UNIVERSITY OF CAPE TOWN
IYUNIVESITHI YASEKAPA • UNIVERSITEIT VAN KAAPSTAD

Data Analysis for High-Frequency Trading

Assignment 1

TAQ Data workflow and EDA

Julian Albert
ALBJUL005

Department of Statistical Sciences
University of Cape Town
South Africa
August 7, 2019

Contents

Introduction

In this assignment I am tasked with creating a trade and quote (TAQ) data workflow and performing some basic exploratory data analysis (EDA). TAQ data contains intraday transactions data (trades and quotes) providing insight into a securities supply and demand that drives price, an essential part of high-frequency data analysis. The data is pulled from Bloomberg [1] such that the trades are in one .csv and the quotes are split into bid and ask with each month being a seperate .csv, as such the data workflow will pull in the months of quotes and aggregate them with the trades after which the data will be processed into a user friendly form. The data workflow has been optimised using data.tables in R as they are more memory efficient and ultimately faster than data.frames. The runtime to pull in the 5 months of TAQ data, clean, classify and compact for both AGL and NPN is approximately 2 minutes.

Data

The TAQ data pulled from [1] consists of trades and quotes for the period 2019-01-17 until 2019-06-19, each set (trades, bids and asks) contains the date, price of event and volume of the event. The bid and ask data are pulled separately and merged to keep all the data, where there is no bid (or ask) data on a specified day the data is filled with the previous non-missing values. The trades consist of larger trades executed against collections of smaller trades, this sequence is still a single trade of a given volume and as such must be *compacted* to reflect it as such. In the data we see two event types (quote and trade), where the data displays same time quotes - sequences of quotes with the same date-time stamps - the quote most recent in sequence quote is retained and the remainder removed. For same time trades - sequences of trades with the same date-time stamps - as well as the same order types (buyer or seller initiated) the sequence of trades is aggregated into a single trade with a single date-time stamp but aggregated volume and the trade price is set to be the volume weighted average price (VWAP). The assumption here is that orders of different sign +1(-1) are not aggregated. To identify order type we must *classify* the trades using the methodology, as originally developed by [3], involving a two-step approach defined by a quote test followed by a tick test

1. In the Quote Test, if the price of a trade is lower than the midpoint of the matched bid and ask, then the trade is classified as seller-initiated. If the price is higher, the trade is considered to be buyer-initiated.
2. When they are equal, the tick test is utilized. In the Tick Test, if the price is lower than the previous price, it is classified as seller-initiated and if it is higher, then the trade is considered to be buyer-initiated.

If the tick test results are not conclusive the trades will be classified as undetermined.

In order to carry out basic EDA I must first prepare the dataset to include some useful measures. The complete dataset must be time ordered and include: date, time, mid-price, mid-price change, volume, normalised volume, bid, bid volume, ask, ask volume, event type, transaction price, transaction volume and volume-weighted average prices (VWAP). To calculate some of these measures we can use the definitions in Table 1. The mid-price represents the mid-point of the current bid-ask spread quoted for a security and acts as a proxy for the trade price and the mid-price change is the change in quote due to a transaction event (take to be log differences to preserve ergodicity). Where the mid-price may be unrealistic when the volume of limit orders

at the best bid and ask prices differ significantly the micro-price is considered as it represents a volume wighted mid-price.

Measure	Formulae
Mid-price (m_{ij})	$m_{ij} = \frac{1}{2}(b_{ij} + a_{ij})$
Mid-price Change (Δm_{ij})	$\Delta m_{ij} = \log\left(\frac{m_{i+1,j}}{m_{i,j}}\right)$
Micro-price (S_t)	$S_t = \left(\frac{v^a_{ij}}{v^a_{ij} + v^b_{ij}}\right) a_{ij} + \left(\frac{v^b_{ij}}{v^a_{ij} + v^b_{ij}}\right) b_{ij}$
Inter-arrival Times (t_{ij})	$t_{ij} = t_{i+1,j} - t_{ij}$
Normalised Trade Volume (ω_{ij})	$\omega_{ij} = \frac{v_{ij}}{\sum_{k=1}^{T_j} v_{kj}} \left(\frac{\sum_{j=1}^{N} T_j}{N}\right)$
VWAP ($vwap_{t+1,t}$)	$vwap_{t+1,t} = \frac{\sum_{i=t}^{t+1} v_i p_i}{\sum_{i=t}^{t+1} v_i}$

Table 1: Useful Definitions for Dataset Measures.

Initial EDA

With the prepared dataset this section will visualise an hour of trading on the two different stocks between 10h00 and 11h00, and 16h00 and 17h00. This visualisation is achieved by plotting the micro-price (in green) in calendar time, the bid and (offer) (best-bid and best-offer) order book events as blue (and red) bubbles that are proportional to the volume on bid (and on offer) and transactions in yellow at the transaction event-times as bubbles proportional to the transaction volumes. For the visualisation a random trading day is selected (a day where a trade certainly occurred) and the a data subset is taken for a specified trading interval, the resulting plots can be seen in Figure 1.

Starting with AGL we can visualise the two time intervals, starting at 10:00am we see the hourly high proceeded by big bid volumes driving the mid-price (and ultimately the transaction and micro-price) downwards, this is met with large transaction volumes between 10:15 and 10:20 from which point on the data seems more stable. Starting at 16:00pm a week later we see the price is higher and rallying up, the price increase is not evidently due to an excess of buyers or sellers driving the price. Curiously at approximately 16:31 we see a large transaction volume (possibly at the best-ask at the time but given the transaction bubble clearly covers any of the best ask bubble) we may determine this to be an error in the data-feed. This data point obscures the visual analysis of this graph.

Moving on to NPN at 10:00am we see a fairly consistent rise and fall over the hour with high asks at around 10:37 and a big transaction at approximately 10:56am. More interesting is NPN a month later, there is a big increase in the price. During the interval from 16:00 until 16:49:59 we see a large bid-ask spread with the ask volumes being distinctly larger than those of the bids

(there are more sellers than buyers). The bid-ask spread may be much wider due to increased volatility prompting market makers to profit from the spread. When securities are increasing in value, investors are willing to pay more giving market makers the opportunity to charge higher premiums. When volatility is low, and uncertainty and risk are at a minimum, the bid-ask spread is narrow.

