# ISYE 6402 Homework Spring 2024

## Part 1: EXT FUNDS Gold Price Exchange Analysis

## Background

In this problem, we will study fluctuations in The NEXT FUNDS Gold Price Exchange Traded Fund that is a type of investment fund that aims to track the performance of gold prices. By investing in this fund, investors can gain exposure to the price movements of gold without having to physically own the metal. The fund holds physical gold as its underlying asset, and its value is based on the market price of gold. You will use the file Fund Prices Data.csv, where monthly prices are from January 2010 to Dec 2022.

### Instructions on reading the data

To read the data in R, save the file in your working directory (make sure you have changed the directory if different from the R working directory) and read the data using the R function read.csv()

You will perform the analysis and modelling on the Close data column.

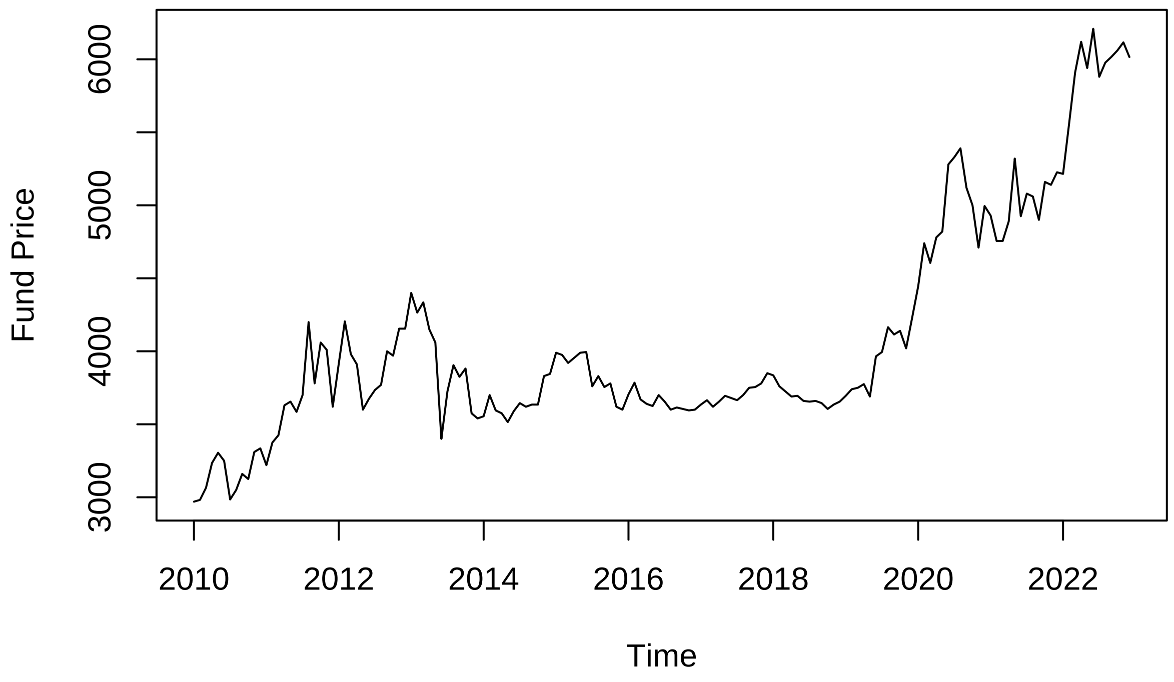
```r
#Here are the libraries you will need:

library(mgcv)
library(TSA)
library(dynlm)
library(ggplot2)
library(reshape2)
library(greybox)
library(mlr)
library(mgcv)
library(lubridate)
library(dplyr)
library(data.table)
library(aTSA)
```

```r
#Run the following code to prepare the data for analysis:

data <- read.csv("Fund Prices Data-1.csv")
```

