Brent Crude Futures & Luxor Trading Strategy

Kopal Jha

10/2/2020

This project was structured around financial data visualization and trading system design on R. Brent crude oil futures were selected for this analysis. Part 1 focuses on visualizing historical prices for Brent crude oil futures and summarizing important features of the historical distribution. Part 2 involves writing a script for the Luxor trading strategy from Jaekle and Tomasini's Trading Systems textbook.

Part 1 - Data Visualization

In the first part of this project, my objective was to get familiar with my dataset of Brent crude futures (historical prices) using baseR and the Quandl and ggplot packages for data visualization.

```
library(Quandl)
```

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

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

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

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

```
library(tidyverse)
```

```
## -- Attaching packages ----------------------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts -------------------------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::first()  masks xts::first()
## x dplyr::lag()    masks stats::lag()
## x dplyr::last()   masks xts::last()
```

```
library(ggthemes)
library(quantmod)
```

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

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

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

```
library(quantstrat)
```

```
## Loading required package: blotter
```

```
## Loading required package: FinancialInstrument
```

```
##
## Attaching package: 'FinancialInstrument'
```

```
## The following object is masked from 'package:tidyr':
##
##     spread
```

```
## Loading required package: PerformanceAnalytics
```

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

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

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

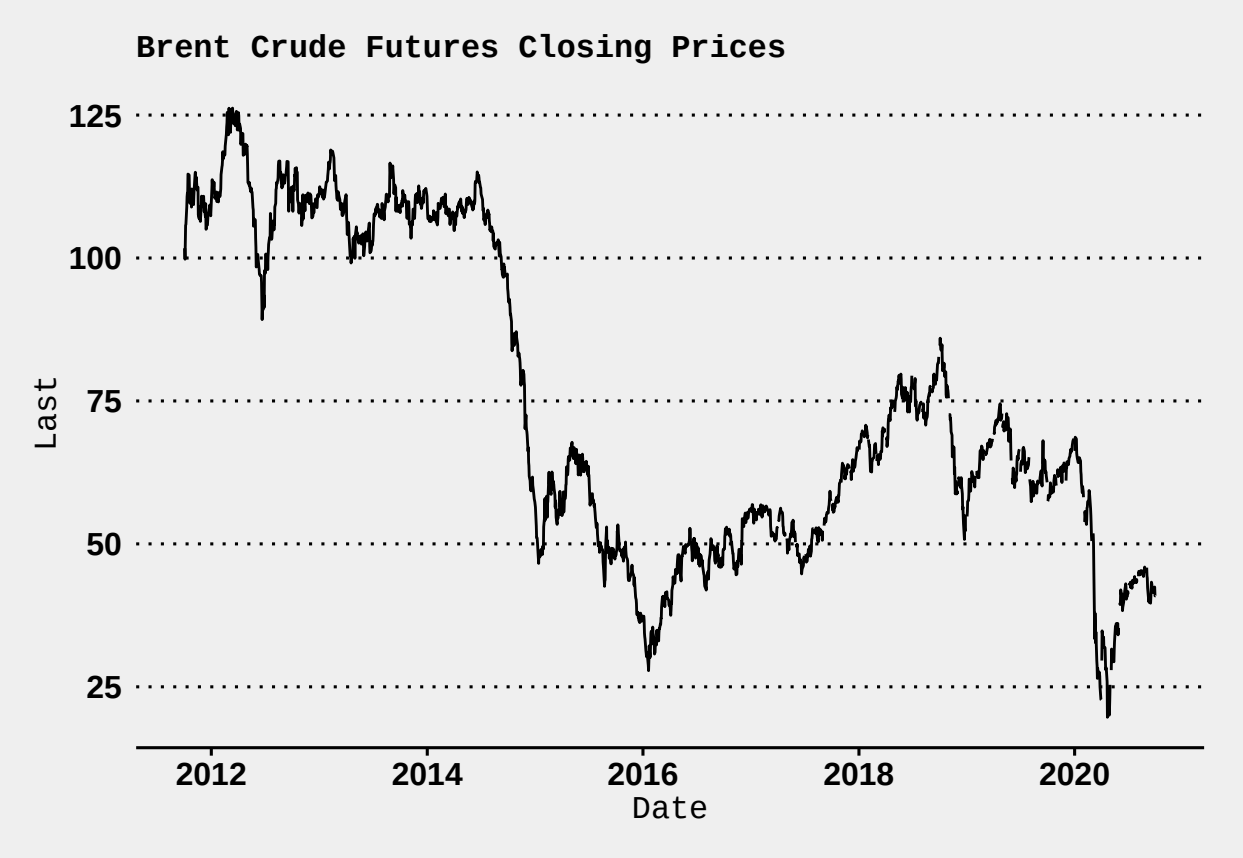
```
library(knitr)
library(blotter)

Quandl.api_key("httWUrzTVAL6RdqaHqSf")

brent <- Quandl('CHRIS/CME_BZ1') # calls first month contract for Brent crude futures on CME exchange
head(brent, n=10)
```

```
##          Date  Open  High   Low  Last Change Settle Volume
## 1  2020-10-01    NA    NA    NA    NA   0.01  40.96      0
## 2  2020-09-30 40.75 40.97 40.31 40.92  -0.08  40.95    517
## 3  2020-09-29 42.43 42.52 40.43 40.90  -1.40  41.03   9403
## 4  2020-09-28 41.85 42.62 41.48 42.46   0.51  42.43  10821
## 5  2020-09-25 41.93 42.33 41.53 41.79  -0.02  41.92  17085
## 6  2020-09-24 41.50 42.02 41.11 41.77   0.17  41.94  19040
## 7  2020-09-23 41.74 42.64 41.22 41.50   0.05  41.77  23179
## 8  2020-09-22 41.79 42.19 41.21 41.73   0.28  41.72  19494
## 9  2020-09-21 43.02 43.30 40.97 41.73  -1.71  41.44  24298
## 10 2020-09-18 43.26 43.81 42.53 43.08  -0.15  43.15  26022
##    Previous Day Open Interest
## 1                      9316
## 2                      9262
## 3                      9927
## 4                     11786
## 5                     14062
## 6                     16794
## 7                     17877
```

```
## 8                        17735
## 9                        18375
## 10                       18361
```

```
# WSJ-style plot of historical futures closing prices
ggplot(brent, aes(x = Date, y = Last)) +
  geom_line() +
  theme_wsj(color = "gray") +
  theme(axis.title=element_text(size=12)) +
  ggtitle("Brent Crude Futures Closing Prices") +
  theme(plot.title = element_text(size = 12, face = "bold"))
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```



Cleaning the data

```
# Arrange data in reverse chronological order (oldest first) and sort by consecutive dates
brent <- brent %>% arrange(rev(rownames(.))) %>% mutate(date = as.Date(Date, "%d-%m-%Y")) %>% arrange(da
brent$returns <- as.numeric(c('NA',diff(log(brent$Last)))) # Adding a new column with the differences b
```

```
## Warning: NAs introduced by coercion
```

```
BZ_DATA <- select_if(brent, is.numeric)
# Get the number of NAN (not a number) values by column
na_cols_count <-sapply(BZ_DATA, function(y) sum(length(which(is.na(y)))))
na_cols_count[c("Open", "High", "Low", "Last", "Change", "Settle", "Volume", "Previous Day Open Interes
```

```
##                     Open                     High
##                       46                       43
```

```
##                         Low                        Last
##                          43                          43
##                      Change                      Settle
##                         606                           0
##                      Volume Previous Day Open Interest
##                           0                           0
##                     returns
##                          86
```

```
# This gives how many missing values are in each column. The "Change" column has too many missing value
BZ_DATA <- select(BZ_DATA,c(1,2,3,4,6,7,8,9))
# To account for the missing data in other columns, we select for only complete cases
BZ_DATA <- BZ_DATA[complete.cases(BZ_DATA),]
```

Summary statistics

