STAT 443: Assignment 1

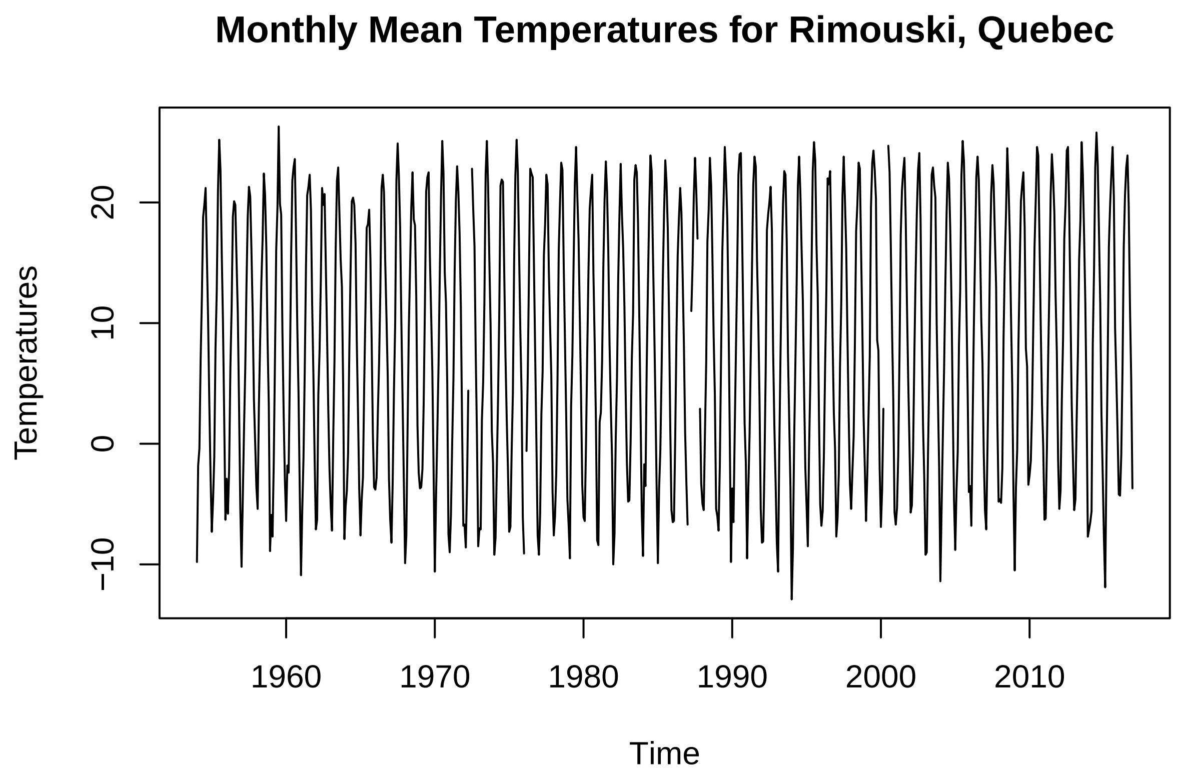
Saksham Sudershan (Student #31339427)

31 January, 2022

Question 1

(a)

```r
# Reading data
dat1 <- read.csv("Data1.csv")

# Creating time series object and plotting
ts1 <- ts(dat1$Mean.Max.Temp..Â.C., start = c(1954,1), frequency = 12)
plot(ts1, main = "Monthly Mean Temperatures for Rimouski, Quebec",
     xlab = "Time", ylab = "Temperatures")
```



From the graph here, it is quite difficult to discern if there is a trend or not. The series is non-stationary and there is a seasonal effect which appears to have a period of 1 year or 12 months. The amplitude of the peaks and troughs doesn't appear to be increasing, and so an additive model would be more suitable to decompose the series.

(b)

```r
# Checking the number of missing values
sum(is.na(dat1$Mean.Max.Temp..Â.C.))
```

```
## [1] 9
```

The number of missing values is 9.

```r
# Printing Year-Month
for (i in 1:nrow(dat1)) {
  if(is.na(dat1[i,8]))
    print(dat1[i,5])
}
```

