact the same.
- Volume runs bars is little harder, and please grade it based on the student's effort. If they seem have worked hard to implement it but the patterns are completely wrong, deduct at most 2 pts for such a case.
- The comments do not need to be exactly the same as long as they read reasonable. However, if they do not comment anything about the sparse and dense patterns, deduct at least 1 pt.

(4) Only grade the last attempt. If the homework was submitted late but within 24 hours, all the above pts are deducted from 15 pts until 0. If the homework was submitted late more than 24 hours, 0 will be the mark.

(5) Under such a general guideline, the grader has some room to decide on the detailed points to use as long as they are consistent among all the students. Please let me know should you have any further questions about the solution and grading.

(Please read the Homework Policy before you start)

Description: In this homework, you will practice how to implement algorithms using R, how to construct various structured financial data from unstructured tick data, and how to compare results from different bars.

Dataset: Please download the data of the Amazon stock from here. Please read the readme file and the following questions will be based on the message tick data "AMZN_2012-06-21_34200000_57600000_message_10.csv".

Practices:

1. Practice how to use R package **lubridate** to correctly convert the timestamp to human readable ones, and keep the decimals for millisecond information. Print out the first 5 rows of the converted data.

```
library(lubridate)
options(digits.secs=3)
dat <- read.csv("~/Dropbox/Teaching/STAT430/datasets/AMZN_2012-06-21_34200000_57600000_message_10.csv",
                header = F)
names(dat) <- c("Time", "Type", "OrderID", "Size", "Price", "Direction")
dat$Size <- as.numeric(dat$Size)
dat$Price <- as.numeric(dat$Price)
demodate <- "2012-06-21"
dat$tStamp <- as_datetime(demodate, tz="US/Eastern") + dat$Time
dat_exc <- subset(dat, Type %in% c(4,5)) ## data with transactions
head(dat_exc)
```

```
##        Time Type  OrderID Size   Price Direction                  tStamp
## 1  34200.02    5        0    1 2238200        -1 2012-06-21 09:30:00.017
## 33 34200.19    4 11885113   21 2238100         1 2012-06-21 09:30:00.190
## 34 34200.19    4 11534792   26 2237500         1 2012-06-21 09:30:00.190
## 38 34200.37    5        0  100 2238400        -1 2012-06-21 09:30:00.372
## 39 34200.38    5        0  100 2238400        -1 2012-06-21 09:30:00.375
## 41 34200.38    5        0  100 2238600        -1 2012-06-21 09:30:00.383
```