```
summary(BZ_DATA) # Shortcut summary
```

```
##       Open             High             Low              Last
##  Min.   : 19.56   Min.   : 21.27   Min.   : 15.99   Min.   : 19.68
##  1st Qu.: 51.74   1st Qu.: 52.54   1st Qu.: 51.14   1st Qu.: 51.72
##  Median : 65.54   Median : 66.37   Median : 64.64   Median : 65.56
##  Mean   : 74.83   Mean   : 75.65   Mean   : 73.95   Mean   : 74.82
##  3rd Qu.:107.01   3rd Qu.:107.86   3rd Qu.:106.24   3rd Qu.:106.98
##  Max.   :125.86   Max.   :128.35   Max.   :125.06   Max.   :126.22
##      Settle           Volume      Previous Day Open Interest
##  Min.   : 19.33   Min.   :    0   Min.   :  623
##  1st Qu.: 51.81   1st Qu.: 4162   1st Qu.: 7836
##  Median : 65.51   Median :19278   Median :15265
##  Mean   : 74.82   Mean   :19352   Mean   :15833
##  3rd Qu.:106.98   3rd Qu.:31108   3rd Qu.:22304
##  Max.   :126.22   Max.   :86635   Max.   :49548
##     returns
##  Min.   :-0.3098155
##  1st Qu.:-0.0091418
##  Median : 0.0000000
##  Mean   :-0.0004738
##  3rd Qu.: 0.0090780
##  Max.   : 0.1615556
```

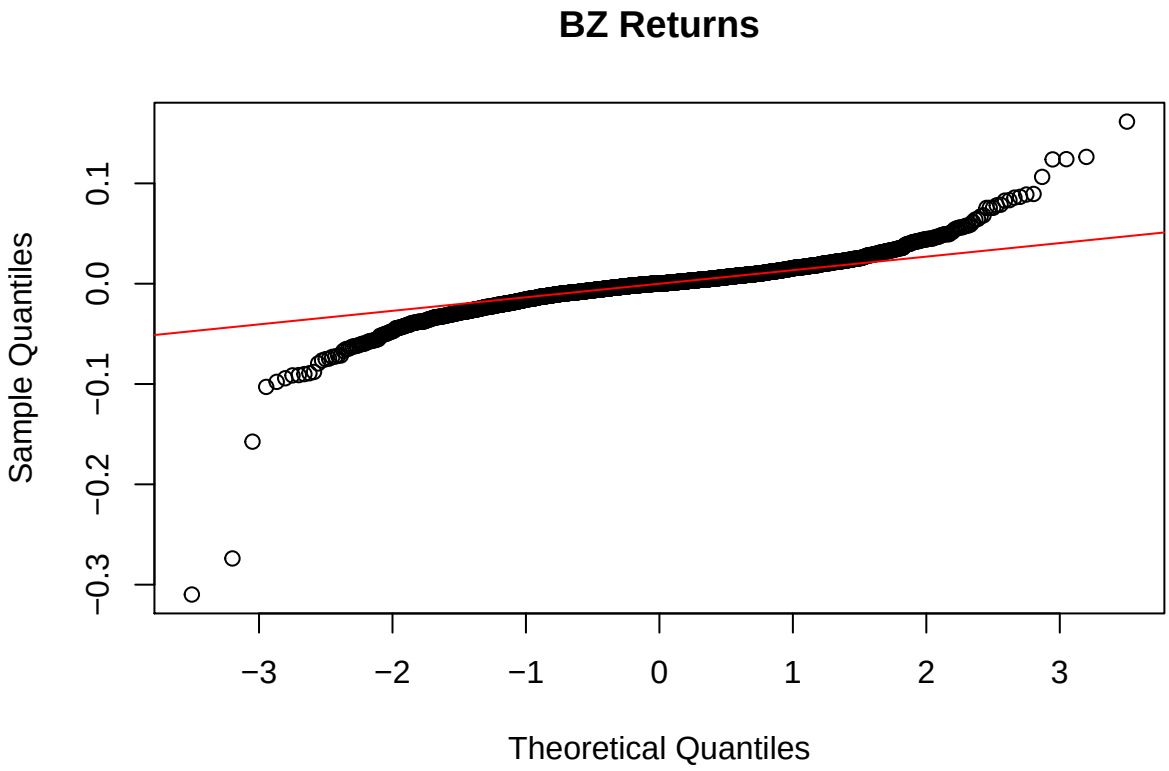
```
(statistics <- do.call(data.frame,
                       list(mean = round(apply(BZ_DATA, 2, mean),4),
                            sd = round(apply(BZ_DATA, 2, sd),4),
                            median = round(apply(BZ_DATA, 2, median),4),
                            min = round(apply(BZ_DATA, 2, min),4),
                            max = round(apply(BZ_DATA, 2, max),4)))) # Better formatted summary
```

```
##                                mean         sd   median      min        max
## Open                        74.8301    27.4028    65.54  19.5600   125.8600
## High                        75.6502    27.3648    66.37  21.2700   128.3500
## Low                         73.9525    27.4403    64.64  15.9900   125.0600
## Last                        74.8174    27.4402    65.56  19.6800   126.2200
## Settle                      74.8164    27.4473    65.51  19.3300   126.2200
## Volume                   19351.5140 15785.8446 19278.00   0.0000 86635.0000
## Previous Day Open Interest 15832.8795  9701.6977 15265.00 623.0000 49548.0000
```

```
## returns                       -0.0005      0.0230      0.00  -0.3098      0.1616
```

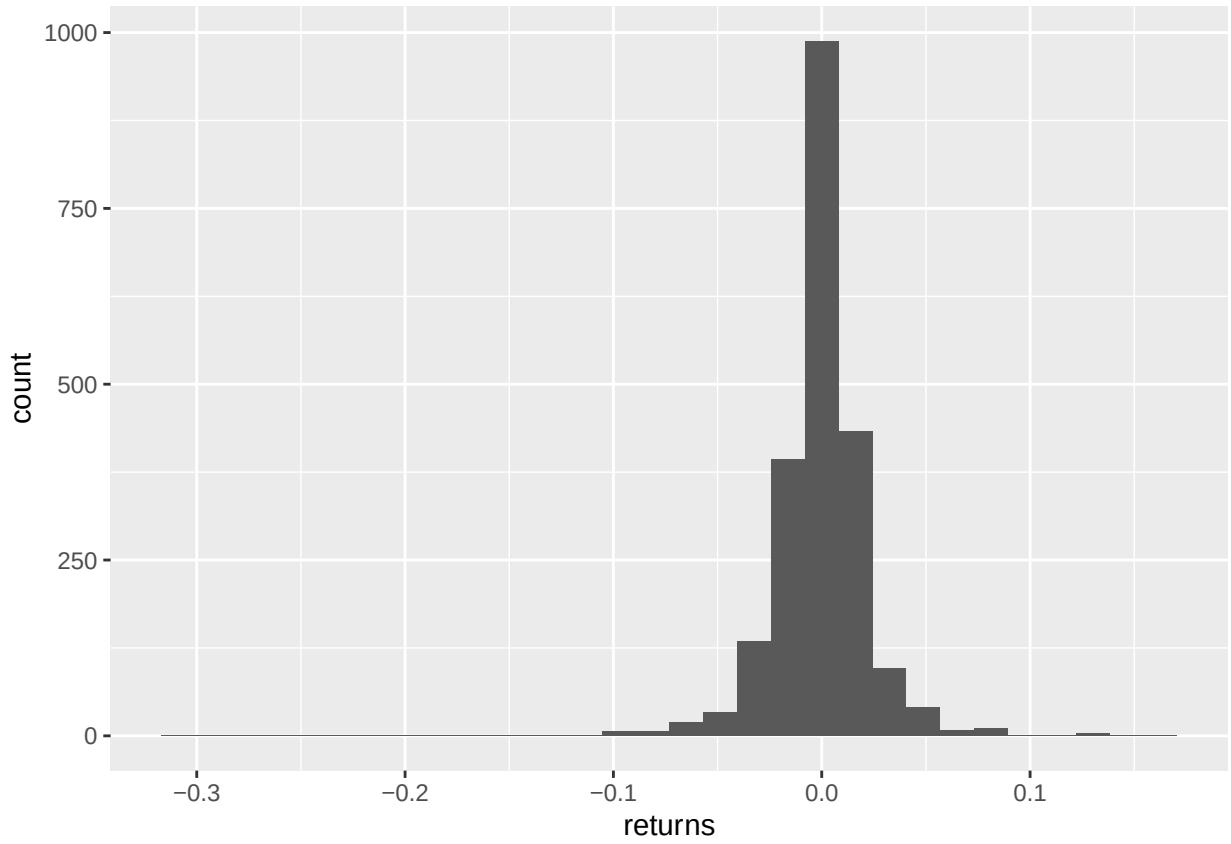
Distribution of returns

```
returns <- brent$returns
qqnorm(returns, main="BZ Returns")
qqline(returns, col="red")
```