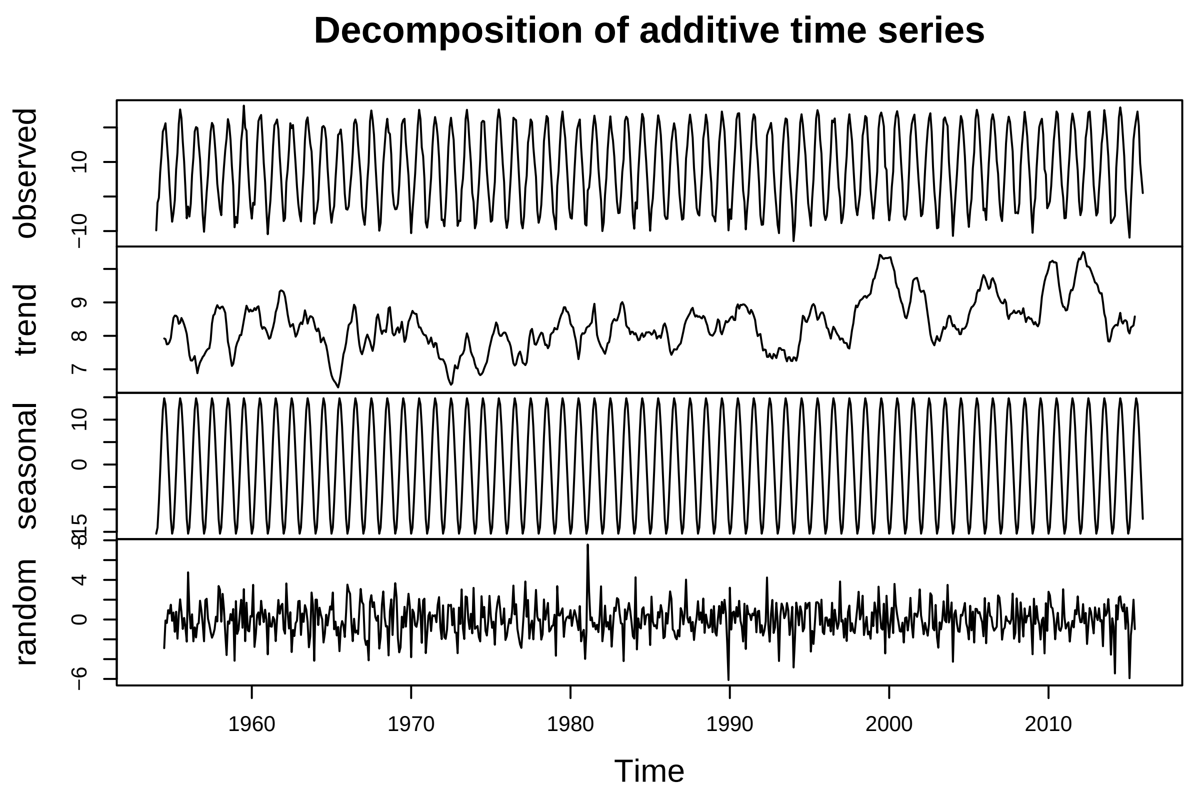
```
## [1] "1972-05"
## [1] "1972-06"
## [1] "1976-02"
## [1] "1987-02"
## [1] "1987-03"
## [1] "1987-10"
## [1] "2000-04"
## [1] "2000-05"
## [1] "2000-06"
```

LVCF is not appropriate to use with this data set as there is seasonality in the time series. An adapted version of LVCF that can be used with this data set is to carry forward the value from the same month in the previous year.

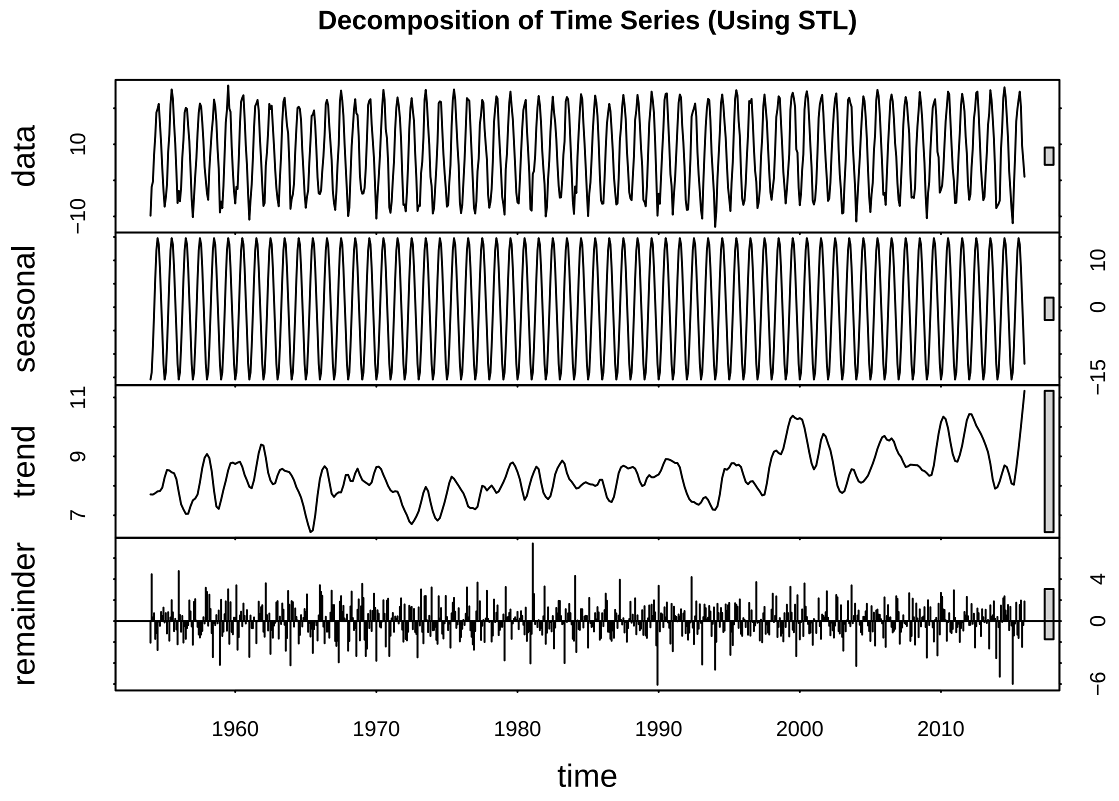
```r
# Applying the imputation
for (i in 1:length(ts1)) {
  if(is.na(ts1[i])){
    ts1[i] <- ts1[i-12]
    dat1[i,8] <- dat1[i-12,8]
  }
}
```