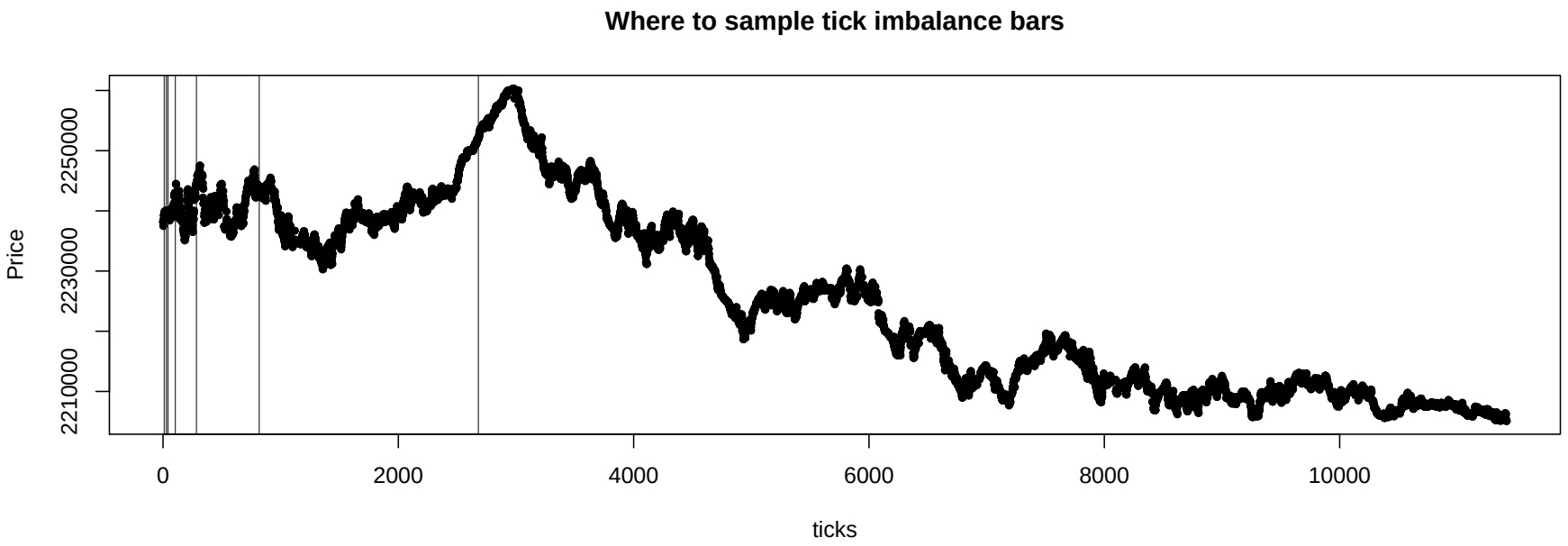
2. Develop R functions for the following bars, and try these functions with the order message data with Type == 4 or 5. You should submit the R codes, and summarize your findings, comparisons, and your thoughts.
   - tick imbalance bars

```
#' The auxiliary function b_t for constructing tick imbalance bars. The first b_t is assigned the value 0
#' because no information is available
#'
#' @param dat dat input with at least the following columns: Price
#' @examples
#'
```

```r
#' set.seed(1)
#' dat <- data.frame(Price = c(rep(0.5, 4), runif(10)))
#'
#' b_t <- imbalance_tick(dat)
#'
#' @export
imbalance_tick <- function(dat)
{
  n <- length(dat$Price)
  imbalance <- rep(0, n)
  price_diff <- diff(dat$Price)
  for(i in 2:n)
  {
    imbalance[i] <- sign(price_diff[i-1])*(price_diff[i-1]!=0) + imbalance[i-1]*(price_diff[i-1]==0)
  }
  imbalance
}

#' Tstar index for Tick Imbalance Bars (bar_tib)
#' @param dat dat input with at least the following columns: Price
#' @param w0 the time window length of the first bar
#' @param bkw_T backward window length when using pracma::movavg for exponentially weighted average T
#' @param bkw_b backward window length when using pracma::movavg for exponentially weighted average b_t
#'
#' @return a vector for the lengths of the tick imbalance bars. For example, if the return is c(10,26),
#' then the 2 tick imbalance bars are (0,10] and (10, 36]
#'
#' @examples
#'
#' set.seed(1)
#' dat <- data.frame(Price = c(rep(0.5, 4), runif(50)))
#' T_tib <- Tstar_tib(dat)
#' b_t <- imbalance_tick(dat)
#' cumsum(b_t)[cumsum(T_tib)] # check the accumulated b_t's where the imbalances occur
#'
#' @export
Tstar_tib <- function(dat, w0=10, bkw_T=5, bkw_b=5)
{
```

```
  nx <- dim(dat)[1]
  b_t <- imbalance_tick(dat)
  w0 <- max(min(which(cumsum(b_t) != 0)), w0) # fix the case when there are always 0 at the beginning
  Tvec <- w0
  E0T <- Tvec
  repeat
  {
    T_last <- sum(Tvec) # the previous T that has been calculated
    nbt <- min(bkw_b, T_last - 1)
    PminusP <- pracma::movavg(b_t[1:T_last], n=nbt, type="e")
    PminusP <- tail(PminusP,1) # the last one is what we need
    b_t_Expected <- E0T*abs(PminusP)
    b_t_psum <- abs(cumsum(b_t[-(1:T_last)]))

    # if max(b_t_psum) < b_t_Expected, then there is no chance of tick imbalance
    if(max(b_t_psum) < b_t_Expected){break}else
    {
      T_new <- min(which(b_t_psum >= b_t_Expected))
    }
    T_last <- T_last + T_new
    if(T_last > nx){break}else
    {
      Tvec <- c(Tvec, T_new)
      nTvec <- length(Tvec)
      if(nTvec <= 2)
      {
        E0T <- mean(Tvec) # not enough T for exponential weighted average, so use the mean
      }else
      {
        nT <- min(bkw_T, length(Tvec)-1)
        E0T <- pracma::movavg(Tvec[1:nTvec], n=nT, type = "e")
        E0T <- tail(E0T,1)
      }
    }
  }
  return(Tvec)
}
```

```r
T_tib <- Tstar_tib(dat_exc)
plot(dat_exc$Price, pch=20, xlab="ticks", ylab="Price", main="Where to sample tick imbalance bars")
abline(v=cumsum(T_tib), lwd=0.2)
```