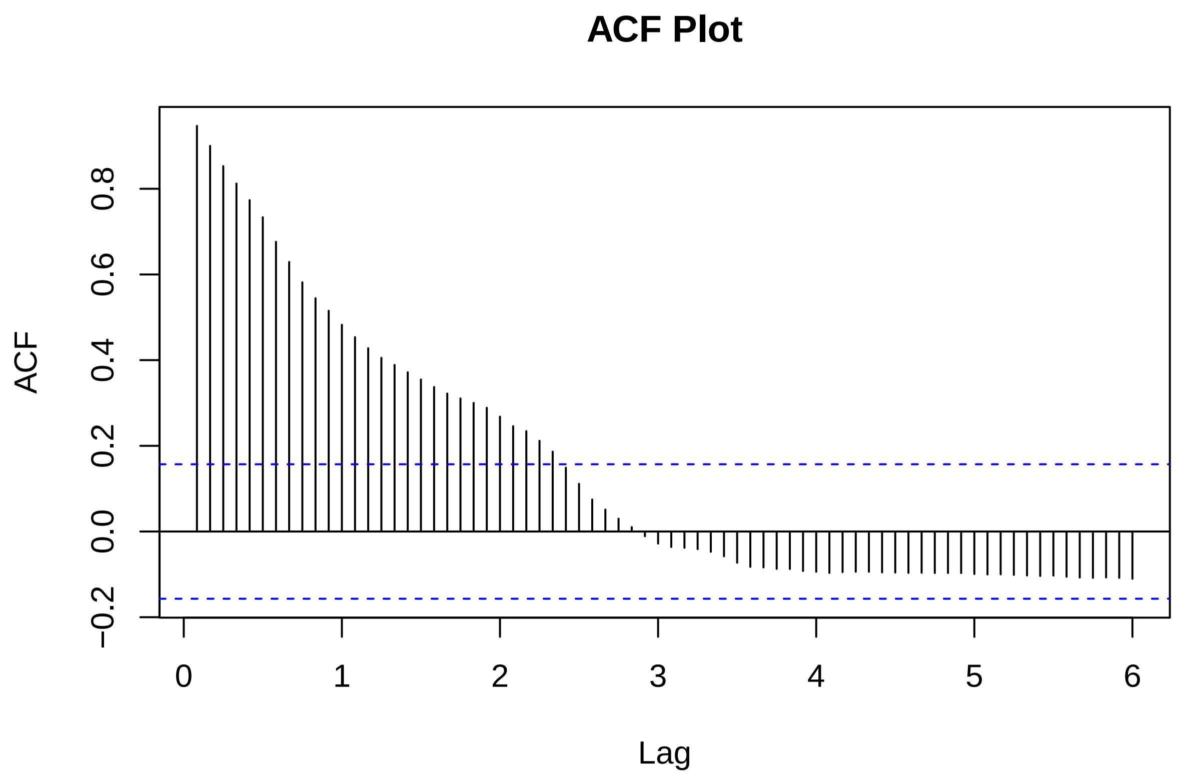
### Question 1a: Exploratory Data Analysis

Plot the Time Series and the ACF plot for the series. Comment on the stationarity of both time series based on these plots. Which (if any) stationarity assumptions are violated for the time series?

```
data$Date <- as.Date(data$Date, format = "%m/%d/%y")
#fund_prices <- as.vector(t(data))
fund_prices <- ts(data$Close, start = c(2010, 1), frequency = 12)
ts.plot(fund_prices,ylab="Fund Price")
```



```
acf_plot <- acf(fund_prices, lag.max = 72, main="ACF Plot")
```



**Response: 1a**

It appears that the autocorrelation decreases very slowly (more than two years, or 24 months) until it goes below the upper bound for the 95% confidence interval, which indicates a trend. Based on this, the ACF plot suggests that the first and third stationarity assumptions are not met. In addition, when inspecting the first plot of the time series, the second condition of finite variability does not seem to be met, since at the beginning, there is a significant increase in variance, then there's little variance from 2014-2019, and then the variance increases significantly.
## Question 1b: Trend Estimation