```
# The qqplots indicate the data is non-normally distributed
# The finding that the futures price data is non-normal is consistent with the fact that futures prices
ggplot(BZ_DATA, aes(x = returns)) +
  geom_histogram()
```

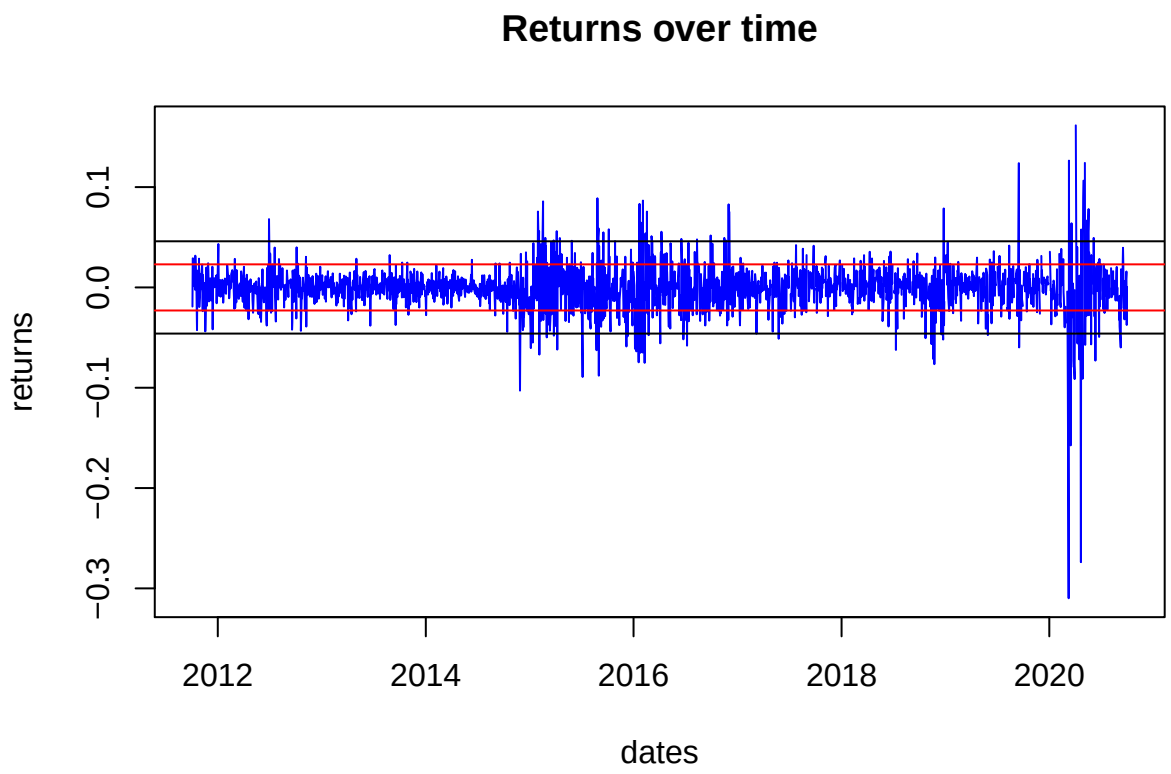
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



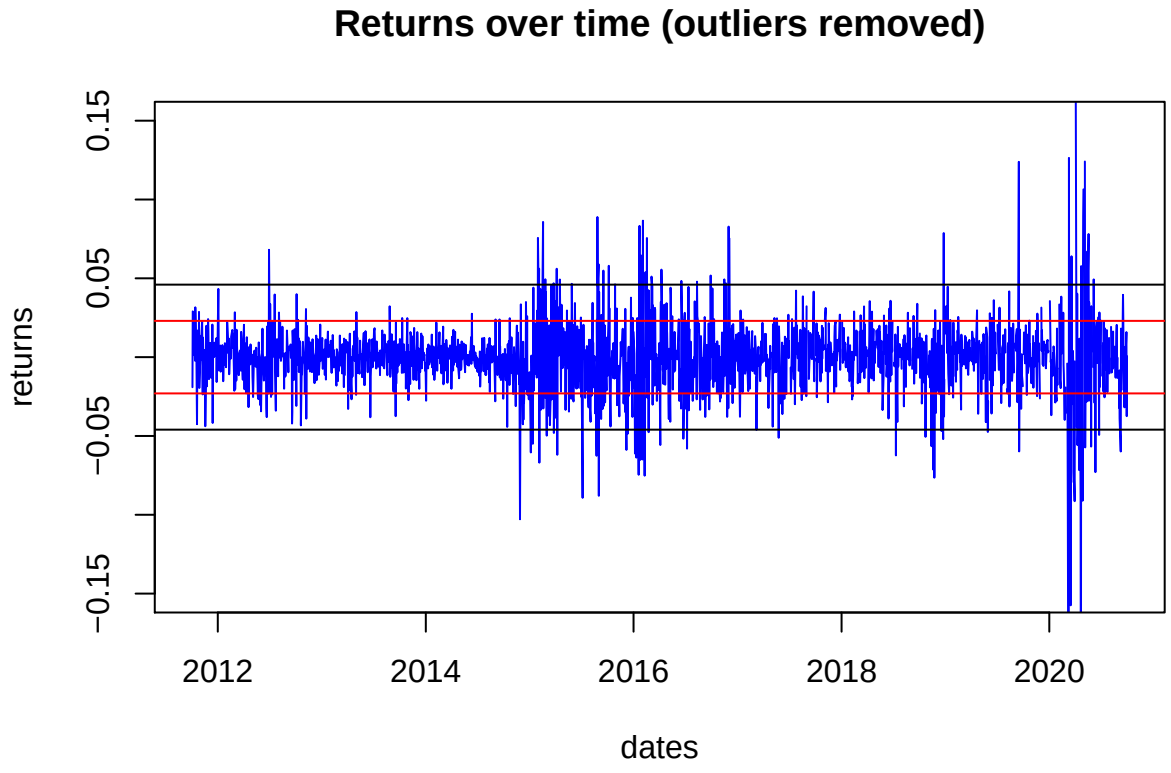

```
# The histogram confirms the leptokurtic nature of these prices
```

Historical volatility

```
dates <- brent$date
one_std <- sd(returns,na.rm = TRUE)
two_std <- one_std * 2
plot(returns ~ dates, type='l',col='blue', main = "Returns over time")
abline(h=c(-one_std,one_std),col='red')
abline(h=c(-two_std,two_std),col='black')
```



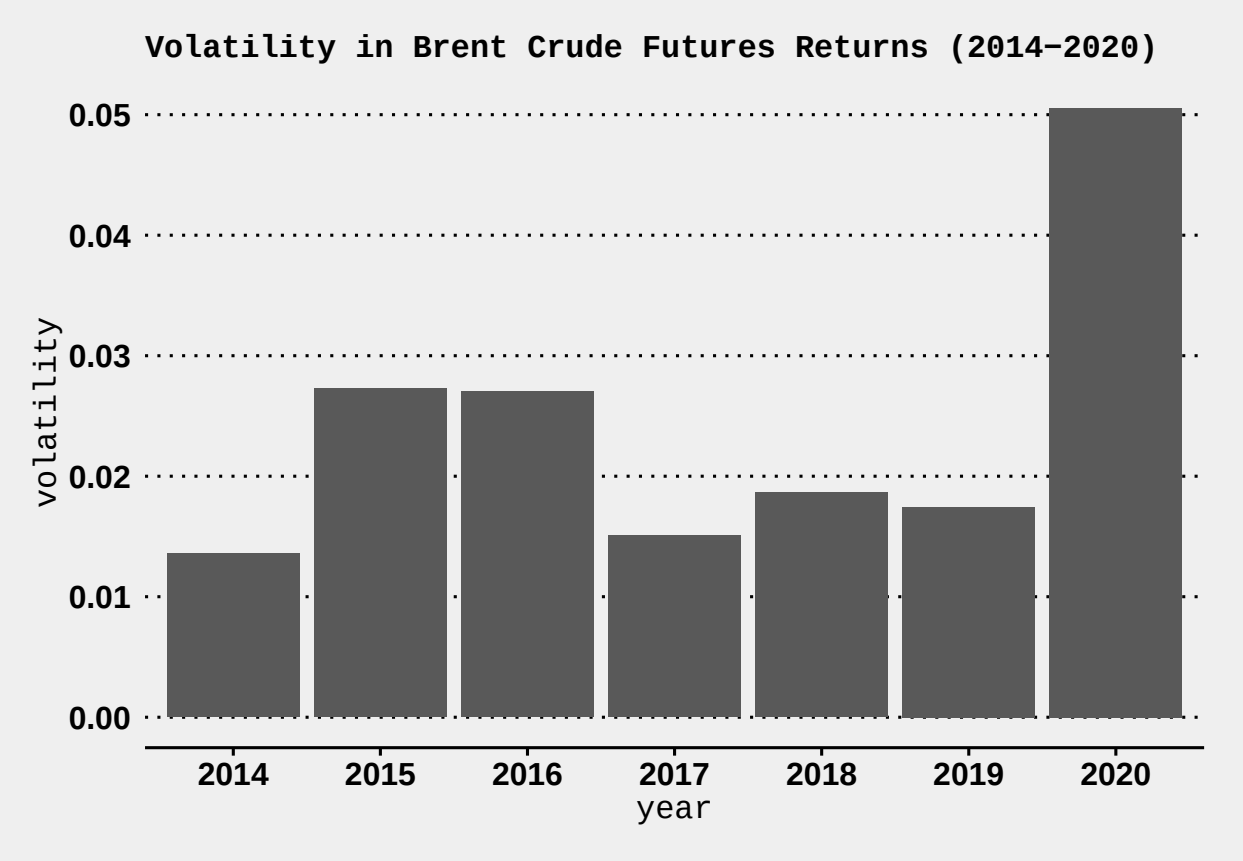

```
# This plot shows that the Brent futures were most volatile in 2020.

plot(returns ~ dates, type='l',col='blue', main = "Returns over time (outliers removed)", ylim = c(-0.15
abline(h=c(-one_std,one_std),col='red')
abline(h=c(-two_std,two_std),col='black')
```



```
brent_volatility <- brent %>% subset(date > '2010-01-01') %>% mutate(year = substr(brent$date,1,4))
brent_volatility <- brent_volatility[complete.cases(brent_volatility), ]
# We then append a new column to this dataset using standard deviation as a measure of volatility
brent_volatility <- brent_volatility %>% group_by(year) %>% summarise(volatility = sd(returns)) %>% muta
```

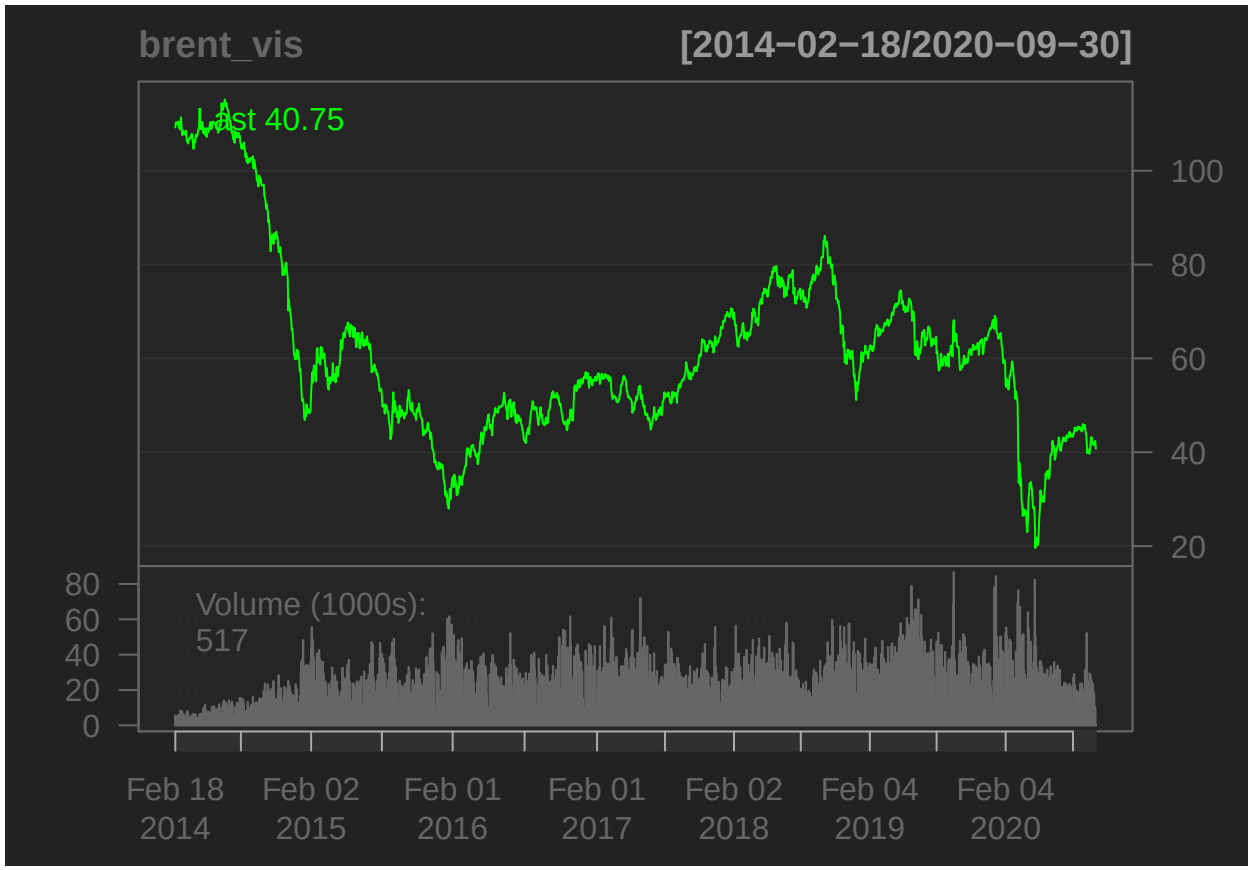
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ggplot(brent_volatility, aes(x=year, y=volatility)) +
  geom_col() +
  theme_wsj(color = "gray") +
  theme(axis.title=element_text(size=12)) +
  ggtitle("Volatility in Brent Crude Futures Returns (2014-2020)") +
  theme(plot.title = element_text(size = 12, face = "bold"))
```