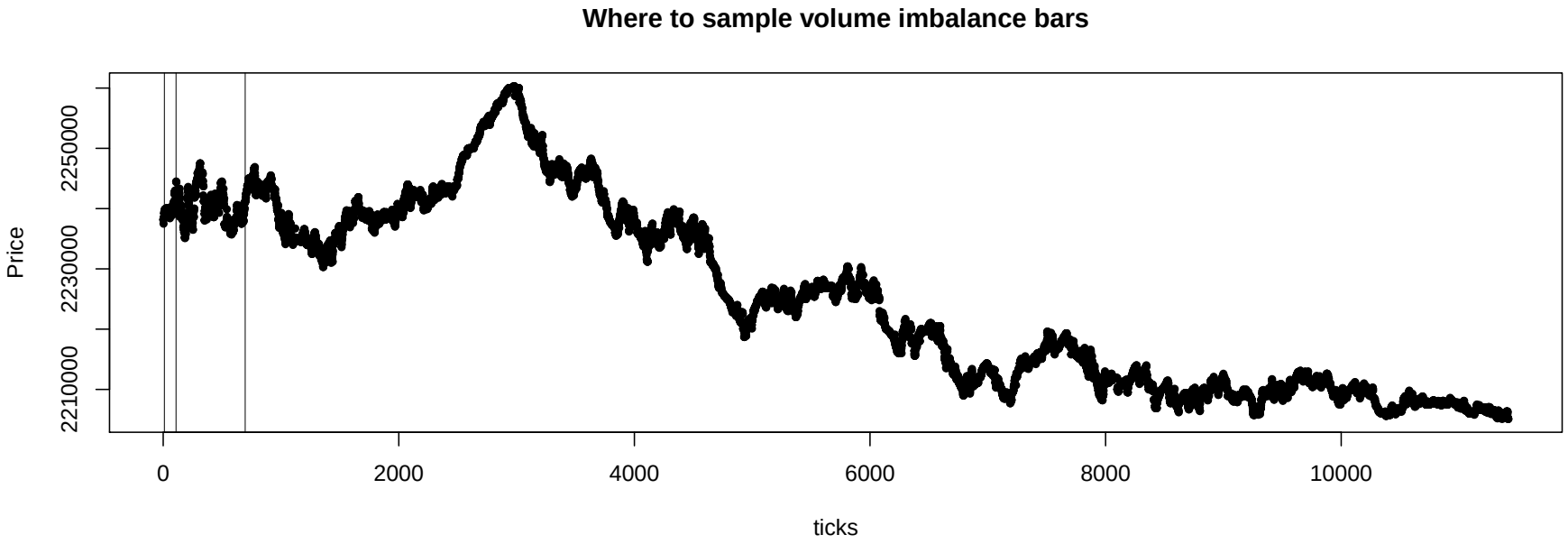


* volume imbalance bars

```r
#' The auxiliary function b_tv_t for constructing volume imbalance bars. The first b_tv_t is assigned the
#' value 0 because no information is available
#'
#' @param dat dat input with at least the following columns: Price, Size
#' @examples
#'
#' set.seed(1)
#' dat <- data.frame(Price = c(rep(0.5, 4), runif(10)), Size = rep(10,14))
#'
#' b_tv_t <- imbalance_volume(dat)
#'
#' @export
imbalance_volume <- function(dat)
{
```

```r
  n <- length(dat$Price)
  vol <- dat$Size  ## the main difference than imbalance_tick
  imbalance <- rep(0, n)
  price_diff <- diff(dat$Price)
  for(i in 2:n)
  {
    imbalance[i] <- sign(price_diff[i-1])*(price_diff[i-1]!=0)*vol[i] + imbalance[i-1]*(price_diff[i-1]==0)
  }
  imbalance
}

#' Tstar index for Volume Imbalance Bars (bar_vib)
#' @param dat dat input with at least the following columns: Price, Size
#' @param w0 the time window length of the first bar
#' @param bkw_T backward window length when using pracma::movavg for exponentially weighted average T
#' @param bkw_b backward window length when using pracma::movavg for exponentially weighted average b_tv_t
#'
#' @return a vector for the lengths of the tick imbalance bars. For example, if the return is c(10,26),
#' then the 2 tick imbalance bars are (0,10] and (10, 36]
#'
#' @examples
#'
#' set.seed(1)
#' dat <- data.frame(Price = c(rep(0.5, 4), runif(50)), Size = rep(10, 54))
#' T_vib <- Tstar_vib(dat)
#' b_tv_t <- imbalance_volume(dat)