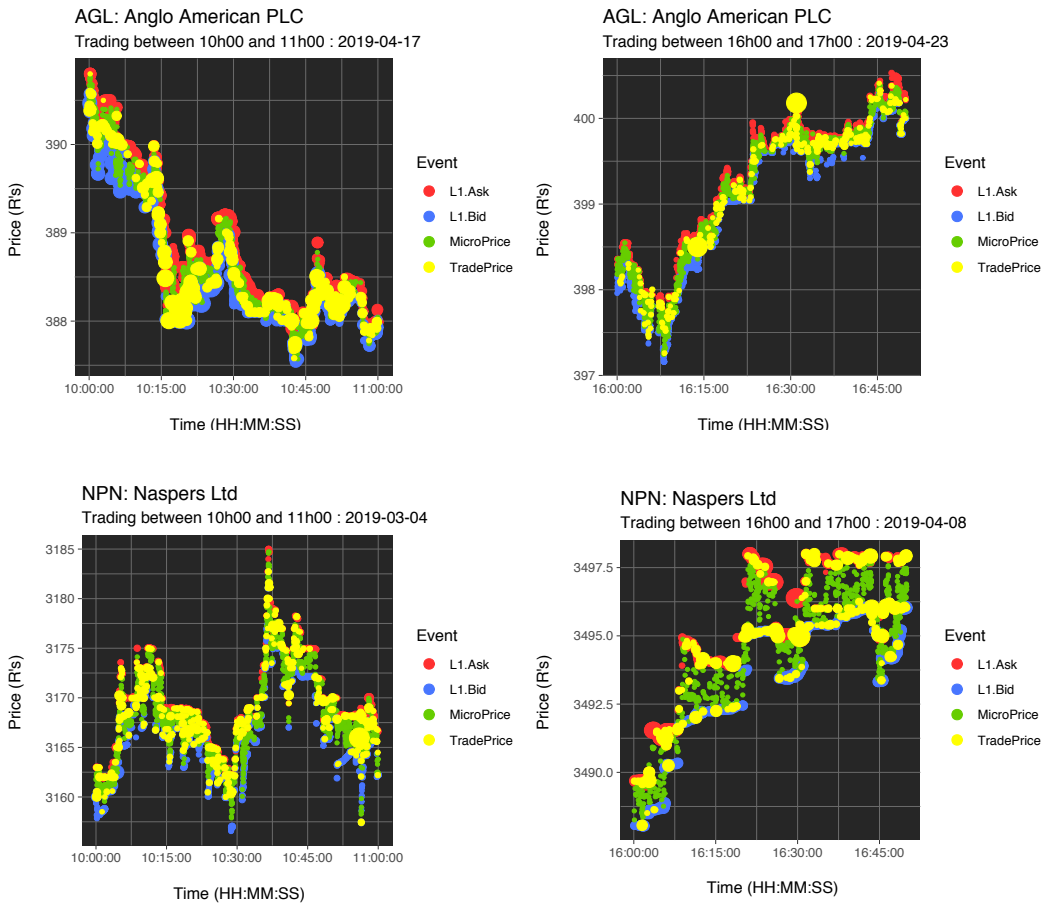

Figure 1: EDA of 1-hour TAQ data.

Order-Flow Auto-Correlation

Order flow can be defined as the the sequence/process which assumes value +1 for buyer initiated trades and -1 for seller initiated trades. Order flow is persistent in the sense that orders to buy tend to be followed by more orders to buy and orders to sell tend to be followed by more orders to sell [4]. Figure 2 illustrates that the autocorrelation function for market order signs decays very slowly and that the autocorrelation function is roughly linear in a double logarithmic scale suggesting that a power law relation might be a reasonable description

for the sample autocorrelation function [2].

Using the ACF() function in R, the auto-correlations for both the trade signs and inter-arrival times are calculated and plotted in Figure 2 and 6 respectively. The auto-correlations are plotted on the log-log scale, as a result the transformation sometimes sets data to null (hence the discontinuities).

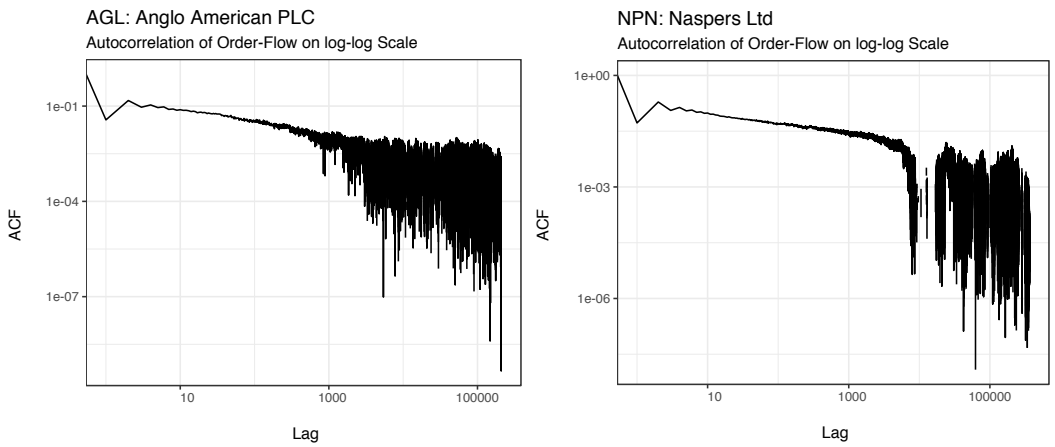

Figure 2: Auto-Correlation of Order Flow for Two Equities.

Autocorrelation is an important feature to understand, if a time series is autocorrelated we can anticipate future states of the series, thus if there were auto-correlations for the trade-signs, high-frequency traders could predict whether subsequent traders will be seller or buyer initiated based on the most recent trade signs.

Inter-arrival Times

Trade inter-arrival times Δt_{ij} represent the time between trades and provide useful insight into the order flow and liquidity of a security. In this section I will consider the frequency of the inter-arrival times, a quantile-quantile analysis and the auto-correlations.

Frequency

We may be interested to know the distribution of inter-arrival times Δt_{ij} as this will give us a sense of how "fast" the security is trading. In Figure 3 we plot the histogram of inter-arrival times over the entire set of trading events for the 135 trading days. We can see that the amount of trading events are higher for Naspers Ltd compared to AGL PLC. In both cases we observe a right skew distribution indicating that the trades occur more frequently together (a sign of higher liquidity), the left column is on the absilute scale where the skew is heavy thus we apply a log scale to get a better sense of the behaviour of inter-arrival times. We can see a subtle difference in that Naspers Ltd is more right skewed indicating more liquidity.