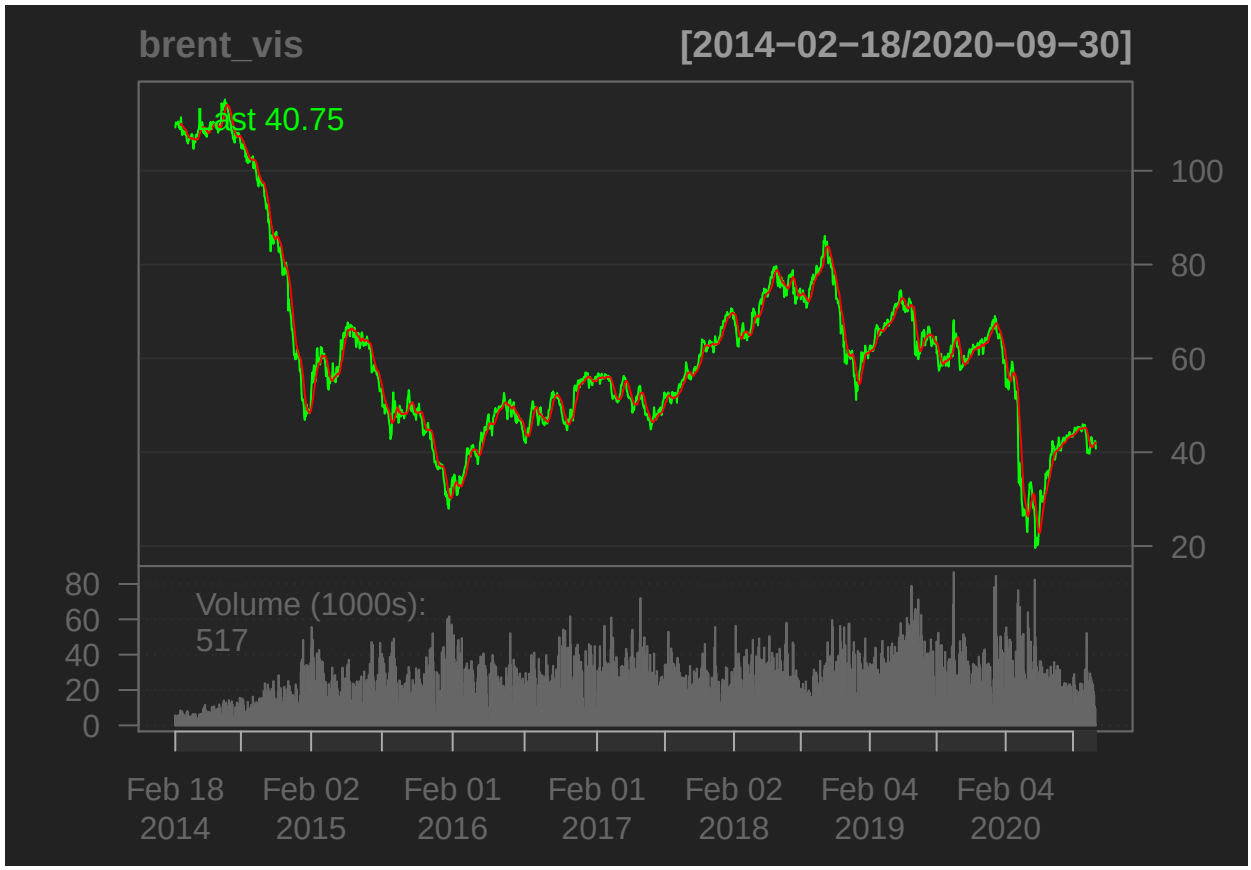


Candle charts and two technical indicators

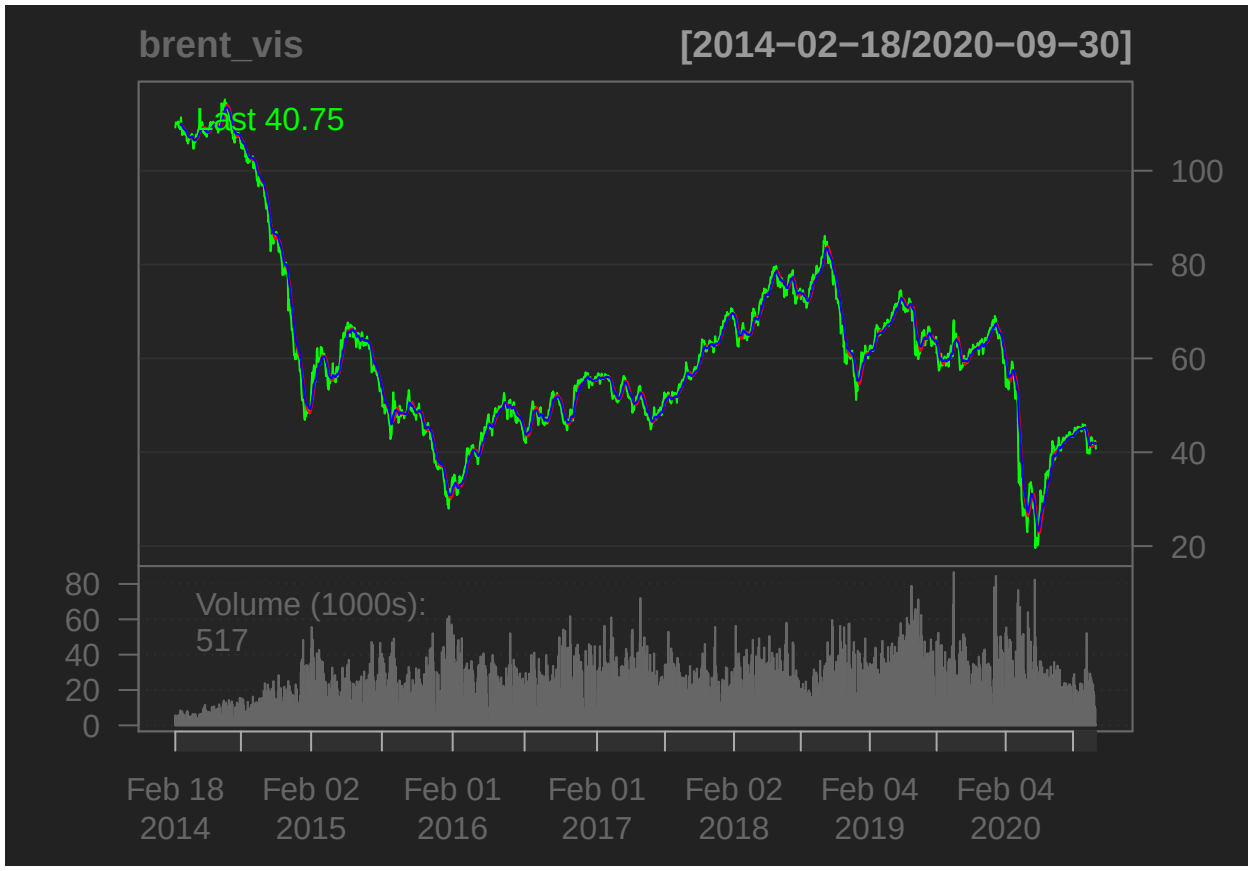
```
brent_vis <- Quandl('CHRIS/CME_BZ1', type = "xts")
candleChart(brent_vis)
```




```
addSMA(col="red")
```


brent_vis
[2014−02−18/2020−09−30]
Last 40.75
100
80
60
40
20
Volume (1000s):
517
80
60
40
20
0
Feb 18
2014
Feb 02
2015
Feb 01
2016
Feb 01
2017
Feb 02
2018
Feb 04
2019
Feb 04
2020