#' cumsum(b_tv_t)[cumsum(T_vib)] # check the accumulated b_t's where the imbalances occur
#'
#' @export
Tstar_vib <- function(dat, w0=10, bkw_T=5, bkw_b=5)
{
  nx <- dim(dat)[1]
  b_tv_t <- imbalance_volume(dat) ## the main difference than imbalance_tick
  w0 <- max(min(which(cumsum(b_tv_t) != 0)), w0) # fix the case when there are always 0 at the beginning
  Tvec <- w0
  EOT <- Tvec
  repeat
  {
```

```
    T_last <- sum(Tvec) # the previous T that has been calculated
    nbt <- min(bkw_b, T_last - 1)
    PminusP <- pracma::movavg(b_tv_t[1:T_last], n=nbt, type="e") # the main difference than tib
    PminusP <- tail(PminusP,1) # the last one is what we need
    b_tv_t_Expected <- E0T*abs(PminusP)
    b_tv_t_psum <- abs(cumsum(b_tv_t[-(1:T_last)]))

    # if max(b_tv_t_psum) < b_tv_t_Expected, then there is no chance of volume imbalance
    if(max(b_tv_t_psum) < b_tv_t_Expected){break}else
    {
      T_new <- min(which(b_tv_t_psum >= b_tv_t_Expected))
    }
    T_last <- T_last + T_new
    if(T_last > nx){break}else
    {
      Tvec <- c(Tvec, T_new)
      nTvec <- length(Tvec)
      if(nTvec <= 2)
      {
        E0T <- mean(Tvec) # not enough T for exponential weighted average, so use the mean
      }else
      {
        nT <- min(bkw_T, length(Tvec)-1)
        E0T <- pracma::movavg(Tvec[1:nTvec], n=nT, type = "e")
        E0T <- tail(E0T,1)
      }
    }
  }
  return(Tvec)
}

T_vib <- Tstar_vib(dat_exc)
plot(dat_exc$Price, pch=20, xlab="ticks", ylab="Price", main="Where to sample volume imbalance bars")
abline(v=cumsum(T_vib), lwd=0.2)
```

