Math 523
Rena Haswah
Professor Tier
Project 2
S&P 500 Index Analysis

July 16, 2019

1 Introduction

The S&P 500 is a collection of the leading 500 companies in the stock market. The following analysis will explore the histroical data of the S&P 500 for the first half of 2019. The stock does yield dividends, but that will not be included in this analysis. The current dividend yield of the index is about 2%. The treasury bill rate is close to 2.2%. By transformation, the data will show valuable properites and patterns.

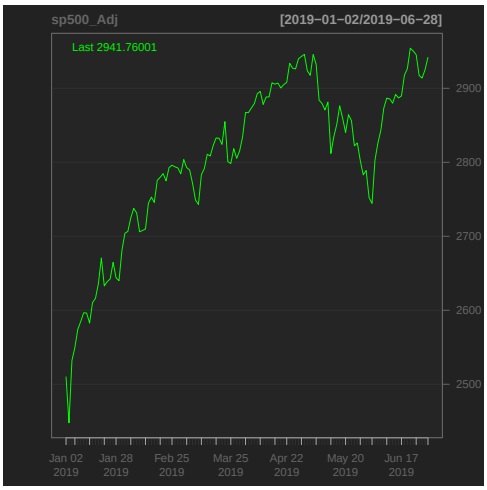
2 Historical Data Analysis

Following is an analysis of the historical S&P 500 data to test for stationarity.

```
suppressWarnings(suppressMessages(library(quantmod)))
suppressWarnings(suppressMessages(library(fBasics)))
library(quantmod)
library(fBasics)
sp500<-new.env()
options("getSymbols.yahoo.warning"=FALSE)
options("getSymbols.warning4.0"=FALSE)
#Retrieves S&P 500 Time Series from Yahoo
getSymbols("^GSPC",env=sp500,src="yahoo",
           from=as.Date("2019-01-01"),
           to =as.Date("2019-07-01"))

## [1] "^GSPC"

GSPC=sp500$GSPC
```

Below is a plot of the adjusted closing values of the S&P 500 over the first half of the year 2019. It looks to be growing upward without any particular bounds. This is great news for investors of the index, but a nuisance for statisticians. This plot is not stationary, but there are certain transformations that can make the data better to work with.

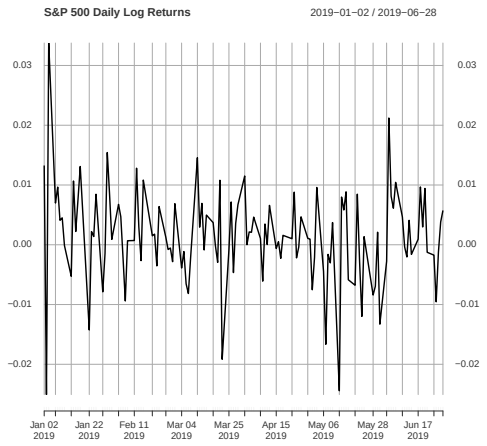
```
sp500_Adj<-GSPC[,6]
chartSeries(sp500_Adj)
```



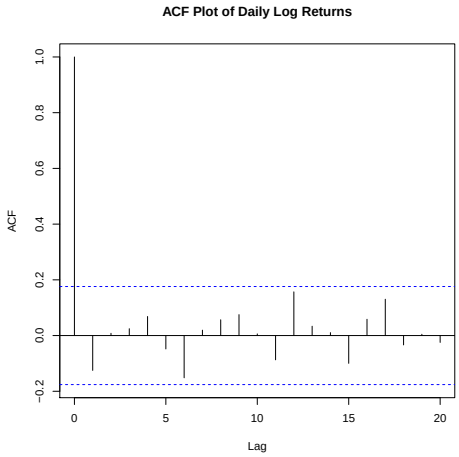
From the above, it is observed that the index does not follow any useful statistical patterns, and appears to be non-stationary as it heads in an upward trajectory. Common practice is to transform the data to the natural log of the daily returns in the hopes that it will look more statioanry.