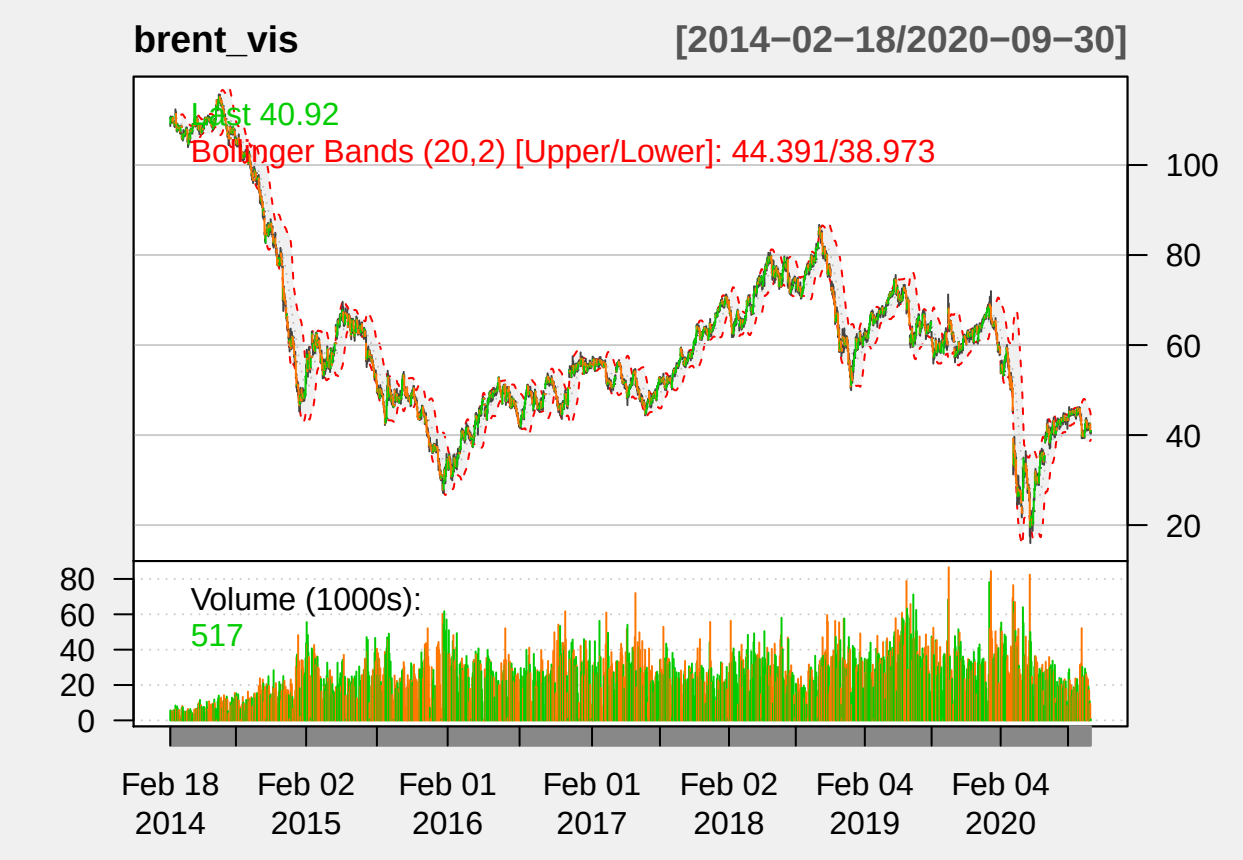
```
addEMA()
```



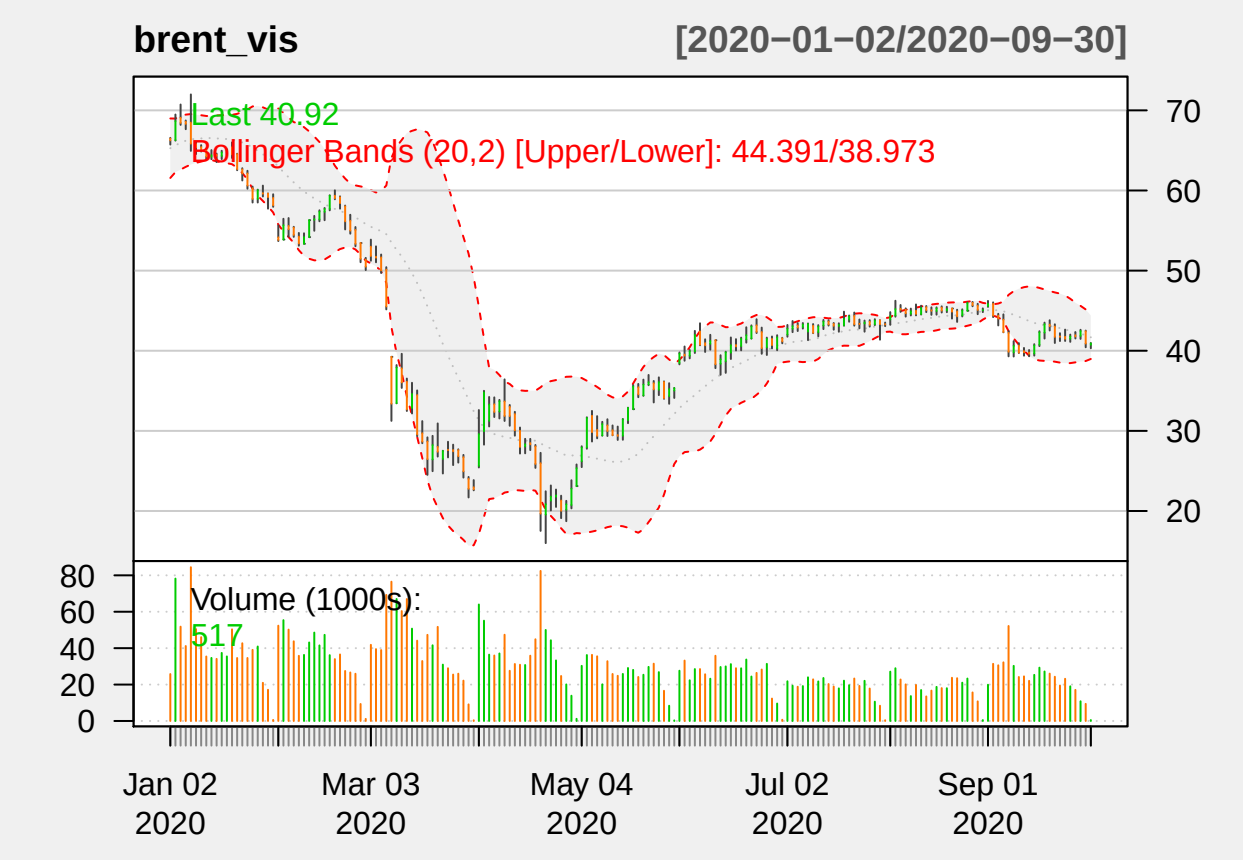

```
colnames(brent_vis)[4] <- "Close"
brent_vis$date <- time(brent_vis) #Extracts dates as numeric values
```

Bollinger Bands and volume

```
chartSeries(brent_vis, subset="2014::2020",
            theme = chartTheme("white"),
            TA = "addVo(); addBBands(n = 20, sd = 2, draw = 'bands', on = -1)")
```

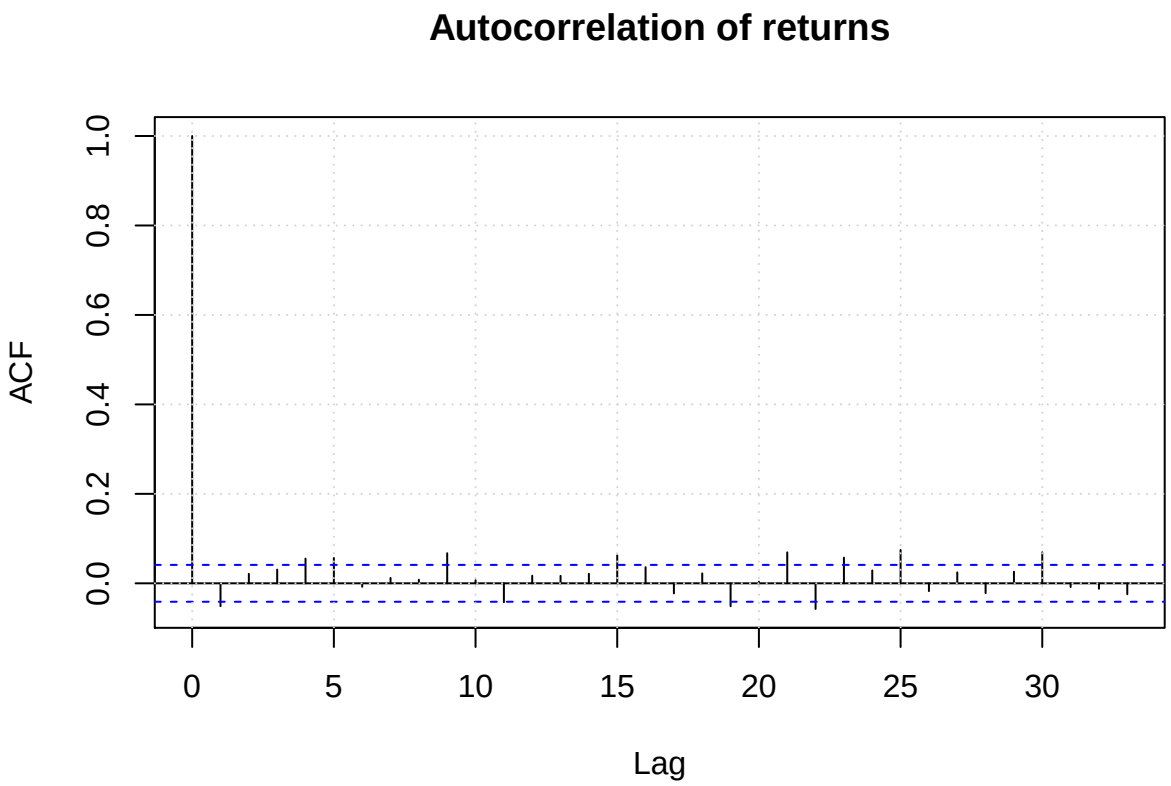



```
zoomChart("2020:", yrange=NULL) # Zooms in to show bands for 2020
```