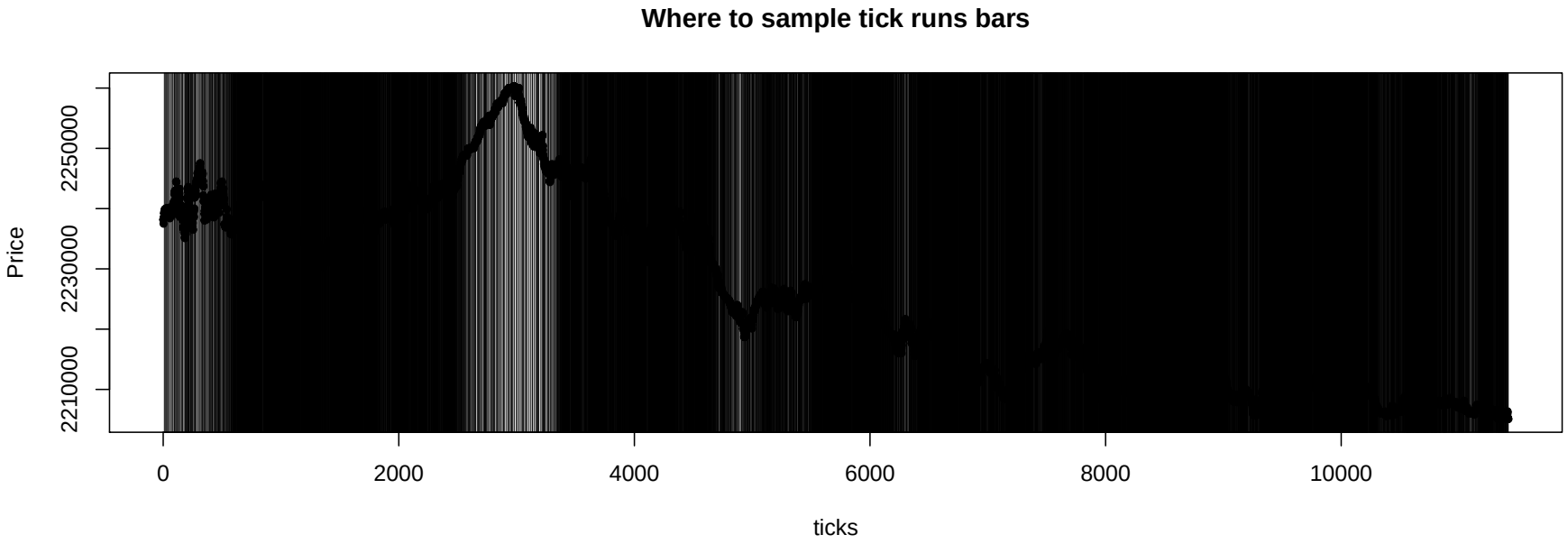


* tick runs bars

```
#' Tstar index for Tick Runs Bars (bar_trb)
#' @param dat dat input with at least the following columns: Price
#' @param w0 the time window length of the first bar
#' @param bkw_T backward window length when using pracma::movavg for exponentially weighted average T
#' @param bkw_b backward window length when using pracma::movavg for exponentially weighted average b_t
#'
#' @return a vector for the lengths of the tick imbalance bars. For example, if the return is c(10,26),
#' then the 2 tick imbalance bars are (0,10] and (10, 36]
#'
#' @examples
#'
#' set.seed(1)
#' dat <- data.frame(Price = c(rep(0.5, 4), runif(50)))
#' T_trb <- Tstar_trb(dat)
#' b_t <- imbalance_tick(dat)
#' cumsum(b_t)[cumsum(T_trb)] # check the accumulated b_t's where the imbalances occur
#'
#' @export
```

```r
Tstar_trb <- function(dat, w0=10, bkw_T=5, bkw_Pb1=5)
{
  b_t <- imbalance_tick(dat)
  nb <- length(b_t)
  nx <- dim(dat)[1]

  # calculate the length of the 1st run
  th_T <- sapply(1:nb, function(i){
    b_t_tmp <- b_t[1:i]
    if(sum(b_t_tmp %in% c(-1,1))==0){out <- 0}else
    {
      out <- max(sum(b_t_tmp[b_t_tmp==1]), -sum(b_t_tmp[b_t_tmp==-1]))
    }
    out
  })

  w0 <- max(min(which(th_T != 0)), w0) # fix the case when there are always 0 at the beginning
  w0 <- max(min(which(b_t==1)), w0) # there must be at least 1 b_t = 1 during the first window
  Tvec <- w0
  E0T <- T_last <- Tvec
  Pb1 <- sum(b_t[1:w0]==1) / w0 # Pb1: Pr[b_t = 1]