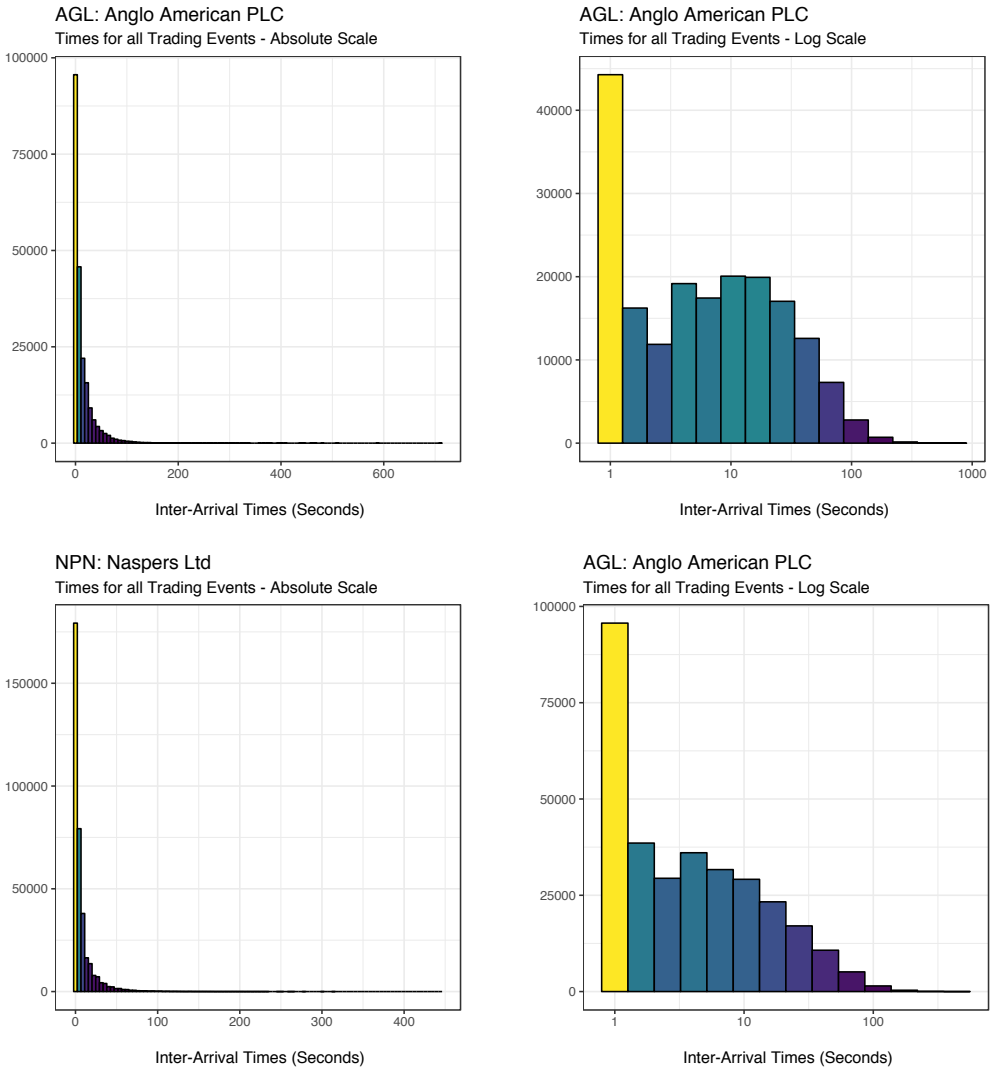

Figure 3: Frequency of Inter-Arrival Times in Absolute and Log Scales

To further analyse the trade frequency we can plot the frequency of inter-arrival times explicitly less than 10 seconds, 1 minute and 10 minutes. The resulting plots are seen in Figure 4 where clearly the bulk of the mass is captured in 1 second whilst we can determine that the inter-arrival times are majority less than 1 minute for both securities indicating that they are both relatively liquid. Interestingly the high frequency of inter-arrival times around 2 seconds fits well with the exponential distribution up to 2 seconds (see the next section and Figure 5).

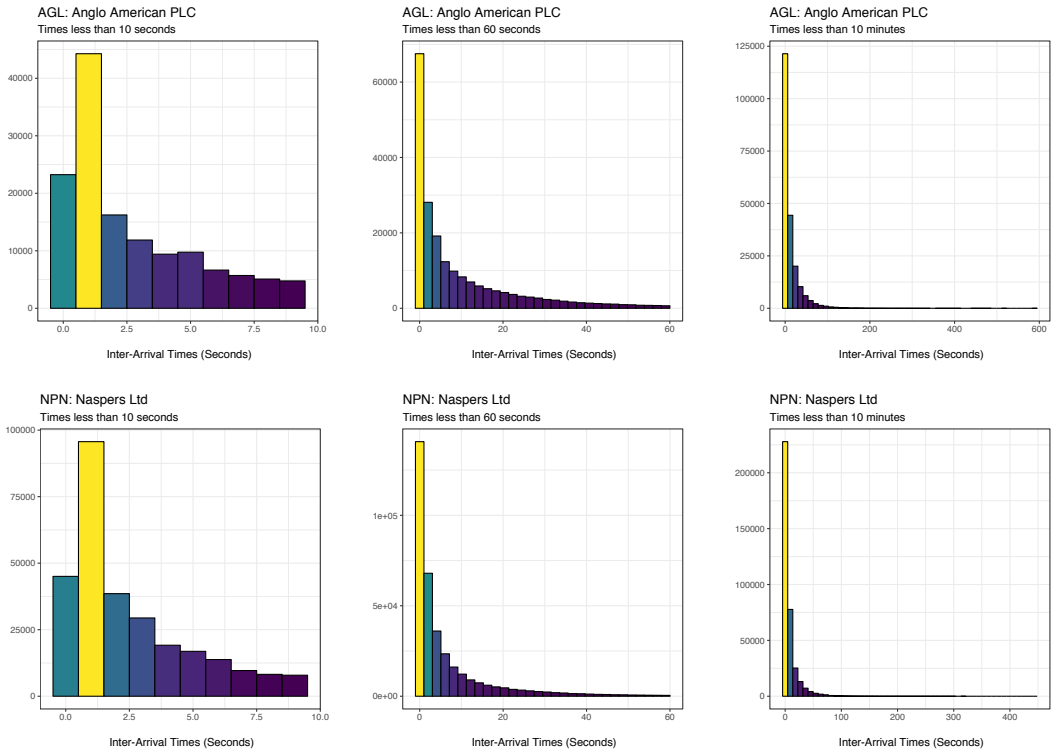

Figure 4: Frequency of Inter-Arrival Times less than 10 seconds, 1 minute and 10 minutes.

QQ Plot

The shape of the frequency for inter-arrival times in Figure 3 and 4 appear Exponential in form. To check this we can consider the quantile-quantile plot (QQ Plot) of the empirical quantiles obtained from the inter-arrival times against the theoretical quantiles of the Exponential distribution. The QQ plot is illustrated in Figure 5 from which we can see that the inter-arrival times of trades are not Exponentially distributed (as a good fit is one in which the points lie along the 45 degree reference line through the origin). For inter-arrival times less than approximately 2.5 seconds the Exponential fits relatively well, however, for trade inter-arrivals greater than 2.5 seconds the empirical quantiles are larger than the empirical quantiles indicating that the data has a fatter right tail than the Exponential distribution. This phenomenon of leptokurtosis is a known real world problem of financial data, the fat tail suggests that inter-arrival times of trades decay slower than the Exponential distribution.