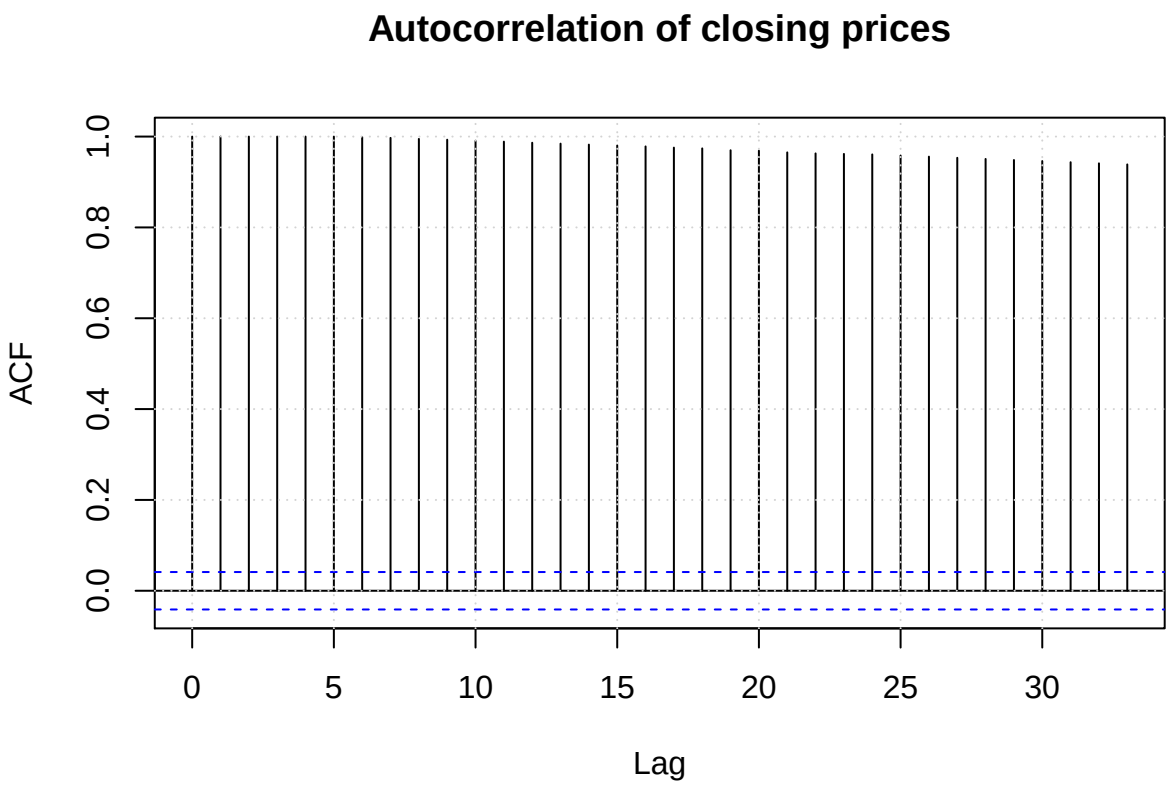


Autocorrelation of returns and closing prices

```
acf(brent$returns, main = "Autocorrelation of returns", na.action = na.pass)
grid()
```




```
acf(brent$Last, main = "Autocorrelation of closing prices", na.action = na.pass)
grid()
```



```
# These autocorrelation plots indicate that closing prices are more autocorrelated than returns
```

Part 2(a) - Luxor Trading Strategy

In the second part of this project, my objective was to script the Luxor trading strategy in order to better understand the workflow of trading strategy development. Luxor is a trend-following method that uses moving averages as entry signal generators. A long trade is initiated if the fast simple moving average (here, a 10-period SMA) is greater than or equal to the slow SMA (here, a 30-period SMA). If SMA(10) < SMA(30), a short order is submitted.

Setup

```
# Data
currency('USD')
```

```
## [1] "USD"
```

```
sym <- na.omit(get(getSymbols("BZ=F"))["2019::2020"])
```

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

```
## Warning: BZ=F contains missing values. Some functions will not work if objects
## contain missing values in the middle of the series. Consider using na.omit(),
## na.approx(), na.fill(), etc to remove or replace them.
```

```
mkdata <- Cl(sym)
future("BZ=F", currency = "USD", multiplier = 1000)
```

```
## Warning in future("BZ=F", currency = "USD", multiplier = 1000): underlying_id
## should only be NULL for cash-settled futures
```

```
## [1] "BZ.F"
```

```
# Set the parameters
initDate <- "2019-01-01" #Date of initiation
from <- "2019-01-01" #Start date of the data set
to <- "2020-10-01" #End date of the data set
initEq <- 1e5 #Initial equity of $100,000

# Assign names to portfolio and account
strategy.st <- portfolio.st <- account.st <- "luxor"

# Initiate portfolio and account
initPortf(portfolio.st, "sym", initDate = initDate) #Initiate portfolio
```

```
## [1] "luxor"
```

```
initAcct(account.st, portfolios = strategy.st, initDate = initDate, initEq = initEq) #Initiate account
```

```
## [1] "luxor"
```

```
initOrders(portfolio = portfolio.st, initDate = initDate) #Initiate account
strategy(strategy.st, store = TRUE) #Store all the events in the strategy
```

Indicators

```
add.indicator(strategy = strategy.st,
              name = "SMA",
              arguments = list(x = quote(Cl(mkdata)), # Note these indicators are using closing prices
                               n = 10),
              label = "nFast") # Fast Moving Average ie SMA(10)
```

```
## [1] "luxor"
```

```
add.indicator(strategy = strategy.st,
              name = "SMA",
              arguments = list(x = quote(Cl(mkdata)),
                               n = 30),
              label = "nSlow") # Slow Moving Average ie SMA(30)
```

```
## [1] "luxor"
```

Signals

```
add.signal(strategy = strategy.st,
           name="sigCrossover",
           arguments = list(columns = c("nFast", "nSlow"),
                            relationship = "gte"), # The long signal occurs when the fast SMA is > or =
           label = "long")
```

```
## [1] "luxor"
```

```
add.signal(strategy = strategy.st,
           name="sigCrossover",
           arguments = list(columns = c("nFast", "nSlow"),
                            relationship = "lt"), # The short signal occurs when the fast SMA is < the
           label = "short")
```

```
## [1] "luxor"
```

Entry rules

```
add.rule(strategy = strategy.st,
         name = "ruleSignal",
         arguments = list(sigcol = "long",
                          sigval = TRUE,
                          orderqty = 100,
                          ordertype = "stoplimit",
                          orderside = "long",
                          threshold = 0.005,
                          prefer = "High",
                          TxnFees = -10,  # Assuming my broker charges $10 for placing a long order
                          replace = FALSE), # Will not replace any open orders
         type = "enter",
         label = "EnterLONG")
```

```
## [1] "luxor"
```

```
add.rule(strategy.st,
         name = "ruleSignal",
         arguments = list(sigcol = "short",
                          sigval = TRUE,
                          orderqty = -100,
                          ordertype = "stoplimit",
                          threshold = -0.005,
                          orderside = "short",
                          replace = FALSE,
                          TxnFees = -10,
                          prefer = "Low"),
         type = "enter",
         label = "EnterSHORT")
```

```
## [1] "luxor"
```

Exit rules

```
add.rule(strategy.st,
         name = "ruleSignal",
         arguments = list(sigcol = "short",
                          sigval = TRUE,
                          orderside = "long", # We exit our long positions when going short
                          ordertype = "market", # Buys the future at the prevailing market price
                          orderqty = "all",
                          TxnFees = -10,
                          replace = TRUE), # Any open orders are replaced
         type = "exit",
         label = "Exit2SHORT")
```

```
## [1] "luxor"
```

```
add.rule(strategy.st,
         name = "ruleSignal",
         arguments = list(sigcol = "long",
                          sigval = TRUE,
                          orderside = "short",
                          ordertype = "market",
                          orderqty = "all",
                          TxnFees = -10,
                          replace = TRUE),
         type = "exit",
         label = "Exit2LONG")
```

```
## [1] "luxor"
```

Execute

```
# Apply the strategy
applyStrategy(strategy.st, portfolio.st)
```

```
## Warning in getInstrument(symbol): instrument sym not found, please create it
## first.
```

```
## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in addTxn(Portfolio = portfolio, Symbol = symbol, TxnDate = txntime, :
## Instrument sym not found, using contract multiplier of 1

## [1] "2019-05-12 20:00:00 sym -100 @ 70.164998"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2019-07-04 20:00:00 sym 100 @ 64.230003"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2019-07-09 20:00:00 sym 100 @ 65.505"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2019-07-25 20:00:00 sym -100 @ 63.459999"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2019-07-25 20:00:00 sym -100 @ 63.024999"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2019-09-09 20:00:00 sym 100 @ 62.380001"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2019-09-09 20:00:00 sym 100 @ 62.995002"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1
```

```
## [1] "2019-10-06 20:00:00 sym -100 @ 58.349998"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2019-10-07 20:00:00 sym -100 @ 57.594998"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2019-10-30 20:00:00 sym 100 @ 60.23"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2019-11-03 19:00:00 sym 100 @ 62"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2020-01-16 19:00:00 sym -100 @ 64.849998"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2020-01-21 19:00:00 sym -100 @ 63.914998"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2020-04-13 20:00:00 sym 100 @ 29.6"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.

## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1

## [1] "2020-05-10 20:00:00 sym 100 @ 30.724999"

## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.
```

```
## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1
```

```
## [1] "2020-09-08 20:00:00 sym -100 @ 40.790001"
```

```
## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.
```

```
## Warning in getInstrument(Symbol): Instrument sym not found, using contract
## multiplier of 1
```

```
## [1] "2020-09-08 20:00:00 sym -100 @ 39.294999"
```

```
# Update portfolio, account, and equity
updatePortf(portfolio.st)
```

```
## Warning in getInstrument(symbol): instrument sym not found, please create it
## first.
```

```
## Warning in getInstrument(Symbol): instrument sym not found, please create it
## first.
```

```
## Warning in .updatePosPL(Portfolio = pname, Symbol = as.character(symbol), :
## Instrument sym not found, things may break
```

```
## [1] "luxor"
```

```
updateAcct(account.st)
```

```
## [1] "luxor"
```

```
updateEndEq(account.st)
```

```
## [1] "luxor"
```