  Pb1vec <- Pb1
  th_T_Expected <- E0T*max(Pb1, 1-Pb1)
  while(T_last<nx-1)
  {
    T_last <- sum(Tvec) # the last T that has been calculated
    for(j in 1:(nb-T_last-1))
    {
      b_t_tmp <- b_t[(T_last+1):(T_last+j)]
      if(sum(b_t_tmp %in% c(-1,1))==0){th_T_tmp <- 0}else
      {
        th_T_tmp <- max(sum(b_t_tmp[b_t_tmp==1]), -sum(b_t_tmp[b_t_tmp==-1]))
      }
      if(th_T_tmp >= th_T_Expected)
      {
        new_flag <- TRUE # new window generated!
        T_new <- j
        Tvec <- c(Tvec, T_new)
```

```
        T_last <- T_last + T_new
        Pb1_new <- sum(b_t_tmp==1) / j
        Pb1vec <- c(Pb1vec, Pb1_new)
        break
      }
    }

    if(new_flag==TRUE)
    {
      new_flag <- FALSE
      nTvec <- length(Tvec) # nTvec should be the same as nPb1vec
      if(nTvec <= 2)
      {
        E0T <- mean(Tvec) # not enough T for exponential weighted average, so use the mean
        Pb1 <- mean(Pb1vec)
      }else
      {
        nT <- min(bkw_T, length(Tvec)-1)
        E0T <- pracma::movavg(Tvec[1:nTvec], n=nT, type = "e")
        E0T <- E0T[length(E0T)]
        nPb1 <- min(bkw_Pb1, length(Tvec)-1)
        Pb1 <- pracma::movavg(Pb1vec[1:nTvec], n=nPb1, type = "e")
        Pb1 <- Pb1[length(Pb1)]
      }
      th_T_Expected <- E0T*max(Pb1, 1-Pb1)
    }else{break}
  }
  return(Tvec)
}

T_trb <- Tstar_trb(dat_exc)
plot(dat_exc$Price, pch=20, xlab="ticks", ylab="Price", main="Where to sample tick runs bars")
abline(v=cumsum(T_trb), lwd=0.2)
```