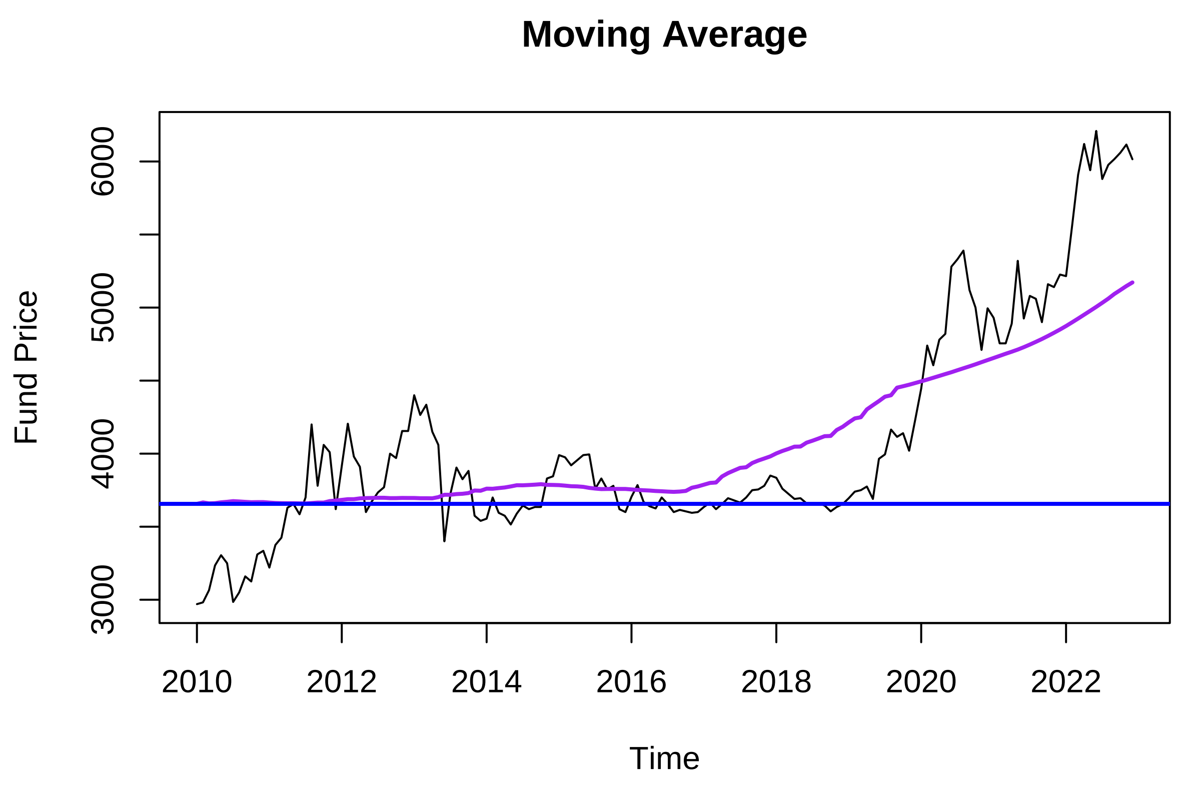
Fit the following trend estimation models:

- Moving average
- Parametric quadratic polynomial
- Local Polynomial
- Splines

Overlay the fitted values derived from each trend estimation model on the corresponding data. Comment on the effectiveness of each model to estimate the trend for the series.

```r
## X-axis points converted to 0-1 scale, common in nonparametric regression
time.pts = c(1:length(fund_prices))
time.pts = c(time.pts - min(time.pts))/max(time.pts)

## Fit a moving average
mav.fit = ksmooth(time.pts, fund_prices, kernel = "box")
fund_prices.fit.mav = ts(mav.fit$y,start=c(2010, 1),frequency=12)
```

```r
ts.plot(fund_prices,ylab="Fund Price", main="Moving Average")
lines(fund_prices.fit.mav,lwd=2,col="purple")
abline(fund_prices.fit.mav[1],0,lwd=2,col="blue")
```