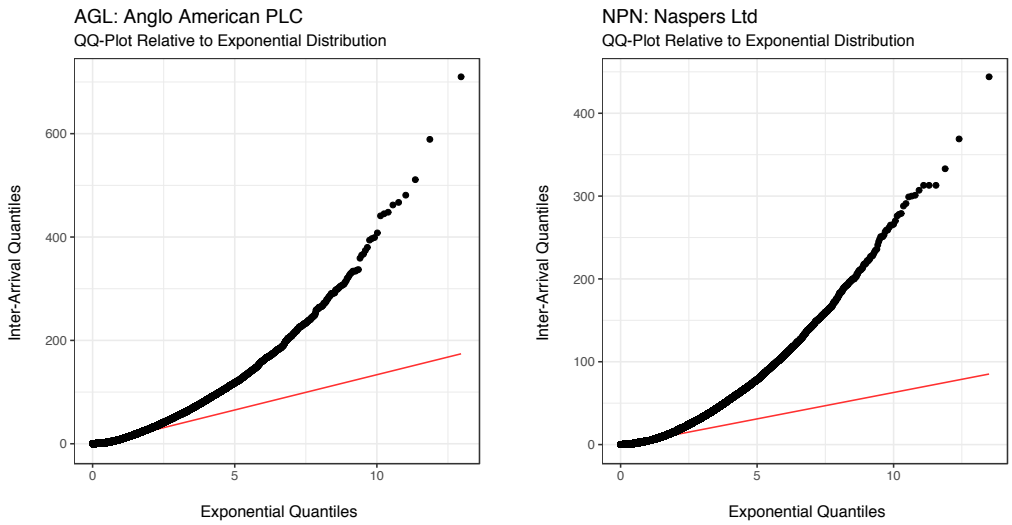

Figure 5: QQ Plots of Inter-Arrival Times Relative to Exponential Distribution.

ACF

If the inter-arrival times were autocorrelated, then we could anticipate the time of the next trade using recent historical inter-arrival times. We can see that the auto-correlations in the inter-arrival times decay slowly. This result indicates that if one observes a market order now, based on this information alone there is some nonvanishing predictability of the market order arrival times [2].

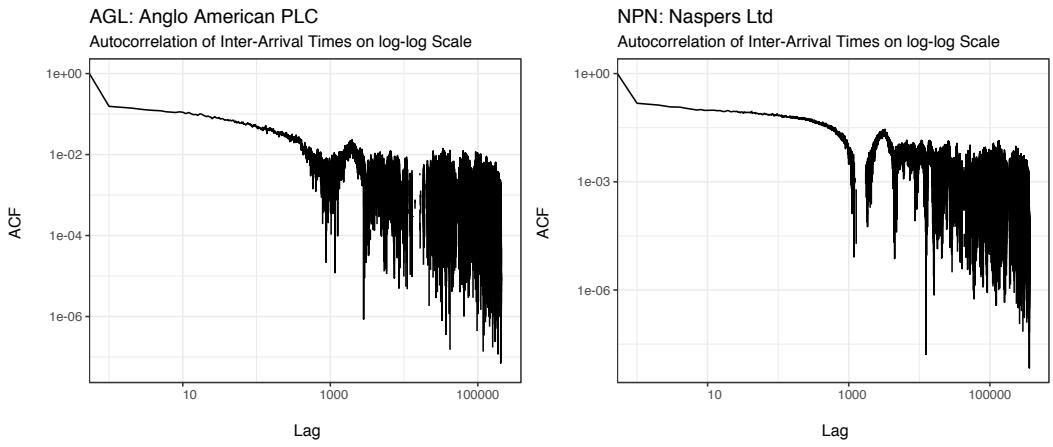

Figure 6: Auto-Correlation of Inter-Arrival Times.

- END -

References

[1] Bloomberg Database. *Bloomberg Professional*. Bloomberg Subscription Service, Online, 2019.

[2] Bouchaud, J.-P., Farmer, J. D., and Lillo, F. How markets slowly digest changes in supply and demand. In *Handbook of financial markets: dynamics and evolution*. Elsevier, 2009, pp. 57–160.

[3] Lee, C. M., and Ready, M. J. Inferring trade direction from intraday data. *The Journal of Finance 46*, 2 (1991), 733–746.

[4] Toth, B., Palit, I., Lillo, F., and Farmer, J. D. Why is equity order flow so persistent? *Journal of Economic Dynamics and Control 51* (2015), 218–239.

Appendix

R Code

Listing 1: Data Pull From Bloomberg

```
##
#
# Author: Julian Albert
# Date: 01 August 2019
#
# Description:
# HFT Data Pull from Bloomberg
#
#--------------------------------------------------------------------------------#


# 0. Clean Workspace, Directory and Library ----

## Clean Workspace
rm(list=ls())
dev.off() # Close Plots

## Packages for Bloomberg PC
# Some might/might not install
install.packages("Rblpapi")
install.packages("lubridate")
library(Rblpapi)
library(lubridate)

## Bloomberg Connection
con <- blpConnect()

## Directory
dir <- "G:/HFT/Data/"

# 1. Variables for Pulling ----

# security <- c("APN SJ EQUITY", "CPI SJ EQUITY", "BIL SJ EQUITY","INP SJ EQUITY",
#               "AGL SJ EQUITY", "TFG SJ EQUITY", "BTI SJ EQUITY", "MTN SJ EQUITY",
#               "NPN SJ EQUITY","SHP SJ EQUITY")

## Securities to pull
security <- c("AGL SJ EQUITY", "NPN SJ EQUITY")
n.securities <- length(security)
## Timezones
timezone <- Sys.timezone()
StartTime <- as.POSIXct("2019-01-01 08:00:00", tz = timezone)
CurrentTime <- Sys.time()
n.months <- interval(StartTime, test1) %/% months(1)

# 2. Pulling data ----

## Daily Sampled OHLCV
barInt <- 540

daily_sampled <- list()
for (i in 1:n.months){
  daily_sampled[[i]] <- getBars(security = security[i], barInterval = 540,
                                startTime = StartTime, tz = timezone, con = con)
  filen <- paste(dir, "Daily/", substr(security[i], 1, 3), ".csv", sep = "")
  write.csv(daily_sampled[[i]], filen, row.names = F)
}

getBars(security = security[1], barInterval = 540,
        startTime = StartTime, tz = timezone, con = con)

## Intraday
StartTime <- as.POSIXct("2019-01-01 08:00:00", tz = timezone)

### Trade data
```

```
for (i in 1:n.months){
  filen_trades = paste(dir, "Intraday/trade_",
                       substr(security[i], 1, 3), ".csv", sep = "")
  temp = getTicks(security = security[i], eventType = c("TRADE"),
                  startTime = StartTime, tz = timezone, con = con)
  write.csv(temp, filen_trades, row.names = FALSE)
}

### Ask data
for (i in 1:n.securities){
  StartTime = StartTime
  EndTime = StartTime
  month(EndTime) <- month(StartTime) + 1
  for (k in 1:6){
    filen_ask <- paste(dir,"Intraday/asks/ask_",
                       substr(security[i], 1, 3), "00", k, ".csv", sep = "")
    tmp_filen <- getTicks(security = security[i], eventType = c("BEST_ASK"),
                      startTime = StartTime, endTime = EndTime,
                      tz = timezone, con = con)
    write.csv(tmp_filen, filen_ask, row.names = FALSE)
    StartTime <- EndTime
    month(EndTime) <- month(EndTime) + 1
  }
}

### Bid data
for (i in 1:n.securities){
  StartTime <- StartTime
  EndTime <- StartTime
  month(EndTime) <- month(StartTime) + 1
  for (k in 1:n.months){
    filen_bid <- paste(dir, "Intraday/bids/bid_",
                       substr(security[i], 1, 3), "00", k, ".csv", sep = "")
    tmp_filen = getTicks(security = security[i], eventType = c("BEST_BID"),
                         startTime = StartTime, endTime = EndTime,
                         tz = timezone, con = con)
    write.csv(tmp_filen, filen_bid, row.names = FALSE)
    StartTime <- EndTime
    month(EndTime) <- month(EndTime) + 1
  }
}

#------------------------------------------------------------------------------#
```