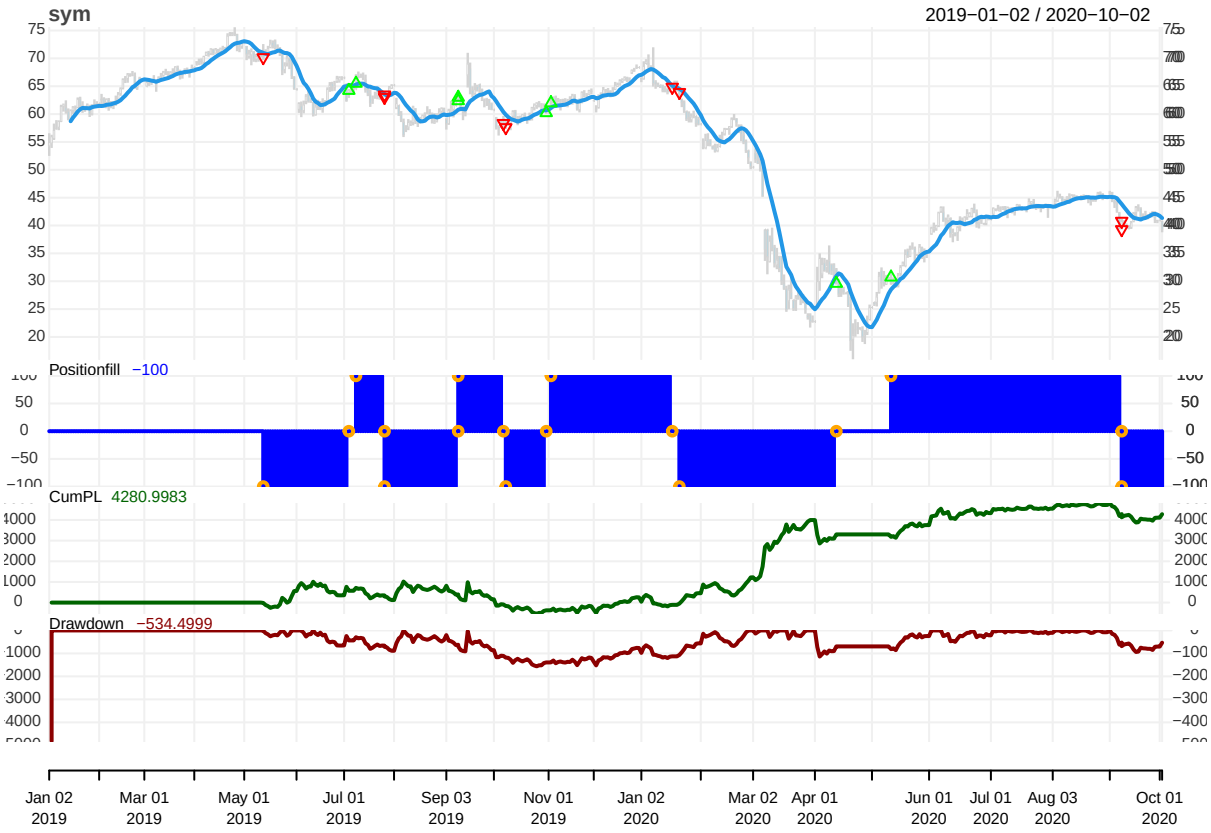
Part 2(b) - Performance Analysis

In part 2(b), I analyzed the results of my trading strategy using several performance measures.

Simple plot of performance

```
myTheme <- chart_theme()
myTheme$col$dn.col<-'lightblue'
myTheme$col$dn.border <- 'lightgray'
myTheme$col$up.border <- 'lightgray'
chart.Posn(strategy.st,theme=myTheme,
           TA='add_SMA(n=10,col=4, on=1, lwd=2)')
```

```
## Warning in mapply(function(name, value) {: longer argument not a multiple of
## length of shorter
```



Trade Statistics

```
tstats <- tradeStats(portfolio.st)
kable(t(tstats))
```

	sym
Portfolio	luxor
Symbol	sym
Num.Txns	17
Num.Trades	8
Net.Trading.PL	4280.998
Avg.Trade.PL	546.0623
Med.Trade.PL	164.7498
Largest.Winner	3421.5
Largest.Loser	-474.5004
Gross.Profits	5330.999
Gross.Losses	-962.5007
Std.Dev.Trade.PL	1258.073
Std.Err.Trade.PL	444.7959
Percent.Positive	62.5
Percent.Negative	37.5
Profit.Factor	5.538696
Avg.Win.Trade	1066.2
Med.Win.Trade	583.4995
Avg.Losing.Trade	-320.8336
Med.Losing.Trade	-273.5002

	sym
Avg.Daily.PL	546.0623
Med.Daily.PL	164.7498
Std.Dev.Daily.PL	1258.073
Std.Err.Daily.PL	444.7959
Ann.Sharpe	6.890277
Max.Drawdown	-1562.001
Profit.To.Max.Draw	2.740715
Avg.WinLoss.Ratio	3.323218
Med.WinLoss.Ratio	2.133452
Max.Equity	4815.498
Min.Equity	-543.5014
End.Equity	4280.998

Trade Related Statistics

Segments tradeStats output to more cleanly display metrics related to trades only.

```
require(dplyr)
tab.trades <- tstats %>%
  mutate(Trades = Num.Trades,
         Win.Percent = Percent.Positive,
         Loss.Percent = Percent.Negative,
         WL.Ratio = Percent.Positive/Percent.Negative) %>%
  select(Trades, Win.Percent, Loss.Percent, WL.Ratio)
kable(t(tab.trades))
```

```
## Warning in kable_pipe(x = structure(c("Trades", "Win.Percent", "Loss.Percent", :
## The table should have a header (column names)
```

Trades	8.000000
Win.Percent	62.500000
Loss.Percent	37.500000
WL.Ratio	1.666667

Profit Related Statistics

Segments tradeStats output to more cleanly display metrics related to profit only.

```
tab.profit <- tstats %>%
  select(Net.Trading.PL, Gross.Profits, Gross.Losses, Profit.Factor)
kable(t(tab.profit))
```

	sym
Net.Trading.PL	4280.998300
Gross.Profits	5330.999100
Gross.Losses	-962.500700
Profit.Factor	5.538696

Averages

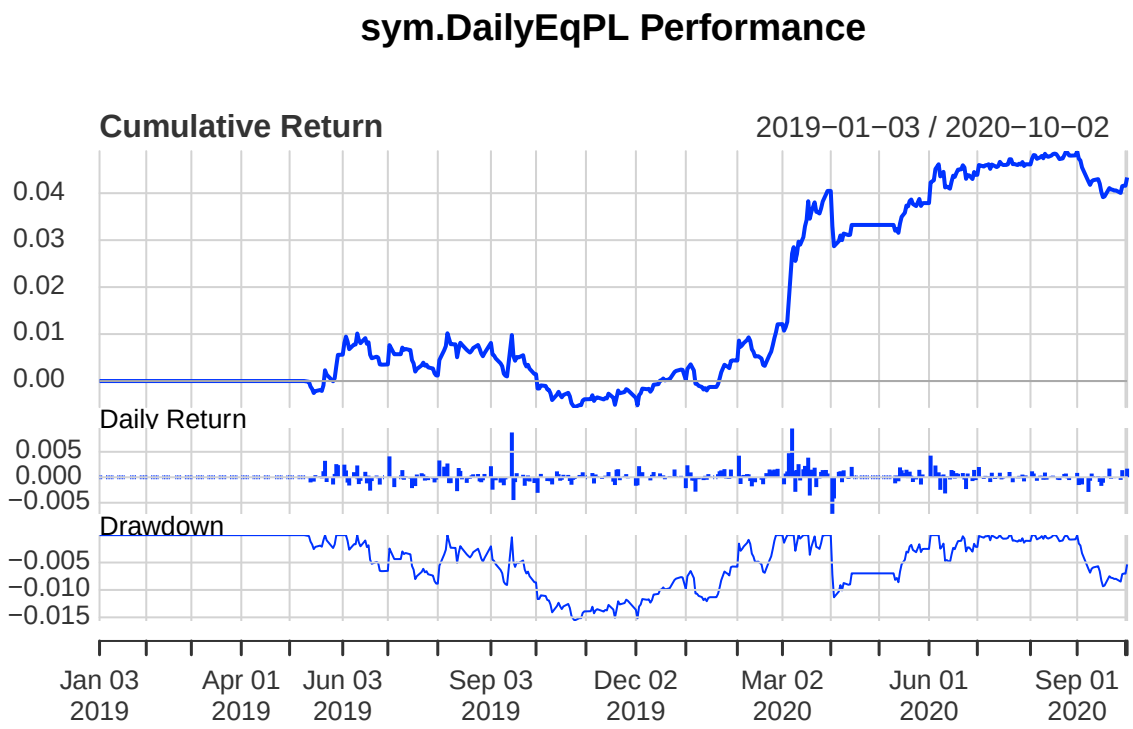
Segments tradeStats output to more cleanly display metrics related to averages only.

```
tab.wins <- tstats %>%
  select(Avg.Trade.PL, Avg.Win.Trade, Avg.Losing.Trade, Avg.WinLoss.Ratio)

kable(t(tab.wins))
```

	sym
Avg.Trade.PL	546.062300
Avg.Win.Trade	1066.199820
Avg.Losing.Trade	-320.833567
Avg.WinLoss.Ratio	3.323218