(c)

```r
# Splitting the time series into training and test data sets
traints <- window(ts1, start = c(1954,1), end = c(2015,12))
testts <- window(ts1, start = c(2016,1), end = c(2016,12))

# Decomposing training data set using moving average method and plotting
decomp1 <- decompose(traints)
plot(decomp1)
```



```
# Decomposing training set using Loess method and plotting
decomp2 <- stl(traints, s.window = "periodic")
plot(decomp2, main = "Decomposition of Time Series (Using STL)")
```



(d)

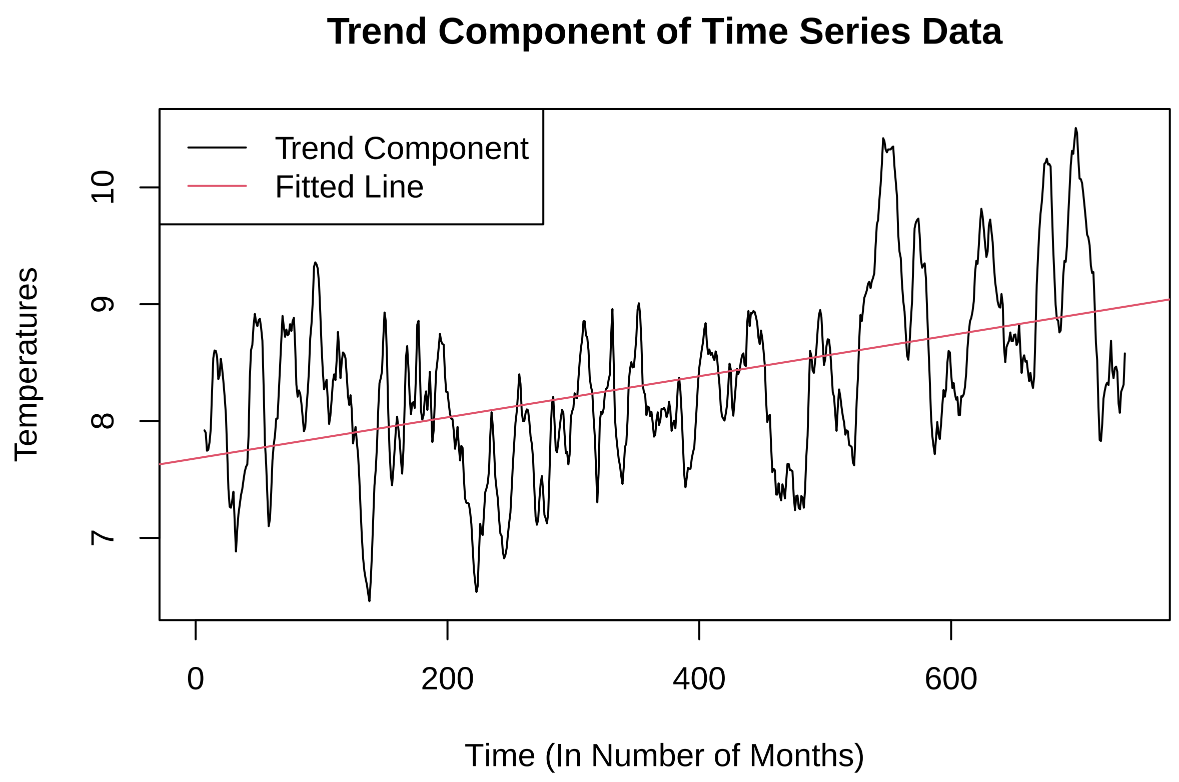
```
# Creating data set with same time length as training time series
dat2 <- data.frame()
for(i in 1:744){
  dat2[i,1] <- i
  dat2[i,2] <- dat1[i,8]
}
# Fitting into linear regression
reg <- lm(decomp1$trend~dat2$V1)

# Summary of regression
summary(reg)
```

```
##
## Call:
## lm(formula = decomp1$trend ~ dat2$V1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.53499 -0.46119 -0.02852  0.46585  1.78017
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 7.6802144  0.0502146   152.9   <2e-16 ***
## dat2$V1     0.0017591  0.0001173    15.0   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6703 on 730 degrees of freedom
##   (12 observations deleted due to missingness)
## Multiple R-squared:  0.2357, Adjusted R-squared:  0.2346
## F-statistic: 225.1 on 1 and 730 DF,  p-value: < 2.2e-16
```

```
# Calculating 95% CI
CI <- c(0.0017591-1.96*0.0001173,0.0017591+1.96*0.0001173)
CI
```

```
## [1] 0.001529192 0.001989008
```

```
# Plotting the trend
plot(dat2$V1,decomp1$trend, type = "l",
     xlab = "Time (In Number of Months)", ylab = "Temperatures", main = "Trend Component of Time Series
abline(a = 7.6802144, b = 0.0017591, col=2)
legend("topleft",legend=c("Trend Component","Fitted Line"), lty=1,col=c(1,2))
```