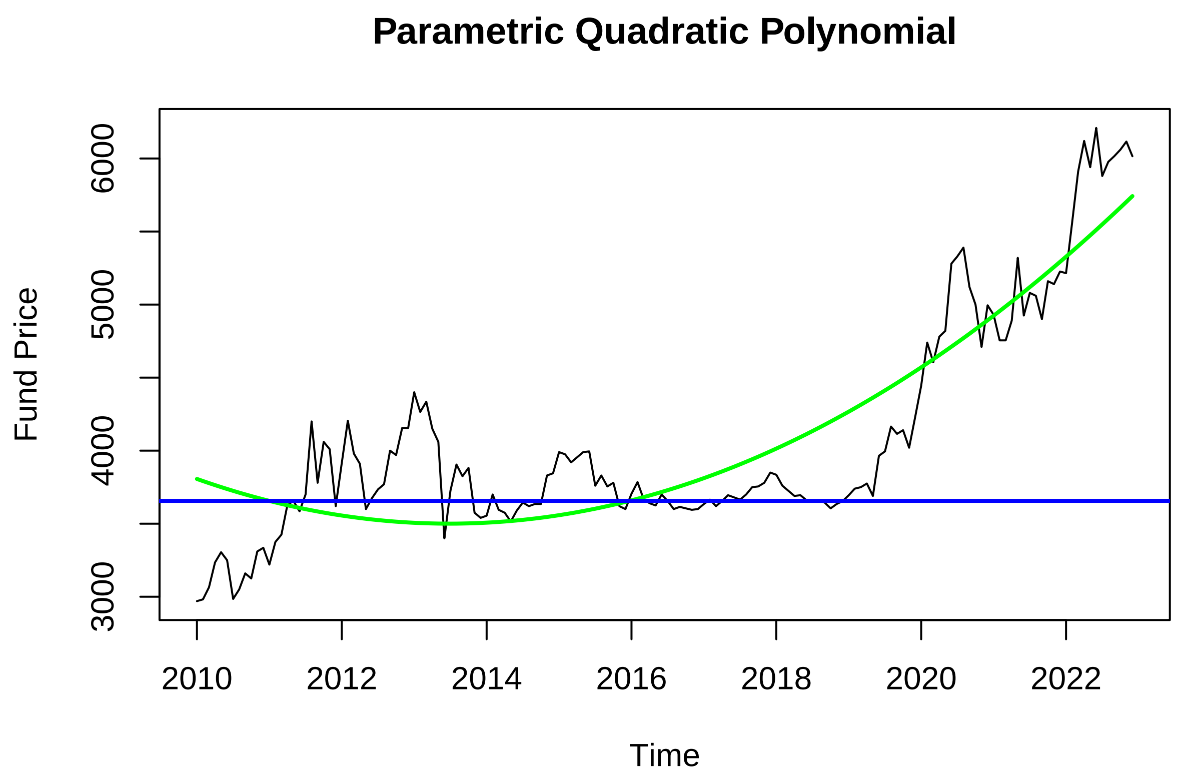


```r
## Fit a parametric quadraric polynomial
x1 = time.pts
x2 = time.pts^2
lm.fit = lm(fund_prices~x1+x2)
summary(lm.fit)
```