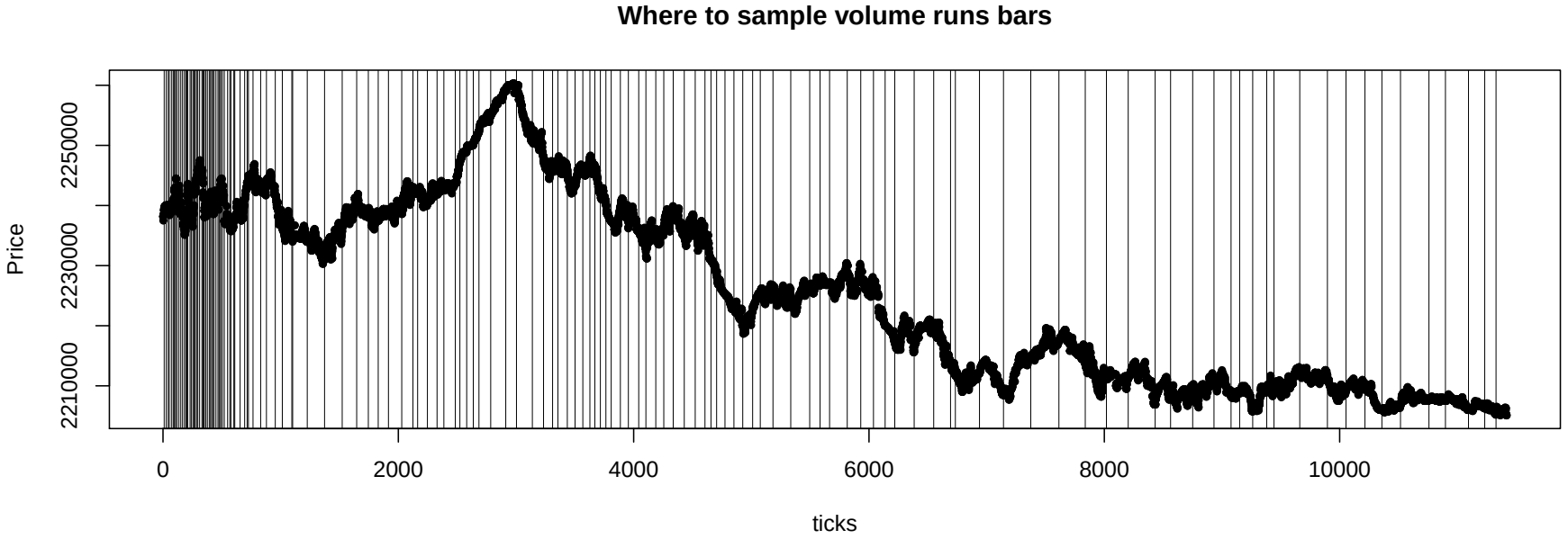
* volume runs bars

```
#' Tstar index for Volume Runs Bars (bar_vrb)
#' @param dat dat input with at least the following columns: Price, Size
#' @param w0 the time window length of the first bar
#' @param bkw_T backward window length when using pracma::movavg for exponentially weighted average T
#' @param bkw_Pb1: backward window length when using pracma::movavg for exponentially weighted average P[b_t=1]
#'
#' @return a vector for the lengths of the tick runs bars. For example, if the return is c(10,26), then
#' the 2 tick runs bars are (0,10] and (10, 36]
#'
#' @examples
#'
#' set.seed(1)
#' dat <- data.frame(Price = c(rep(0.5, 4), runif(50)), Size = rep(10,54))
#' T_vrb <- Tstar_vrb(dat)
#'
#' @export
Tstar_vrb <- function(dat, w0=10, bkw_T=5, bkw_Pb1=5)
{
```

```r
b_t <- imbalance_tick(dat)
nb <- length(b_t)
nx <- dim(dat)[1]
vol <- dat$Size[1:nb]

# calculate the length of the 1st run
th_T <- sapply(1:nb, function(i){
  b_t_tmp <- b_t[1:i]
  vol_t_tmp <- vol[1:i]
  if(sum(b_t_tmp %in% c(-1,1))==0){out <- 0}else
  {
    out <- max( sum( b_t_tmp[b_t_tmp==1]*vol_t_tmp[b_t_tmp==1] ),
                -sum( b_t_tmp[b_t_tmp==-1]*vol_t_tmp[b_t_tmp==-1] ) )
  }
  out
})

w0 <- max(min(which(th_T != 0)), w0) # fix the case when there are always 0 at the beginning
w0 <- max(min(which(b_t==1)), w0) # there must be at least 1 b_t = 1 during the first window
Tvec <- w0
E0T <- T_last <- Tvec
Pb1 <- sum(b_t[1:w0]==1) / w0 # Pb1: Pr[b_t = 1]
Pb1vec <- Pb1

# the main difference than the tick runs bar
vol_tmp <- vol[1:w0]
volvec_up <- vol_tmp_up <- mean(vol_tmp[b_t[1:w0] == 1]) # the first expected v_t|b_t=1
volvec_down <- vol_tmp_down <- mean(vol_tmp[b_t[1:w0] == -1]) # the first expected v_t|b_t=-1
th_T_Expected <- E0T*max(Pb1*vol_tmp_up, (1-Pb1)*vol_tmp_down)

while(T_last<nx-1)
{
  T_last <- sum(Tvec) # the last T that has been calculated
  for(j in 1:(nb-T_last-1))
  {
    b_t_tmp <- b_t[(T_last+1):(T_last+j)]
    vol_tmp <- vol[(T_last+1):(T_last+j)]
    if(sum(b_t_tmp %in% c(-1,1))==0){th_T_tmp <- 0}else
```

```
    {
      th_T_tmp <- max( sum( b_t_tmp[b_t_tmp==1]*vol_tmp[b_t_tmp==1] ),
                       -sum( b_t_tmp[b_t_tmp==-1]*vol_tmp[b_t_tmp==-1] ) )
    }
    if(th_T_tmp >= th_T_Expected)
    {
      new_flag <- TRUE # new window generated!
      T_new <- j
      Tvec <- c(Tvec, T_new)
      T_last <- T_last + T_new

      Pb1_new <- sum(b_t_tmp==1) / j
      volup_new <- ifelse(sum(b_t_tmp == 1)>0, mean(vol_tmp[b_t_tmp == 1]),0)
      voldown_new <- ifelse(sum(b_t_tmp == -1)>0, mean(vol_tmp[b_t_tmp == -1]),0)

      Pb1vec <- c(Pb1vec, Pb1_new)
      volvec_up <- c(volvec_up, volup_new)
      volvec_down <- c(volvec_down, voldown_new)