Performance Summary

```
rets <- PortfReturns(Account = account.st)
rownames(rets) <- NULL
charts.PerformanceSummary(rets, colorset = bluefocus)
```



Performance Statistics

```
tab.perf <- table.Arbitrary(rets,
                            metrics=c(
                              "Return.cumulative",
                              "Return.annualized",
                              "SharpeRatio.annualized",
                              "CalmarRatio"),
                            metricsNames=c(
                              "Cumulative Return",
                              "Annualized Return",
```

```
                                "Annualized Sharpe Ratio",
                                "Calmar Ratio"))
kable(tab.perf)
```

	sym.DailyEqPL
Cumulative Return	0.0433421
Annualized Return	0.0231094
Annualized Sharpe Ratio	1.1420455
Calmar Ratio	1.4836065

Risk Statistics

```
tab.risk <- table.Arbitrary(rets,
                            metrics=c(
                              "StdDev.annualized",
                              "maxDrawdown",
                              "VaR",
                              "ES"),
                            metricsNames=c(
                              "Annualized StdDev",
                              "Max DrawDown",
                              "Value-at-Risk",
                              "Conditional VaR"))
kable(tab.risk)
```

	sym.DailyEqPL
Annualized StdDev	0.0202351
Max DrawDown	0.0155765
Value-at-Risk	-0.0011569
Conditional VaR	-0.0011569

Order book

```
(ob <- getOrderBook(portfolio.st))
```

```
## $luxor
## $luxor$sym
##            Order.Qty Order.Price Order.Type  Order.Side Order.Threshold
## 2019-05-10 "-100"    "70.164998" "stoplimit" "short"    "-0.005"
## 2019-07-02 "all"     "62.400002" "market"    "short"    NA
## 2019-07-02 "100"     "65.505"    "stoplimit" "long"     "0.005"
## 2019-07-25 "all"     "63.389999" "market"    "long"     NA
## 2019-07-25 "-100"    "63.024999" "stoplimit" "short"    "-0.005"
## 2019-09-09 "all"     "62.59"     "market"    "short"    NA
## 2019-09-09 "100"     "62.995002" "stoplimit" "long"     "0.005"
## 2019-10-04 "all"     "58.369999" "market"    "long"     NA
## 2019-10-04 "-100"    "57.594998" "stoplimit" "short"    "-0.005"
## 2019-10-30 "all"     "60.610001" "market"    "short"    NA
## 2019-10-30 "100"     "61.915"    "stoplimit" "long"     "0.005"
## 2020-01-16 "all"     "64.620003" "market"    "long"     NA
```

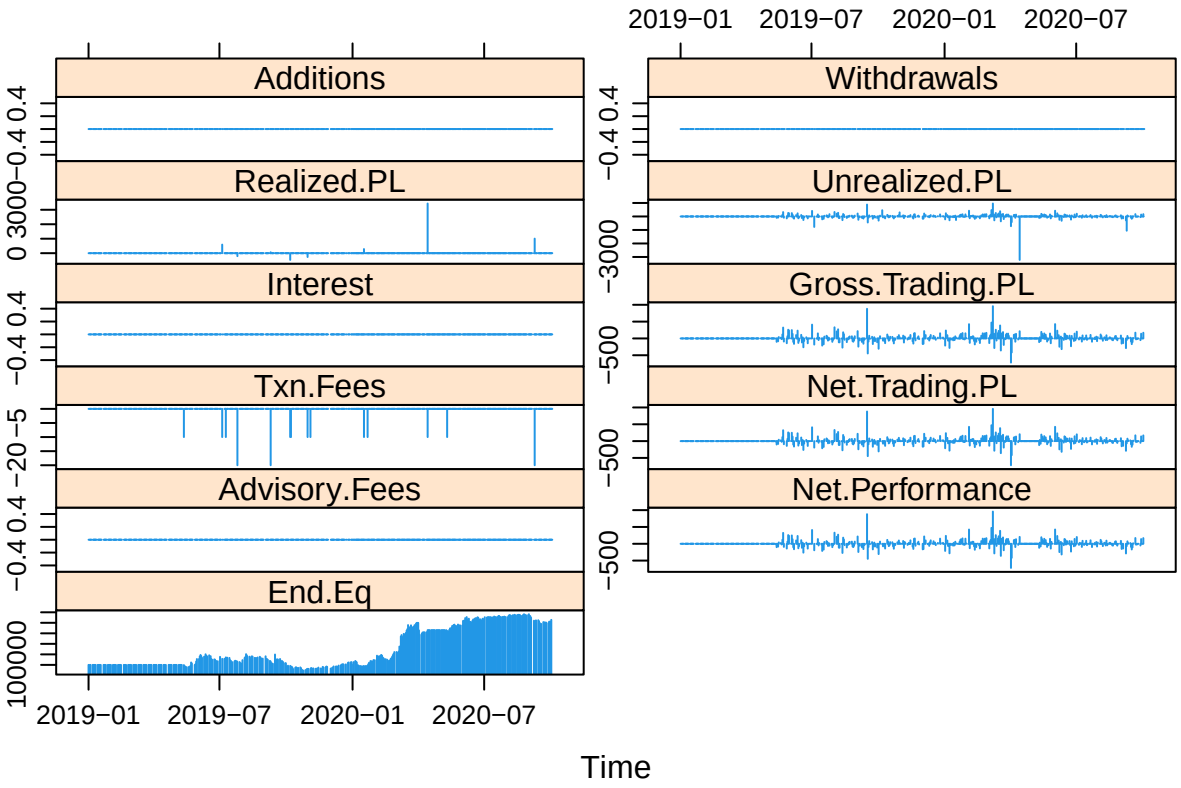
```
## 2020-01-16 "-100"    "63.914998" "stoplimit" "short"    "-0.005"
## 2020-04-13 "all"     "31.74"     "market"    "short"    NA
## 2020-04-13 "100"     "33.235"    "stoplimit" "long"     "0.005"
## 2020-04-22 "-100"    "15.995"    "stoplimit" "short"    "-0.005"
## 2020-05-10 "100"     "30.724999" "stoplimit" "long"     "0.005"
## 2020-09-08 "all"     "39.779999" "market"    "long"     NA
## 2020-09-08 "-100"    "39.294999" "stoplimit" "short"    "-0.005"
##            Order.Status Order.StatusTime      Prefer Order.Set Txn.Fees
## 2019-05-10 "closed"     "2019-05-13 00:00:00" "Low"  NA        "-10"
## 2019-07-02 "closed"     "2019-07-05 00:00:00" ""     NA        "-10"
## 2019-07-02 "closed"     "2019-07-10 00:00:00" "High" NA        "-10"
## 2019-07-25 "closed"     "2019-07-26 00:00:00" ""     NA        "-10"
## 2019-07-25 "closed"     "2019-07-26 00:00:00" "Low"  NA        "-10"
## 2019-09-09 "closed"     "2019-09-10 00:00:00" ""     NA        "-10"
## 2019-09-09 "closed"     "2019-09-10 00:00:00" "High" NA        "-10"
## 2019-10-04 "closed"     "2019-10-07 00:00:00" ""     NA        "-10"
## 2019-10-04 "closed"     "2019-10-08 00:00:00" "Low"  NA        "-10"
## 2019-10-30 "closed"     "2019-10-31 00:00:00" ""     NA        "-10"
## 2019-10-30 "closed"     "2019-11-04 00:00:00" "High" NA        "-10"
## 2020-01-16 "closed"     "2020-01-17 00:00:00" ""     NA        "-10"
## 2020-01-16 "closed"     "2020-01-22 00:00:00" "Low"  NA        "-10"
## 2020-04-13 "closed"     "2020-04-14 00:00:00" ""     NA        "-10"
## 2020-04-13 "replaced"   "2020-04-22"           "High" NA        "-10"
## 2020-04-22 "replaced"   "2020-05-10"           "Low"  NA        "-10"
## 2020-05-10 "closed"     "2020-05-11 00:00:00" "High" NA        "-10"
## 2020-09-08 "closed"     "2020-09-09 00:00:00" ""     NA        "-10"
## 2020-09-08 "closed"     "2020-09-09 00:00:00" "Low"  NA        "-10"
##            Rule         Time.In.Force
## 2019-05-10 "EnterSHORT" ""
## 2019-07-02 "Exit2LONG"  ""
## 2019-07-02 "EnterLONG"  ""
## 2019-07-25 "Exit2SHORT" ""
## 2019-07-25 "EnterSHORT" ""
## 2019-09-09 "Exit2LONG"  ""
## 2019-09-09 "EnterLONG"  ""
## 2019-10-04 "Exit2SHORT" ""
## 2019-10-04 "EnterSHORT" ""
## 2019-10-30 "Exit2LONG"  ""
## 2019-10-30 "EnterLONG"  ""
## 2020-01-16 "Exit2SHORT" ""
## 2020-01-16 "EnterSHORT" ""
## 2020-04-13 "Exit2LONG"  ""
## 2020-04-13 "EnterLONG"  ""
## 2020-04-22 "EnterSHORT" ""
## 2020-05-10 "EnterLONG"  ""
## 2020-09-08 "Exit2SHORT" ""
## 2020-09-08 "EnterSHORT" ""
##
##
## attr(,"class")
## [1] "order_book"
```

Account summary

```
require(lattice)
```

```
## Loading required package: lattice
```

```
a <- getAccount(account.st)
xyplot(a$summary, type = "h", col = 4)
```



Equity curve

```
equity <- a$summary$End.Eq
plot(equity, main = "Equity Curve")
```

Future Directions

This project enabled me to strengthen my familiarity with trading strategy scripting in R. The Luxor strategy was selected because of its simplicity and popularity among quantstrat tutorials. Future projects may focus on selecting strategies that target the specific features of Brent crude oil futures. One interesting project may involve testing a binary options trading strategy for BZ futures to capture volatility around inventory releases, with signals for inventory releases specified through the sigTimestamp argument.