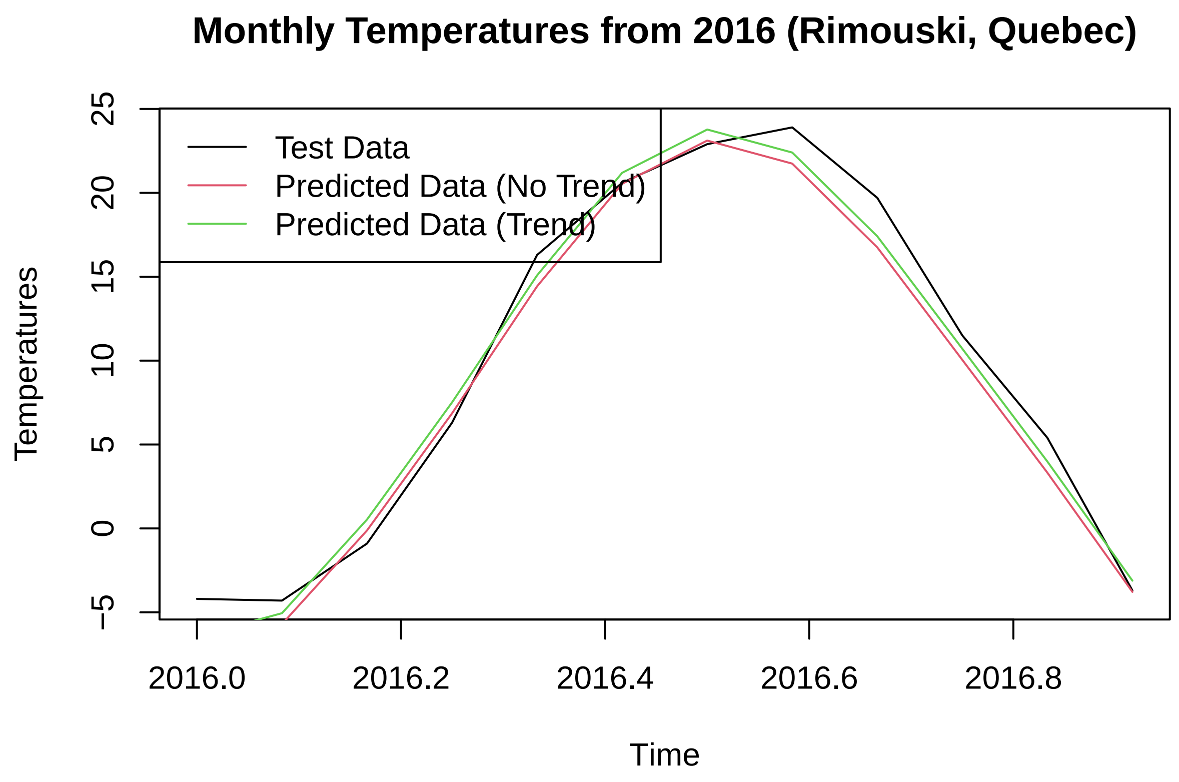


The 95% Confidence Interval is $[0.001529192, 0.001989008]$. Since this does not contain 0, we can confidently say that there is an increasing trend.

As there is a more or less a constant increasing trend in the data as seen from the graph, we can use the linear model of trend to make predictions.

(e)

```r
# Predicting monthly max temperatures for test data set
# With no trend
pred1 <- ts(mean(traints-decomp1$seasonal)+decomp1$seasonal[1:12], start = c(2016,1), end = c(2016,12),

# With trend
pred2_dat <- data.frame()

for (j in 1:12) {
  pred2_dat[j,1] = (7.6802144 + (0.0017591)*(j+744)+decomp1$seasonal[j])
}
pred2 <- ts(pred2_dat$V1, start = c(2016,1), end = c(2016,12), frequency = 12)

# Plotting both predictions and the test data set
plot(testts, xlab = "Time", ylab = "Temperatures",
     main = "Monthly Temperatures from 2016 (Rimouski, Quebec)")
lines(pred1, type = "l", col=2)
lines(pred2, type = "l", col=3)
legend("topleft",legend=c("Test Data","Predicted Data (No Trend)",
                          "Predicted Data (Trend)"), lty=1,col=c(1,2,3))
```



```r
# Calculating the MSPEs for both predictions
MSPE_calc_dset <- data.frame()
```

```r
for(i in 1:12){
  MSPE_calc_dset[i,1] <- (pred1[i]-testts[i])^2
  MSPE_calc_dset[i,2] <- (pred2[i]-testts[i])^2
}

MSPE_1 <- mean(MSPE_calc_dset$V1)
MSPE_2 <- mean(MSPE_calc_dset$V2)
MSPE_1
```

```
## [1] 2.876668
```