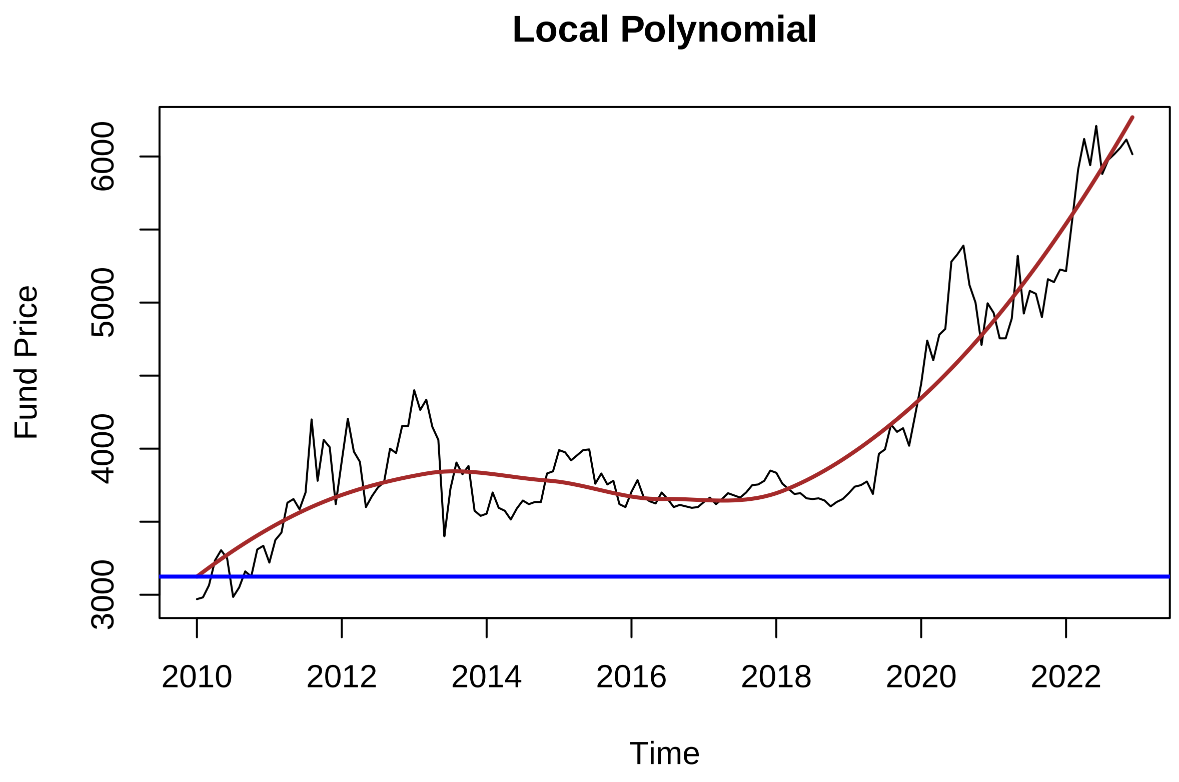
```
##
## Call:
## lm(formula = fund_prices ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -836.35 -234.92  -14.34  286.09  893.73
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3806.3       90.5  42.057  < 2e-16 ***
## x1           -2283.2      420.9  -5.425 2.22e-07 ***
## x2            4259.1      410.0  10.388  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 381.6 on 153 