Stationarity is observed below as the plot of the log returns is mean reverting, and the auto-correlation, ACF in R, goes to zero very quickly. .

```
daily=dailyReturn(GSPC,type='log')
plot(daily,type='l',ylab="Daily Log Returns",
     main="S&P 500 Daily Log Returns")
```



```
acf(daily,main="ACF Plot of Daily Log Returns")
```



The basic statistics also show a constant mean and variance can be computed

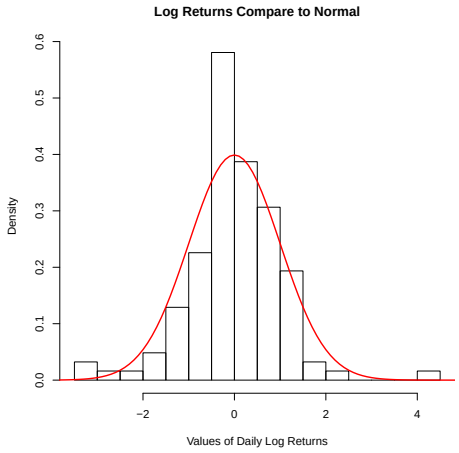
```
basicStats(daily)

##             daily.returns
## nobs           124.000000
## NAs              0.000000
## Minimum         -0.025068
## Maximum          0.033759
## 1. Quartile     -0.002202
## 3. Quartile      0.006440
## Mean             0.001387
## Median           0.001151
## Sum              0.171976
## SE Mean          0.000717
## LCL Mean        -0.000032
## UCL Mean         0.002805
## Variance         0.000064
## Stdev            0.007980
## Skewness        -0.115323
## Kurtosis         2.782268
```

Generate a set of points from N(0,1) to be used later in a comparison between the log transformed data and an actual normal density curve.

```
x<- seq(-10,10,0.1)
y <- dnorm(x,0,1)
```

Standardize the daily log return data for better scaling.

```
standaily<-(daily-mean(daily))/sd(daily)
hist(standaily,breaks=11,freq=FALSE,
     xlab="Values of Daily Log Returns",
     main="Log Returns Compare to Normal")
par(new=TRUE)
lines(x=x,y=y,col="red",lwd=2)
```




```
sigma=sd(daily)*sqrt(252)
sigma

## [1] 0.126677
```

3 Trading Options

In finance, an option is a contract that allows an investor to buy or sell an underlying asset. It can be thought of as a bet. This analysis will only consider European trading options, which can only be exercised at a specified expiration time. These trading options have two main positions.

The put option is a stock market device which gives the owner the right, but not the obligation, to sell an asset to the seller of the put at a predetermined date, the "expiry",at a specified price, known as the strike price. This option is quite risky for the seller of the put, but the purchaser's risk is limited to the price of the put and their payoff is unlimited.

The call option gives the buyer the right, but not the obligation, to buy an

underlying asset from the seller of the option at a certain time (the expiration date) for a certain price (the strike price). The payoff outlook for a call is also better for the buyer. The seller is under a lot of risk with options.

Getting into the mathematics, these options must follow the put-call parity. Assuming that the put and call both have the same strike price and time to expiry, the portfolio of a long call option and a short put option is equivalent to a single forward contract. The following equation mathematically describes the Put-Call Parity:

$$c(s,t) - p(s,t) = s - Ke^{-r(T-t)}$$

where
c(t) is the value of the call at time t,
p(t) is the value of the put of the same expiration date,
s(t) is the spot price of the underlying asset,
K is the strike price, and
$e^{-r(T-t)}$ is the present value of a zero-coupon bond that matures at time T.