Listing 2: Data Workflow - Clean, Classify and Compact

```
##
#
# Author: Julian Albert
# Date: 01 August 2019
#
# Description:
# HFT Assignment 1 - Data Load and Clean
#
#------------------------------------------------------------------------------#

# 0. Clean Workspace, Directory and Library ----

## Clean Workspace
rm(list=ls())
dev.off() # Close Plots
setwd("~") # Clear Path to User

## Locations
project_folder <- "/Documents/UCT/Coursework/HFT"
loc_script <- "/Assignment_1/UCT_Assignment/Code"
loc_data <- "/Data"

## Directories
dir_script <- paste("~", project_folder, loc_script, sep = '')
dir_data <- paste("~", project_folder, loc_data, sep = '')

## Set Working Directory to Script Location
```

```r
setwd(dir_script)

## Libraries - Lazyload
if (!require("pacman")) install.packages("pacman")
p_load(tidyverse, openxlsx, lubridate, zoo, data.table)

## Options
options(digits.secs = 3)

# 1. Load in data ----

## Variables
securities <- c("AGL", "NPN")
n.securities <- length(securities)
n.months <- 5
timezone <- "Africa/Johannesburg"
dat_TAQ_final <- list()

# 2. Cleaning Everything ----

## Start Time
ptm <- proc.time()

## For eeach security
for(security in 1:length(securities))
{

  # Quotes are in multiple .csv by month >> read into list
  dat.quotes_CLEAN <- list()

  # Per month into one list
  for(month in 1:n.months){

    ## Filenames for Bid and Ask
    filen_ask <- paste(dir_data, "/ask_",
                       substr(securities[security], 1, 3),"00", month, ".csv",
                       sep = "")

    filen_bid <- paste(dir_data, "/bid_",
                       substr(securities[security], 1, 3),"00", month, ".csv",
                       sep = "")

    ## Read Data for Bid and Ask
    tmp.dat_ask <- fread(filen_ask, drop = "condcode")
    tmp.dat_bid <- fread(filen_bid, drop = "condcode")

    ## Join the Data and put into Desired Format >> X = Ask, Y = Bid
    tmp.dat_quotes <- full_join(tmp.dat_ask, tmp.dat_bid, by = "times") %>%
      rename(DateTimeL = times,
             L1.Ask = value.x, Volume.Ask = size.x,
             L1.Bid = value.y, Volume.Bid = size.y) %>%
      arrange(DateTimeL) %>%
      mutate(Type = "Quote", type.x = NULL, type.y = NULL) %>%
      setDT()

    ## Clean the quotes >> Only take Last Quote >> Do now to reduce size ASAP
    dat.quotes_CLEAN[[month]] <- tmp.dat_quotes[, .SD[.N], DateTimeL]

  }

  ## Remove the Auction >> Set DateTime Format >> Fill NA Asks/Bids
  df_Quotes <- dat.quotes_CLEAN %>%
    bind_rows() %>%
    .[,  DateTimeL := parse_date_time(DateTimeL, "Ymd HMS", tz = timezone)] %>%
    setDF() %>%
    mutate(hms = format(as.POSIXct(DateTimeL), "%H:%M:%S")) %>%
    .[.$hms >= "09:10:00" & .$hms <= "16:49:59", ] %>%
    mutate(hms = NULL, Price = NA, Volume.Trade = NA) %>%
    fill(L1.Ask, Volume.Ask, L1.Bid, Volume.Bid) %>%
    as_tibble()

  ## Read in the trades (only one .csv with all trades)
  filen_trades <- paste(dir_data, "/trade_",
```

```
                        securities[security], ".csv",
                        sep = "")

  ## Remove the Auction
  tmp.dat_trades <- fread(filen_trades, drop = c("condcode", "type")) %>%
    rename(DateTimeL = times, Price.Trade = value, Volume.Trade = size) %>%
    .[,  DateTimeL := parse_date_time(DateTimeL, "Ymd HMS", tz = timezone)] %>%
    mutate(Type = "Trade", hms = format(as.POSIXct(DateTimeL), "%H:%M:%S")) %>%
    .[.$hms >= "09:10:00" & .$hms <= "16:49:59", ] %>%
    mutate(hms = NULL) %>%
    setDT()

  ## CLEAN TRADES
  df_Trades <- tibble(DateTimeL = tmp.dat_trades$DateTimeL,
                      L1.Ask = NA, Volume.Ask = NA,
                      L1.Bid = NA, Volume.Bid = NA,
                      Type = tmp.dat_trades$Type,
                      Price = tmp.dat_trades$Price,
                      Volume.Trade = tmp.dat_trades$Volume.Trade)

  ## TAQ data is Trades and Quotes
  df_TAQ <- bind_rows(df_Quotes, df_Trades) %>%
    arrange(DateTimeL) %>%
    setDT() %>%
    .[, MidPrice := 0.5*(L1.Ask + L1.Bid)] %>%
    fill(MidPrice) %>%
    mutate(QuoteRule = ifelse(Price < MidPrice, -1,
                              ifelse(Price > MidPrice, 1, 0)),
           tmpdiff = c(NA, diff(Price)),
           TickRule = ifelse(tmpdiff > 0, 1,
                             ifelse(tmpdiff < 0, -1, 0)),
           Trade.Sign = ifelse(!is.na(QuoteRule) & QuoteRule != 0, QuoteRule,
                               ifelse(QuoteRule == 0, TickRule, NA)))

  df_TAQ_CLEAN <- df_TAQ %>%
    mutate(QuoteRule = NULL, tmpdiff = NULL,  TickRule = NULL)

  ## Calculate measures on the Trades
  dat_transactions <- df_TAQ_CLEAN %>% filter(Type == "Trade") %>% setDT()

  dat_transactions <- dat_transactions %>%
    .[, keyby = .(DateTimeL, Trade.Sign),
      c("VWAP", "TotVolume") :=
        .(weighted.mean(Price, Volume.Trade),
          sum(Volume.Trade))] %>% # Calc VWAP and sum(Volume)/Trade Type/day
    .[, keyby = .(DateTimeL, Trade.Sign), .SD[.N]] %>%
    .[, by = floor_date(DateTimeL, "day"),
      c("DailyPropVol", "No.DailyTrades", "day", "Inter.Arrival.times") :=
        .(TotVolume/sum(TotVolume), .N,
          floor_date(DateTimeL, "day"),
          c(NA, diff(DateTimeL)))] %>% # Calc. Total Volume on Day, no Trades, t
    .[, c("Price", "Volume.Trade", "VWAP", "TotVolume",
          "TotalTradingDays") :=
        .(VWAP, TotVolume, NULL, NULL,uniqueN(day))] %>% # Clean and get N
    .[, NormalisedVolume :=
        DailyPropVol*(No.DailyTrades/TotalTradingDays)] %>% # Normalised Volume
    .[, c("day", "MidPrice", "Mid.Price.Change", "MicroPrice") :=
        .(NULL, NA, NA, NA)] # Clean

  ## Calculate measures on the Quotes
  dat_quotes <- df_TAQ_CLEAN %>% filter(Type == "Quote")%>% setDT()

  dat_quotes <- dat_quotes %>%
    .[, .SD[.N], by = DateTimeL] %>%
    .[, c("DailyPropVol", "No.DailyTrades", "TotalTradingDays",
          "NormalisedVolume", "Inter.Arrival.times", "MidPrice",
          "Mid.Price.Change", "MicroPrice") :=
        .(NA, NA, NA, NA, NA, 0.5*(L1.Bid + L1.Ask), c(NA, diff(log(MidPrice))),
          (Volume.Ask/(Volume.Ask+Volume.Bid))*L1.Ask +
            (Volume.Bid/(Volume.Ask+Volume.Bid))*L1.Bid)]

  ## Reorder into a Nice Format
  tmp_order_names <- c("DateTimeL", "Type", "L1.Ask", "L1.Bid", "MidPrice",
```

```
                        "Price","Trade.Sign", "Volume.Ask", "Volume.Bid",
                        "Volume.Trade", "NormalisedVolume", "MicroPrice",
                        "No.DailyTrades", "TotalTradingDays", "DailyPropVol",
                        "Inter.Arrival.times","Mid.Price.Change")

  setcolorder(dat_transactions, tmp_order_names)
  setcolorder(dat_quotes, tmp_order_names)

  ## Some Reason dates are mimatched, start from min(date) end at min(lastdate)
  first_date <- min(first(dat_quotes$DateTimeL),
                    first(dat_transactions$DateTimeL))

  last_date <- min(last(dat_quotes$DateTimeL),
                   last(dat_transactions$DateTimeL))

## Clean and Compact Data >> Store Per Ticker
dat_TAQ_final[[security]] <- bind_rows(dat_transactions, dat_quotes) %>%
  arrange(DateTimeL) %>%
  setDT() %>% # data with everything needed
  .[DateTimeL %between% c(first_date, last_date)] %>%
  as_tibble()

}

names(dat_TAQ_final) <- securities
proc.time() - ptm


# 3. Save Cleaned, Classified, Compacted Dataset ----
saveRDS(dat_TAQ_final, file = "dat_TAQ.rds")

#------------------------------------------------------------------------------#
```