degrees of freedom
## Multiple R-squared:  0.7449, Adjusted R-squared:  0.7416
## F-statistic: 223.4 on 2 and 153 DF,  p-value: < 2.2e-16
```

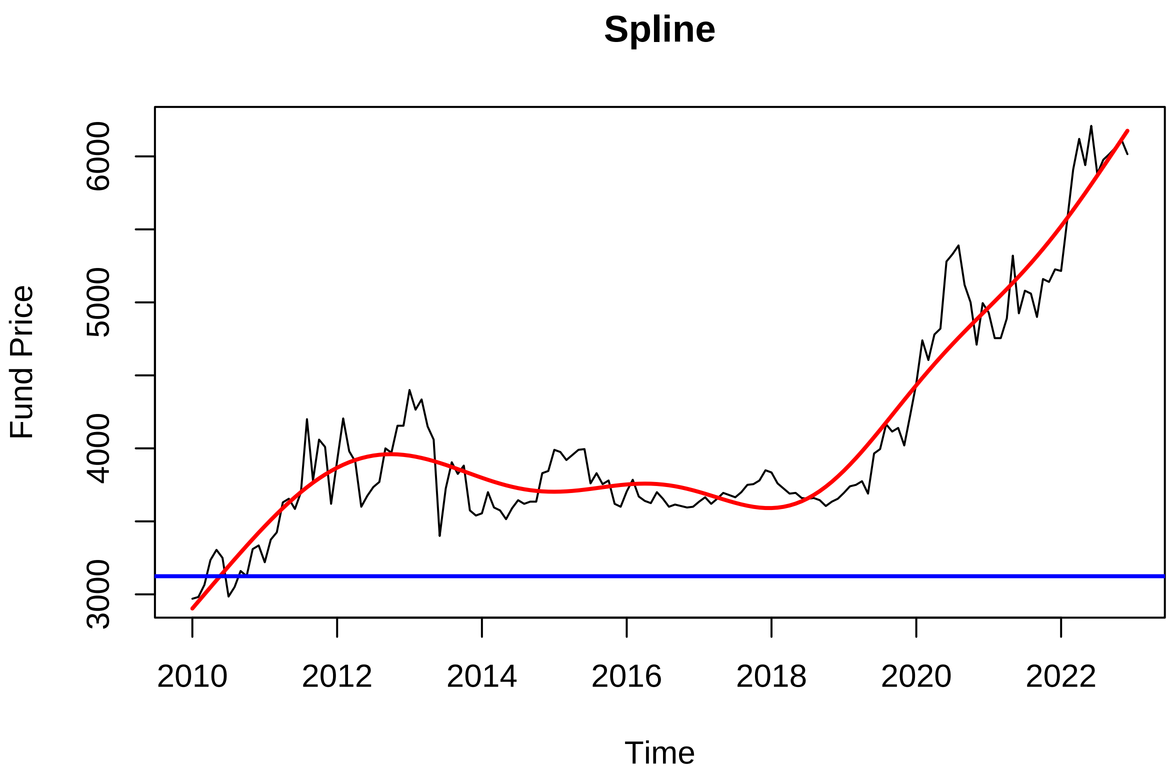
```
fund_prices.fit.lm = ts(fitted(lm.fit),start=c(2010, 1),frequency=12)
ts.plot(fund_prices,ylab="Fund Price", main="Parametric Quadratic Polynomial")
lines(fund_prices.fit.lm,lwd=2,col="green")
abline(fund_prices.fit.mav[1],0,lwd=2,col="blue")
```