Following is an experiment testing the Put-Call Parity on the S&P 500 index. Some discrepancies are observed as the put-call parity fails to perfectly equate with real world data, but it definitely comes close.

```
PCparity <- data.frame("s" = c(2994.73,2999.91,2999.91,2990.90),
                "K" = c(2675,2750,2800,3000),
                "c" = c(413.98,245.57,205.60,106),
                "p"=c(0.15,0.35,7.69,106.05),
                "Texp"=c(6/252,6/252,6/252,113/252),
                "LHSparity"=c(413.81,245.22,197.91,0.05),
                "RHSparity"=c(321.13,251.30,201.38,20.35))
PCparity
```

```
##         s    K      c      p       Texp LHSparity RHSparity
## 1 2994.73 2675 413.98   0.15 0.02380952    413.81    321.13
## 2 2999.91 2750 245.57   0.35 0.02380952    245.22    251.30
## 3 2999.91 2800 205.60   7.69 0.02380952    197.91    201.38
## 4 2990.90 3000 106.00 106.05 0.44841270      0.05     20.35
```

3.1 Black-Scholes-Merton Pricing Formula

The Black-Scholes-Merton (BSM) pricing formula is the most calculated mathematical equation in the world on a day-to-day basis. The formula is derived from the BSM partial differential equation, and is used frequently in option pricing as well as for other derivative pricing and risk management. T1 is representing expiry time, since R has built in capital T.

```
bsmcall<-function(s,t,k,r,T1,sigma)
{
  a=log(s/k)
  b=(r+sigma^2/2)*(T1-t)
  c=sigma*sqrt(T1-t)
  d1=(a+b)/c
  d2=d1-c
  bcall=s*pnorm(d1)-k*exp(-r*(T1-t))*pnorm(d2)
  print(bcall)
}
bsmput<-function(s,t,k,r,T1,sigma)
{
  a=log(s/k)
  b=(r+sigma^2/2)*(T1-t)
  c=sigma*sqrt(T1-t)
  d1=(a+b)/c
  d2=d1-c
  bput=k*exp(-r*(T1-t))*pnorm(-d2)-s*pnorm(-d1)
  print(bput)
}
```

With the put and call functions defined with BSM, real world data can be used to test the validity of the model.

```
options(scipen = 999)
a=bsmcall(s=2994.73,t=0,k=2675,r=.022,T1=6/252,sigma)

## [1] 321.1308

b=bsmcall(s=2999.91,t=0,k=2750,r=.022,T1=6/252,sigma)

## [1] 251.3501

c=bsmcall(s=2999.91,t=0,k=2800,r=.022,T1=6/252,sigma)

## [1] 201.3789

d=bsmcall(s=2990,t=0,k=3000,r=.022,T1=113/252,sigma)

## [1] 110.849

callBSM<-c(a,b,c,d)
callActual<-PCparity$c
callCompare<-data.frame(callBSM,callActual)

h=bsmput(s=2994.73,t=0,k=2675,r=.022,T1=6/252,sigma)
```

```
## [1] 0.0000000294322

i=bsmcall(s=2999.91,t=0,k=2750,r=.022,T1=6/252,sigma)

## [1] 251.3501

j=bsmcall(s=2999.91,t=0,k=2800,r=.022,T1=6/252,sigma)

## [1] 201.3789

k=bsmcall(s=2990,t=0,k=3000,r=.022,T1=113/252,sigma)

## [1] 110.849

putBSM<-c(h,i,j,k)
putActual<-PCparity$p
putCompare<-data.frame(putBSM,putActual)
```

Aside from the very first value, the BSM formula performed pretty well in computing the call.

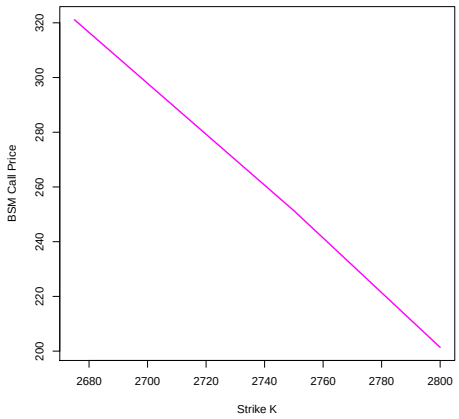
```
callCompare

##    callBSM callActual
## 1 321.1308     413.98
## 2 251.3501     245.57
## 3 201.3789     205.60
## 4 110.8490     106.00
```

The first and last values are reasonably close, but the second and third have severe miscalculations, which could have something to do with setting the dividends equal to zero.