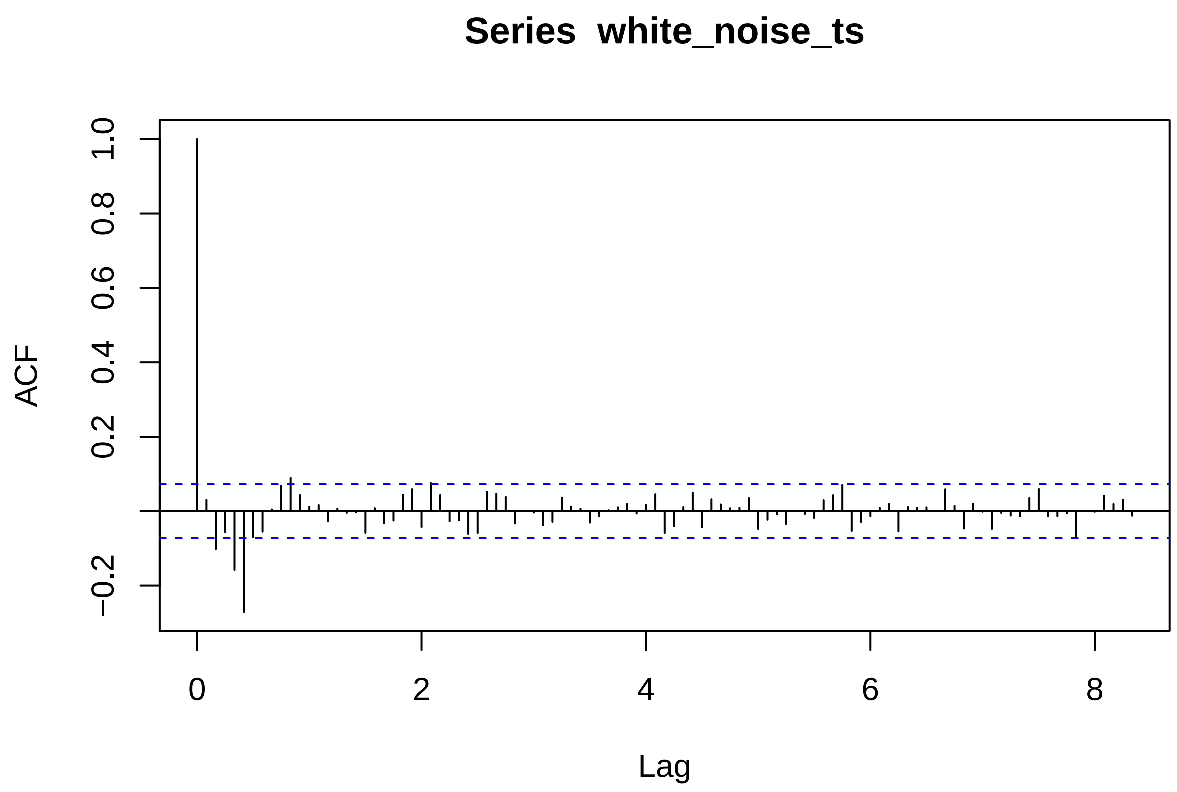
```r
MSPE_2
```

```
## [1] 1.841399
```

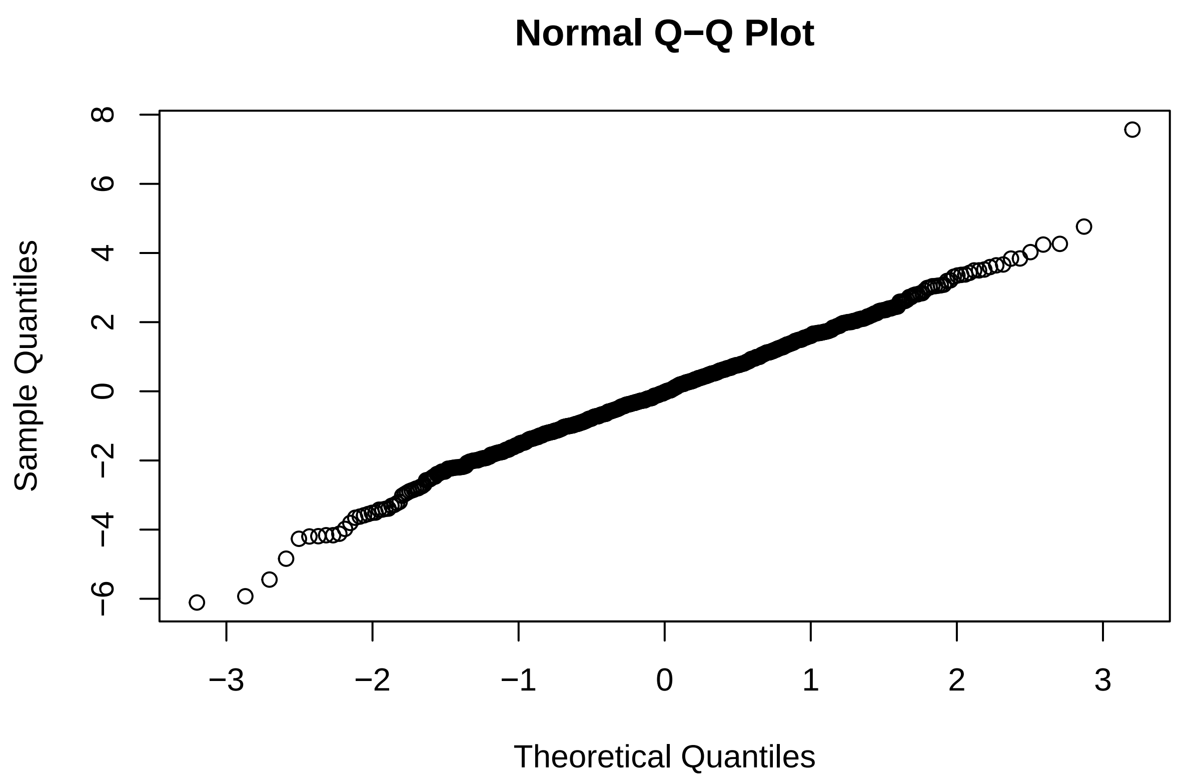
The MSPE for the prediction when it is assumed that there is no trend is 2.876668, while the MSPE for the prediction when it is assumed that there is a trend is 1.841399. Thus, the model where it is assumed that there is a trend has a smaller MSPE.