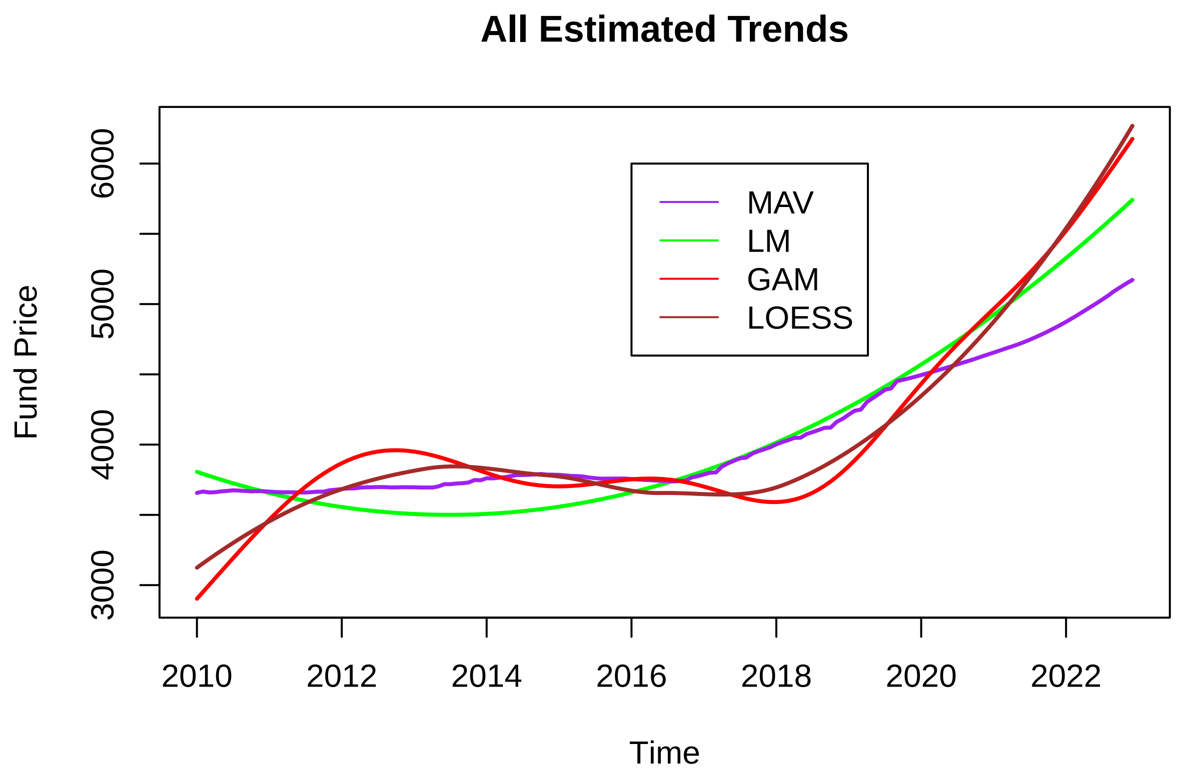
```
## Local Polynomial Trend Estimation
loc.fit = loess(fund_prices~time.pts)
fund_prices.fit.loc = ts(fitted(loc.fit),start=c(2010, 1),frequency=12)
ts.plot(fund_prices,ylab="Fund Price", main = "Local Polynomial")
lines(fund_prices.fit.loc,lwd=2,col="brown")
abline(fund_prices.fit.loc[1],0,lwd=2,col="blue")
```



```
## Splines Trend Estimation
gam.fit = gam(fund_prices~s(time.pts))
fund_prices.fit.gam = ts(fitted(gam.fit),start=c(2010, 1),frequency=12)
ts.plot(fund_prices,ylab="Fund Price", main = "Spline")
lines(fund_prices.fit.gam,lwd=2,col="red")
abline(fund_prices.fit.loc[1],0,lwd=2,col="blue")
```



```
## Compare all estimated trends
all.val = c(fund_prices.fit.mav, fund_prices.fit.lm, fund_prices.fit.gam, fund_prices.fit.loc)
ylim= c(min(all.val),max(all.val))
ts.plot(fund_prices.fit.lm,lwd=2,col="green",ylim=ylim,ylab="Fund Price", main = "All Estimated Trends")
lines(fund_prices.fit.mav,lwd=2,col="purple")
lines(fund_prices.fit.gam,lwd=2,col="red")
lines(fund_prices.fit.loc,lwd=2,col="brown")
legend(x=2016,y=6000,legend=c("MAV","LM","GAM","LOESS"),lty = 1, col=c("purple","green","red","brown"))
```