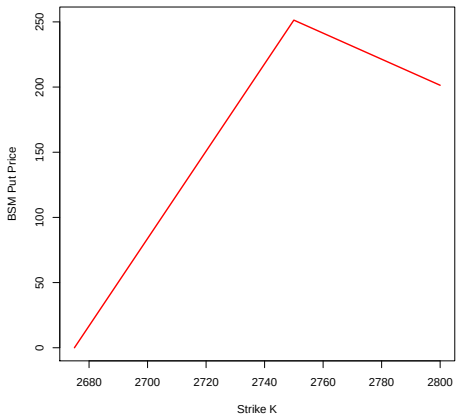
```
putCompare

##              putBSM putActual
## 1   0.0000000294322      0.15
## 2 251.3501426412804      0.35
## 3 201.3789411210851      7.69
## 4 110.8490173749597    106.05
```

The first three values in the data frame happen to have the same value for T-t, therefore those three will go into the plots. The call prices for BSM show a perfect linear relationship. With the previous issues from the puts, the plot is not perfect, as was expected.

```
Strike<-PCparity$K
strk<-Strike[1:3]
plot(strk,callBSM[1:3],xlab="Strike K",ylab="BSM Call Price",type="l",lwd=2,col="magenta")
```

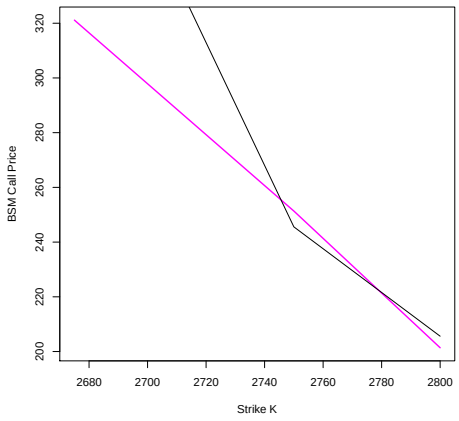



```
plot(strk,putBSM[1:3],xlab="Strike K",ylab="BSM Put Price",type="l",lwd=2,col="red")
```

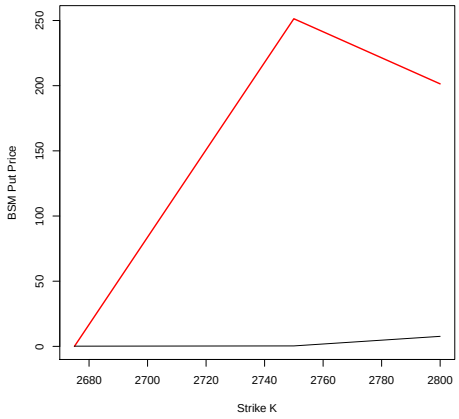


Comparing the BSM to actual prices

```
plot(strk,callBSM[1:3],xlab="Strike K",ylab="BSM Call Price",type="l",lwd=2,col="magenta")
lines(strk,callActual[1:3],type="l",add=TRUE)

## Warning in plot.xy(xy.coords(x, y), type = type, ...):  "add" is
not a graphical parameter
```




```
plot(strk,putBSM[1:3],xlab="Strike K",ylab="BSM Put Price",type="l",lwd=2,col="red")
lines(strk,putActual[1:3],type="l",add=TRUE)

## Warning in plot.xy(xy.coords(x, y), type = type, ...):  "add" is
not a graphical parameter
```



4 Conclusion

In conclusion, the log returns of the S&P 500 prices appear to satisfy the conditions for stationarity. The Black-Scholes-Merton equation may not be the best approximator, but did reasonably okay for the call options. The put call parity held up well against the real data provided.