Listing 3: EDA with Processed Data

```
##
#
# Author: Julian Albert
# Date: 02 August 2019
#
# Description:
# HFT Assignment 1 - Basically want to work with TAQ data and perform EDA to
# better understand the data and the trading environment.
#
#------------------------------------------------------------------------------#

# 0. Clean Workspace, Directory and Library ----

## Clean Workspace
rm(list=ls())
# dev.off() # Close Plots
setwd("~") # Clear Path to User

## Locations
project_folder <- "/Documents/UCT/Coursework/HFT"
loc_script <- "/Assignment_1/UCT_Assignment/Code"
loc_figs <- "/Assignment_1/UCT_Assignment/Figs"

## Directories
dir_script <- paste("~", project_folder, loc_script, sep = '')
dir_figs <- paste("~", project_folder, loc_figs, sep = '')

## Set Working Directory to Script Location
setwd(dir_script)

## Libraries - Lazyload
if (!require("pacman")) install.packages("pacman")
p_load(tidyverse, lubridate, zoo, data.table, Cairo, viridis)

## Pull Clean Data
dat_TAQ <- readRDS(file = "dat_TAQ.rds")

## Options
```

```r
options(digits.secs = 3)

# 1. Create Plots >> Trades/Quotes/MicroPrice ----

StartTime1 <- "10:00:00"
EndTime1 <- "11:00:00"
StartTime2 <- "16:00:00"
EndTime2 <- "17:00:00"
timezone <- "Africa/Johannesburg"

Stock1 <- "AGL: Anglo American PLC"
Stock2 <- "NPN: Naspers Ltd"

func.plot_EDA_1hour <- function(Data, StartTime, EndTime, Stock, Legend = FALSE)
{

  ## Randomly Select a Trading Day
  tmp_random_date <- Data %>%
    filter(Type == 'Trade') %>%
    slice(runif(1, 1, NROW(.))) %>%
    dplyr::select(DateTimeL)

  ## Define the Day
  tmp_ymd <- paste(year(tmp_random_date$DateTimeL),
                   month(tmp_random_date$DateTimeL),
                   day(tmp_random_date$DateTimeL), sep = "-")

  ## Need to subset for the Trading Hours
  date_start <- as.POSIXct(paste(tmp_ymd, StartTime, sep = " "), tz = timezone)
  date_end <- as.POSIXct(paste(tmp_ymd, EndTime, sep = " "), tz = timezone)
  plot_interval <- interval(date_start, date_end)

  dat_hourly <- Data[Data$DateTimeL %within% plot_interval, ]

  ## Data in nice plotting format >> Long
  testprice <- dat_hourly %>%
    rename(TradePrice = Price) %>%
    gather(key, price, c("L1.Ask", "L1.Bid","MicroPrice", "TradePrice")) %>%
    select(DateTimeL, key, price)

  testvols_quotes <- dat_hourly %>%
    gather(key, volume, Volume.Ask:Volume.Bid) %>%
    select(DateTimeL, key, volume)

  testvols_mp <- data.frame(DateTimeL = dat_hourly$DateTimeL,
                        key = "Volume.MicroPrice",
                        volume = ifelse(!is.na(dat_hourly$L1.Ask),
                                        0.5*min(testvols_quotes$volume,
                                                na.rm = TRUE), NA)) %>%
    mutate(key = as.character(key))

  testvols_trades <- dat_hourly %>%
    gather(key, volume, Volume.Trade) %>%
    select(DateTimeL, key, volume)

  testvols <- bind_rows(testvols_quotes, testvols_mp, testvols_trades)

  test_comb <- data.frame(DateTimeL = testprice$DateTimeL,
                          Event = as.factor(testprice$key),
                          Value = testprice$price,
                          Volume = testvols$volume)

  ## Generic Titles
  time.title <- paste("Trading between ",
                      unlist(strsplit(StartTime, ":"))[1],
                      "h", unlist(strsplit(StartTime, ":"))[2], " and ",
                      unlist(strsplit(EndTime, ":"))[1],
                      "h", unlist(strsplit(EndTime, ":"))[2], sep = "")

  date.title <- date(date_start)

  ## Plot EDA