(f)

```r
holder_1 <- decomp1$random
white_noise_ts <- na.omit(holder_1)
acf(white_noise_ts, lag.max = 100)
```



```
qqnorm(white_noise_ts)
```

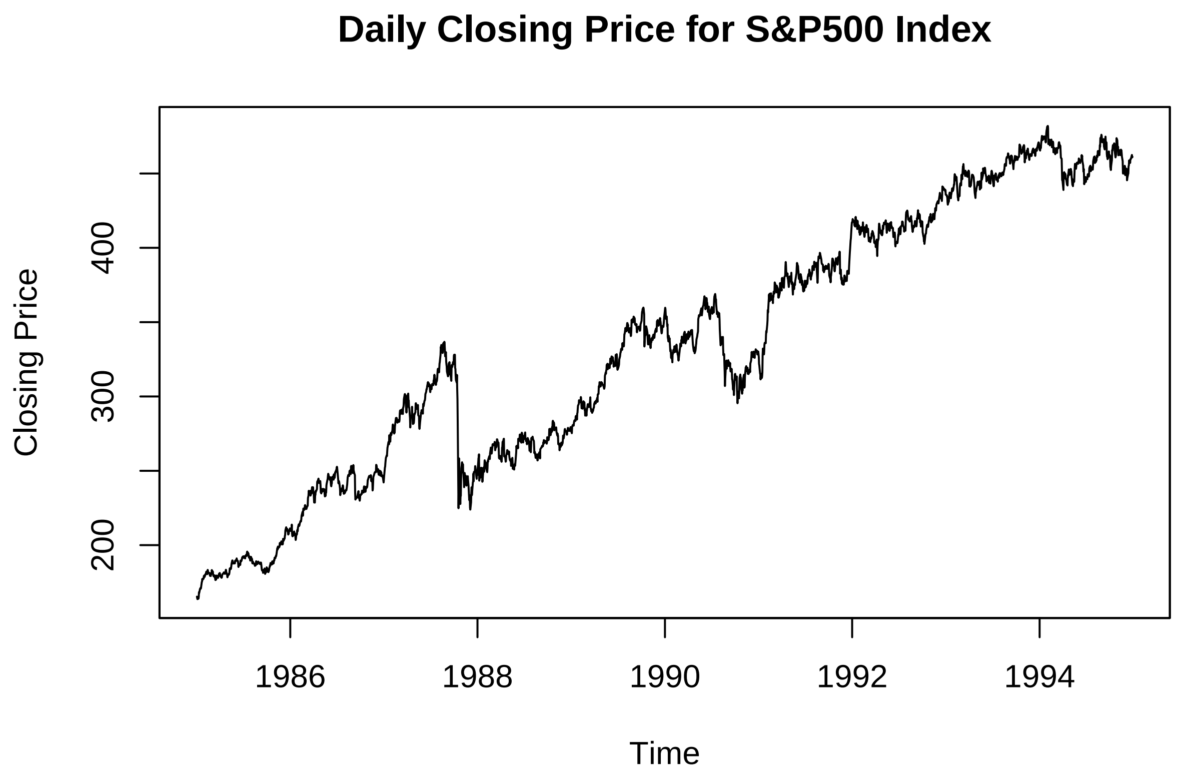


The error term seems to be from a normal distribution according to the Q-Q plot and most of the ACF values also lie within the confidence limits. Thus, it can be said that the Gaussian white noise assumption is appropriate here.