      break # whenever new window is generated, we should stop the for loop and then deal with the new window
    }
  }

  if(new_flag==TRUE)
  {
    new_flag <- FALSE
    nTvec <- length(Tvec) # nTvec should be the same as nPb1vec
    if(nTvec <= 2)
    {
      E0T <- mean(Tvec) # not enough T for exponential weighted average, so use the mean
      Pb1 <- mean(Pb1vec)
      E0V_up <- mean(volvec_up)
      E0V_down <- mean(volvec_down)
    }else
    {
      nT <- min(bkw_T, length(Tvec)-1)
      E0T <- pracma::movavg(Tvec[1:nTvec], n=nT, type = "e")
      E0T <- E0T[length(E0T)] # equivalent to tail(E0T,1)
```

```
        nPb1 <- min(bkw_Pb1, length(Tvec)-1)
        Pb1 <- pracma::movavg(Pb1vec[1:nTvec], n=nPb1, type = "e")
        Pb1 <- Pb1[length(Pb1)]
        E0V_up <- pracma::movavg(volvec_up[1:nTvec], n=nPb1, type = "e") # use the same nPb1 for E0V_up
        E0V_up <- E0V_up[length(E0V_up)]
        E0V_down <- pracma::movavg(volvec_down[1:nTvec], n=nPb1, type = "e") # use the same nPb1 for E0V_down
        E0V_down <- E0V_down[length(E0V_down)]
      }
      th_T_Expected <- E0T*max(Pb1*E0V_up, (1-Pb1)*E0V_down)
    }else{break}
  }
  return(Tvec)
}

T_vrb <- Tstar_vrb(dat_exc)
plot(dat_exc$Price, pch=20, xlab="ticks", ylab="Price", main="Where to sample volume runs bars")
abline(v=cumsum(T_vrb), lwd=0.2)
```



- Some comments about the comparisons:
  - Generally speaking, various imbalance bars share a similar pattern, and various runs bars share a similar pattern.

- Because imbalance bars try to capture imbalance between buy and sell orders, it can seldom sample bars during a short time period such as the sample data. That means such a market during the time period is quite balanced, which is very common for developed market where most participants are institutional investors and there are also liquidity providers that create sufficient both buy and sell orders to keep a liquid market. Such an imbalance might only be observed when there are a large amount of one-direction orders during a given time period.
- Either tick runs bars or volume runs bars (and actually dollar runs bars) can lead to a lot more bars than imbalance bars, and these methods would be more appropriate under certain situations for example when one does not have the dataset with a very long history, or one wants to involve more data points for conducting inference. However, a larger sample size might not be always better because on the one hand it may bring more less informative data points, and on the other hand, some machine learning methods do not scale well with large data set. In this regard, a features filer such as the CUSUM filter used on such runs bars can be useful when one needs to further decrease the sample size while keeping the most relevant information.