```

```r
  plot <- test_comb %>% na.omit() %>%
    ggplot(aes(x = DateTimeL, y = Value/100, size = Volume, color = Event)) +
    geom_point() +
    theme_dark() +
    scale_color_manual(breaks = c("L1.Ask", "L1.Bid",
                                  "MicroPrice", "TradePrice"),
                       values=c("firebrick1", "royalblue1",
                                "chartreuse3", "yellow")) +
    labs(x = "\n Time (HH:MM:SS)", y = "Price (R's) \n", title = Stock,
         subtitle = paste(time.title, date.title, sep = " : ")) +
    scale_x_datetime(date_labels = "%H:%M:%S",
                     timezone = "Africa/Johannesburg") +
    theme(panel.background = element_rect(fill = 'grey15')) +
    theme(legend.key=element_blank()) +
    scale_size_continuous(guide = FALSE) +
    guides(color = guide_legend(override.aes = list(size = 3)))

  if(Legend == TRUE) {plot <- plot + theme(aspect.ratio=1)}
  else{plot <- plot + theme(legend.position ="none") + theme(aspect.ratio=1)}

  return(plot)

}

setwd(dir_figs)

# cairo_pdf("HFT_Ass1_fig_EDA_AGL_1011.pdf", height = 5, width = 5)
func.plot_EDA_1hour(dat_TAQ$AGL , StartTime1, EndTime1, Stock1, Legend = TRUE)
# dev.off()

# cairo_pdf("HFT_Ass1_fig_EDA_AGL_1617.pdf", height = 5, width = 5)
func.plot_EDA_1hour(dat_TAQ$AGL , StartTime2, EndTime2, Stock1, Legend = TRUE)
# dev.off()

# cairo_pdf("HFT_Ass1_fig_EDA_NPN_1011.pdf", height = 5, width = 5)
func.plot_EDA_1hour(dat_TAQ$NPN , StartTime1, EndTime1, Stock2, Legend = TRUE)
# dev.off()

# cairo_pdf("HFT_Ass1_fig_EDA_NPN_1617.pdf", height = 5, width = 5)
func.plot_EDA_1hour(dat_TAQ$NPN , StartTime2, EndTime2, Stock2, Legend = TRUE)
# dev.off()

setwd(dir_script)

# 2. Create Plots >> ACF ----

## Get TAQ that has +1(-1) Trade Signs only
dat_tradesigns <- lapply(dat_TAQ, function(x) x %>%
                           setDT() %>% .[!is.na(Trade.Sign) & Trade.Sign != 0])

## Total Length for N days
counts <- lapply(dat_tradesigns, function(x)  x %>%
                   setDT() %>% .[, length(Trade.Sign)])

## Get ACFs
acfs <- lapply( dat_tradesigns, function(x) x %>%
                  .[, acf(Trade.Sign, lag.max = length(Trade.Sign))] )

## Nice dataframes
dat_AGL_ACF <- tibble(ACF = as.numeric(acfs$AGL$acf),
                      Lag = as.numeric(acfs$AGL$lag))

dat_NPN_ACF <- tibble(ACF = as.numeric(acfs$NPN$acf),
                      Lag = as.numeric(acfs$NPN$lag))

## Plot the things
setwd(dir_figs)

# cairo_pdf("HFT_Ass1_fig_ACF_Orderflow_AGL.pdf", height = 5, width = 5)
ggplot(dat_AGL_ACF, aes(x = Lag, y = ACF)) +
  geom_line() +
  scale_x_log10(labels = function(x) format(x, scientific = FALSE)) +
  scale_y_log10(labels = function(x) format(x, scientific = TRUE)) +
```

```
  labs(y = "ACF \n", x = "\n Lag", title = Stock1,
       subtitle = "Autocorrelation of Order-Flow on log-log Scale") +
  theme_bw() +
  theme(aspect.ratio = 0.75)
# dev.off()

# cairo_pdf("HFT_Ass1_fig_ACF_Orderflow_NPN.pdf", height = 5, width = 5)
ggplot(dat_NPN_ACF, aes(x = Lag, y = ACF)) +
  geom_line() +
  scale_x_log10(labels = function(x) format(x, scientific = FALSE)) +
  scale_y_log10(labels = function(x) format(x, scientific = TRUE)) +
  labs(y = "ACF \n", x = "\n Lag", title = Stock2,
       subtitle = "Autocorrelation of Order-Flow on log-log Scale") +
  theme_bw() +
  theme(aspect.ratio = 0.75)
# dev.off()

setwd(dir_script)

# 3. Create Plots >> Inter-Arrival Times ----

## Frequency of Inter-Arrival Times

setwd(dir_figs)

### Anglo
dat_AGL <- dat_TAQ$AGL

# cairo_pdf("HFT_Ass1_fig_Freq10s_InterArrival_AGL.pdf", height = 5, width = 5)
ggplot(dat_AGL[Inter.Arrival.times < 10], aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 10) +
  scale_fill_viridis() +
  labs(title = Stock1,
       subtitle = "Times less than 10 seconds",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

# cairo_pdf("HFT_Ass1_fig_Freq60s_InterArrival_AGL.pdf", height = 5, width = 5)
ggplot(dat_AGL[Inter.Arrival.times < 60], aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 30) +
  scale_fill_viridis() +
  labs(title = Stock1,
       subtitle = "Times less than 60 seconds",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

# cairo_pdf("HFT_Ass1_fig_Freq10m_InterArrival_AGL.pdf", height = 5, width = 5)
ggplot(dat_AGL[Inter.Arrival.times < 600], aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 50) +
  scale_fill_viridis() +
  labs(title = Stock1,
       subtitle = "Times less than 10 minutes",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

# cairo_pdf("HFT_Ass1_fig_FreqAll_InterArrival_AGL.pdf", height = 5, width = 5)
ggplot(dat_AGL, aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 100) +
  scale_fill_viridis() +
  labs(title = Stock1,
       subtitle = "Times for all Trading Events - Absolute Scale",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

# cairo_pdf("HFT_Ass1_fig_FreqAllLog_InterArrival_AGL.pdf", height = 5, width = 5)
```

```
ggplot(dat_AGL, aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 15) +
  scale_fill_viridis() +
  scale_x_log10() +
  labs(title = Stock1,
       subtitle = "Times for all Trading Events - Log Scale",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

### Naspers
dat_NPN <- dat_TAQ$NPN