Question 2

(a)

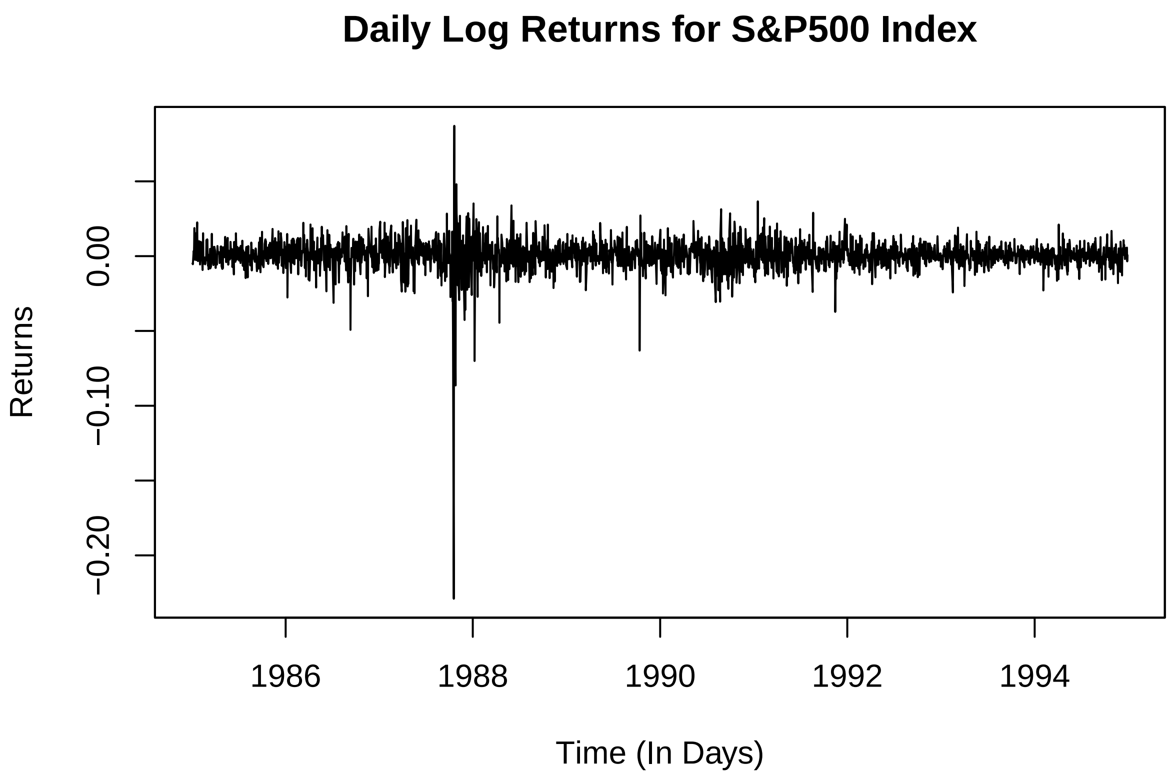
```
# Reading data and making time series object
dat3 <- read.csv("GSPC.csv")
ts2<- zoo(x=dat3$GSPC.Close, order.by = strptime(dat3$Date,format = c("%Y-%m-%d")))
# Plotting
plot(ts2, xlab = "Time", ylab = "Closing Price", main =
       "Daily Closing Price for S&P500 Index")
```



There is an overall increasing trend in the data. However, there is no discernible seasonality that can be seen from the above graph. The time series is not stationary.

(b)

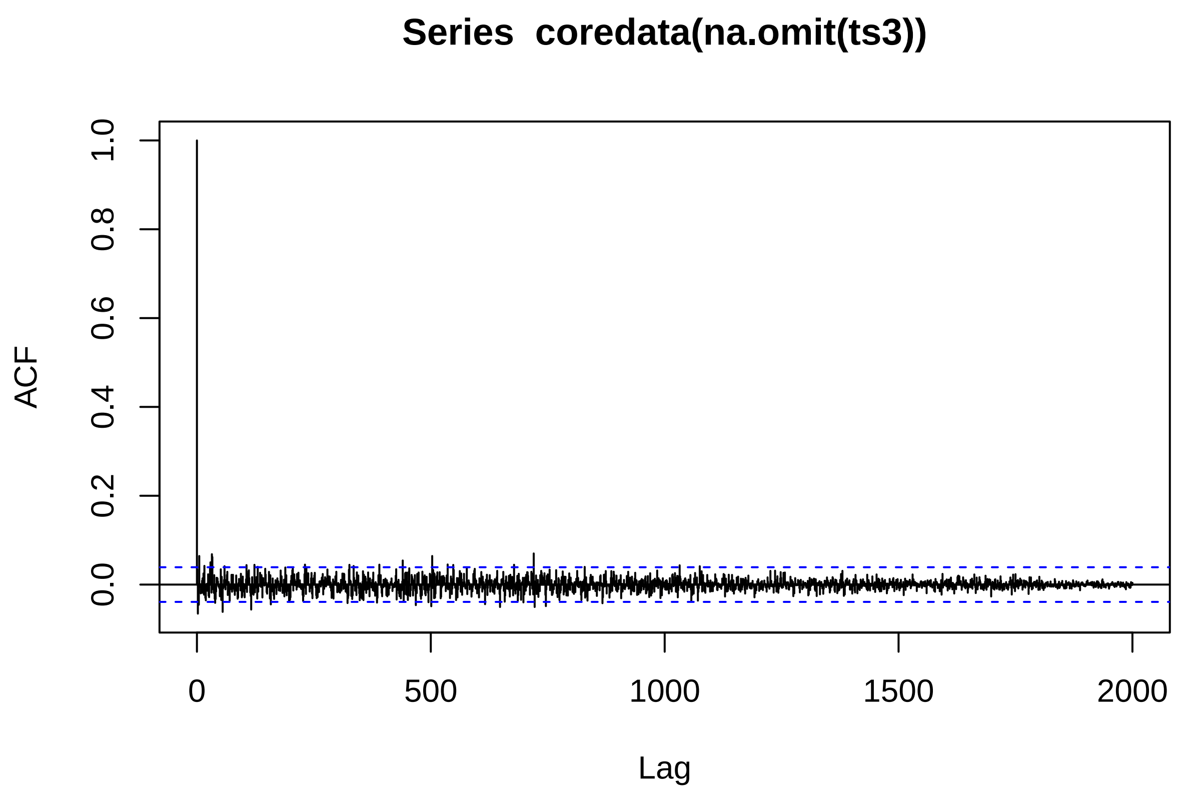
```r
# Creating log returns data set and time series
daily_log_returns <- data.frame()
for(i in 2:nrow(dat3)){
    daily_log_returns[i,1] <- log(dat3[i,5]/dat3[i-1,5])
}
ts3 <- ts(daily_log_returns$V1)
ts3<- zoo(x=daily_log_returns$V1, order.by = strptime(dat3$Date,format = c("%Y-%m-%d")))
# Plotting
plot(ts3, xlab = "Time (In Days)", ylab = "Returns", main =
       "Daily Log Returns for S&P500 Index")
```