# cairo_pdf("HFT_Ass1_fig_Freq10s_InterArrival_NPN.pdf", height = 5, width = 5)
ggplot(dat_NPN[Inter.Arrival.times < 10], aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 10) +
  scale_fill_viridis() +
  labs(title = Stock2,
       subtitle = "Times less than 10 seconds",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

# cairo_pdf("HFT_Ass1_fig_Freq60s_InterArrival_NPN.pdf", height = 5, width = 5)
ggplot(dat_NPN[Inter.Arrival.times < 60], aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 30) +
  scale_fill_viridis() +
  labs(title = Stock2,
       subtitle = "Times less than 60 seconds",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

# cairo_pdf("HFT_Ass1_fig_Freq10m_InterArrival_NPN.pdf", height = 5, width = 5)
ggplot(dat_NPN[Inter.Arrival.times < 600], aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 50) +
  scale_fill_viridis() +
  labs(title = Stock2,
       subtitle = "Times less than 10 minutes",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

# cairo_pdf("HFT_Ass1_fig_FreqAll_InterArrival_NPN.pdf", height = 5, width = 5)
ggplot(dat_NPN, aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 100) +
  scale_fill_viridis() +
  labs(title = Stock2,
       subtitle = "Times for all Trading Events - Absolute Scale",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

# cairo_pdf("HFT_Ass1_fig_FreqAllLog_InterArrival_NPN.pdf", height = 5, width = 5)
ggplot(dat_NPN, aes(x = Inter.Arrival.times)) +
  geom_histogram(aes(fill = ..count..), colour="black", bins = 14) +
  scale_fill_viridis() +
  scale_x_log10() +
  labs(title = Stock1,
       subtitle = "Times for all Trading Events - Log Scale",
       y = "", x = "\n Inter-Arrival Times (Seconds)") +
  theme_bw() +
  theme(aspect.ratio = 1, legend.position = "none")
# dev.off()

## QQ Plots

# cairo_pdf("HFT_Ass1_fig_QQExp_InterArrival_AGL.pdf", height = 5, width = 5)
```

```r
ggplot(dat_AGL, aes(sample = Inter.Arrival.times)) +
  stat_qq_line(distribution = stats::qexp, colour = "firebrick1") +
  stat_qq(distribution = stats::qexp, geom = "point") +
  labs(x = "\n Exponential Quantiles", y = "Inter-Arrival Quantiles \n",
       title = Stock1,
       subtitle = "QQ-Plot Relative to Exponential Distribution") +
  theme_bw() +
  theme(aspect.ratio = 1)
# dev.off()

# cairo_pdf("HFT_Ass1_fig_QQExp_InterArrival_NPN.pdf", height = 5, width = 5)
ggplot(dat_NPN, aes(sample = Inter.Arrival.times)) +
  stat_qq_line(distribution = stats::qexp, colour = "firebrick1") +
  stat_qq(distribution = stats::qexp, geom = "point") +
  labs(x = "\n Exponential Quantiles", y = "Inter-Arrival Quantiles \n",
       title = Stock2,
       subtitle = "QQ-Plot Relative to Exponential Distribution") +
  theme_bw() +
  theme(aspect.ratio = 1)
# dev.off()

## ACF Plots of Inter-Arrival Times

acfs2 <- lapply( dat_tradesigns, function(x) x %>%
                   .[, acf(na.omit(Inter.Arrival.times),
                           lag.max = length(na.omit(Inter.Arrival.times)))] )

## Nice dataframes
dat_AGL_ACF_times <- tibble(ACF = as.numeric(acfs2$AGL$acf),
                            Lag = as.numeric(acfs2$AGL$lag))

dat_NPN_ACF_times <- tibble(ACF = as.numeric(acfs2$NPN$acf),
                            Lag = as.numeric(acfs2$NPN$lag))

## Plot the things
setwd(dir_figs)

# cairo_pdf("HFT_Ass1_fig_ACF_InterArrival_AGL.pdf", height = 5, width = 5)
ggplot(dat_AGL_ACF_times, aes(x = Lag, y = ACF)) +
  geom_line() +
  scale_x_log10(labels = function(x) format(x, scientific = FALSE)) +
  scale_y_log10(labels = function(x) format(x, scientific = TRUE)) +
  labs(y = "ACF \n", x = "\n Lag", title = Stock1,
       subtitle = "Autocorrelation of Inter-Arrival Times on log-log Scale") +
  theme_bw() +
  theme(aspect.ratio = 0.75)
# dev.off()

# cairo_pdf("HFT_Ass1_fig_ACF_InterArrival_NPN.pdf", height = 5, width = 5)
ggplot(dat_NPN_ACF_times, aes(x = Lag, y = ACF)) +
  geom_line() +
  scale_x_log10(labels = function(x) format(x, scientific = FALSE)) +
  scale_y_log10(labels = function(x) format(x, scientific = TRUE)) +
  labs(y = "ACF \n", x = "\n Lag", title = Stock2,
       subtitle = "Autocorrelation of Inter-Arrival Times on log-log Scale") +
  theme_bw() +
  theme(aspect.ratio = 0.75)
# dev.off()

setwd(dir_script)

#------------------------------------------------------------------------------#
```

Plagiarism Declaration Form

A copy of this form, completed and signed, to be attached to all coursework submissions to the Statistical Sciences Department.

COURSE CODE:	**STA5091Z**
COURSE NAME:	**Data Analysis for High-Frequency Trading**
STUDENT NAME:	**Julian Albert**
STUDENT NUMBER:	**ALBJUL005**
GROUP NUMBER:	**1**

PLAGIARISM DECLARATION

- I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
- I have used a generally accepted citation and referencing style. Each contribution to, and quotation in, this tutorial/report/project from the work(s) of other people has been attributed, and has been cited and referenced.
- This tutorial/report/project is my own work.
- I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
- I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.
- Agreement to this statement does not exonerate me from the University's plagiarism rules.

Signature:  Date: August 7, 2019