The daily log returns time series is stationary. It is more convenient to work with stationary data as the properties of the model do not depend upon the time at which it is observed, and thus, it is much easier to predict future values taking into account the past values.

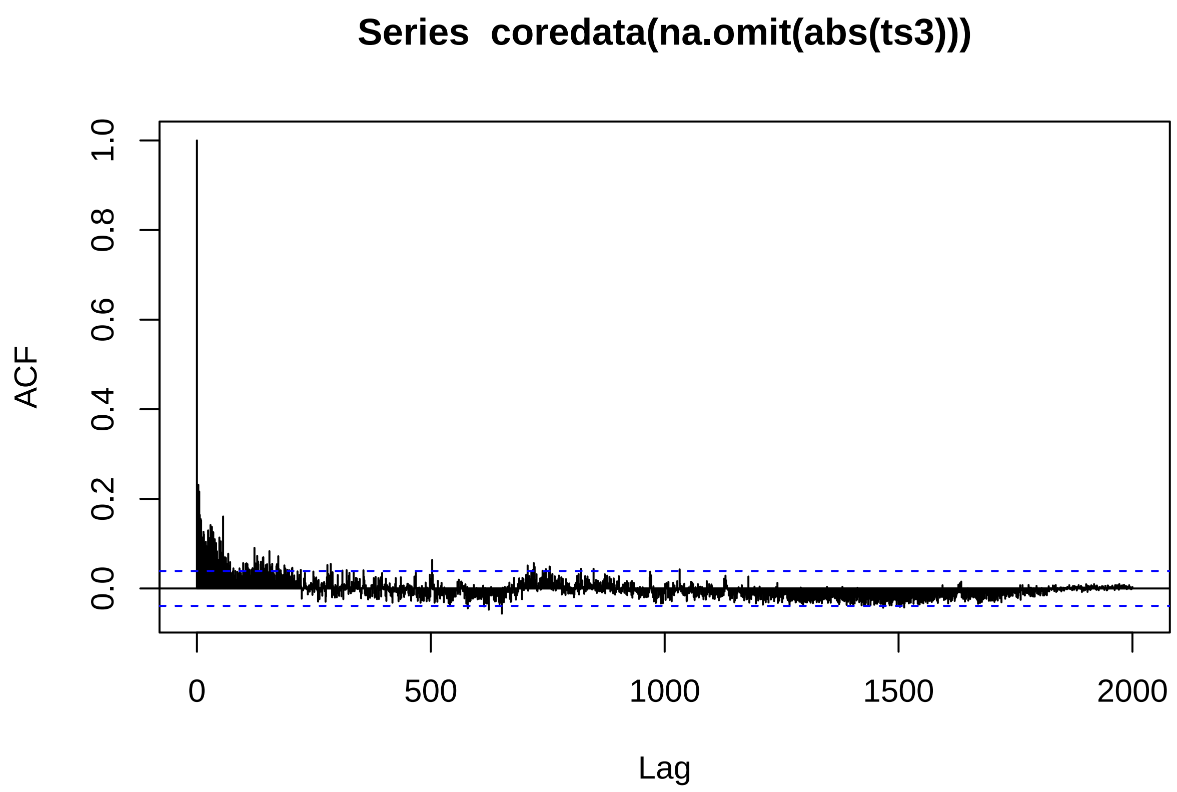
(c)

```
# Plotting ACF
acf(coredata(na.omit(ts3)), lag.max = 2000)
```



The autocorrelation function graph for log returns looks like that of a white noise process, with few autocorrelations falling outside the 95% confidence limits. This might be due to the stochastic nature of the returns.

```r
# Plotting ACF for absolute value series
acf(coredata(na.omit(abs(ts3))), lag.max = 2000)
```



The autocorrelation function graph for absolute value of log returns looks to decreasing at the start and then oscillates between negative and positive. Most of the acf at lag $h = 100$ or greater fall within the confidence limits and are not significant, however the acf values before those are significant.