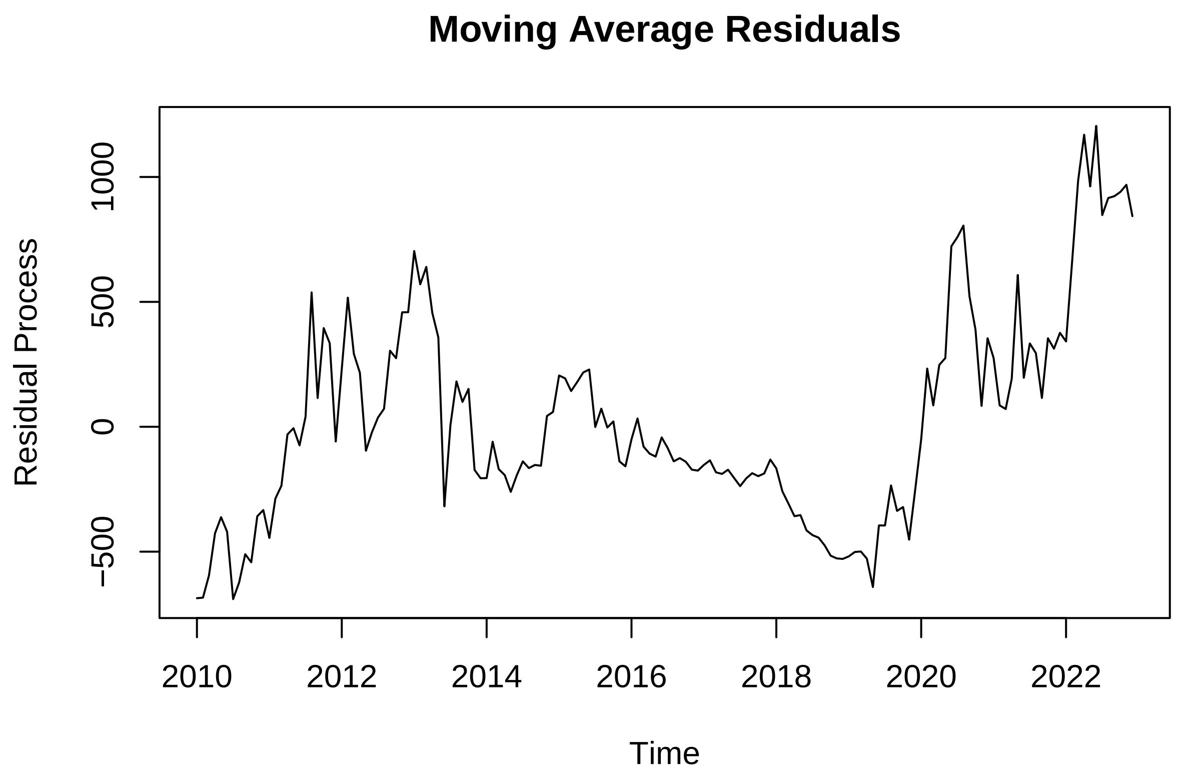


**Response: 1b** All fits show a trend in the time series data. The moving average fit appears to have low variance and high bias, and so it does not have as much sensitivity to changes in the data. The quadratic polynomial also has low variance and high bias, and does relatively well with most of the data from 2012 onwards, but poorly before then. The local polynomial trend match quite well, but it is a little more complex than the quadratic polynomial. Lastly, the spline has a similar fit than the local polynomial trend, but is even more complex than it, with several ups and downs. The local polynomial appears to be the best fit and tradeoff between bias and variance.

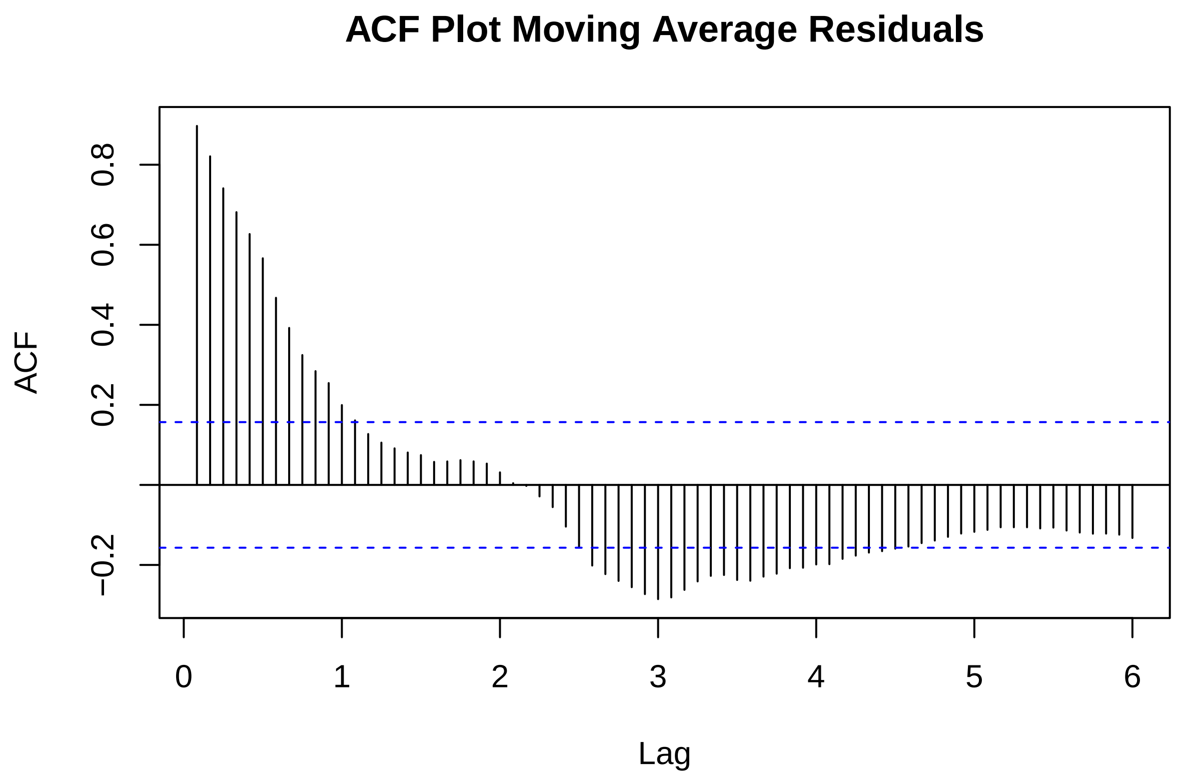
## Question 1c: Residual Analysis

Evaluate the quality of each fit using the residual analysis.

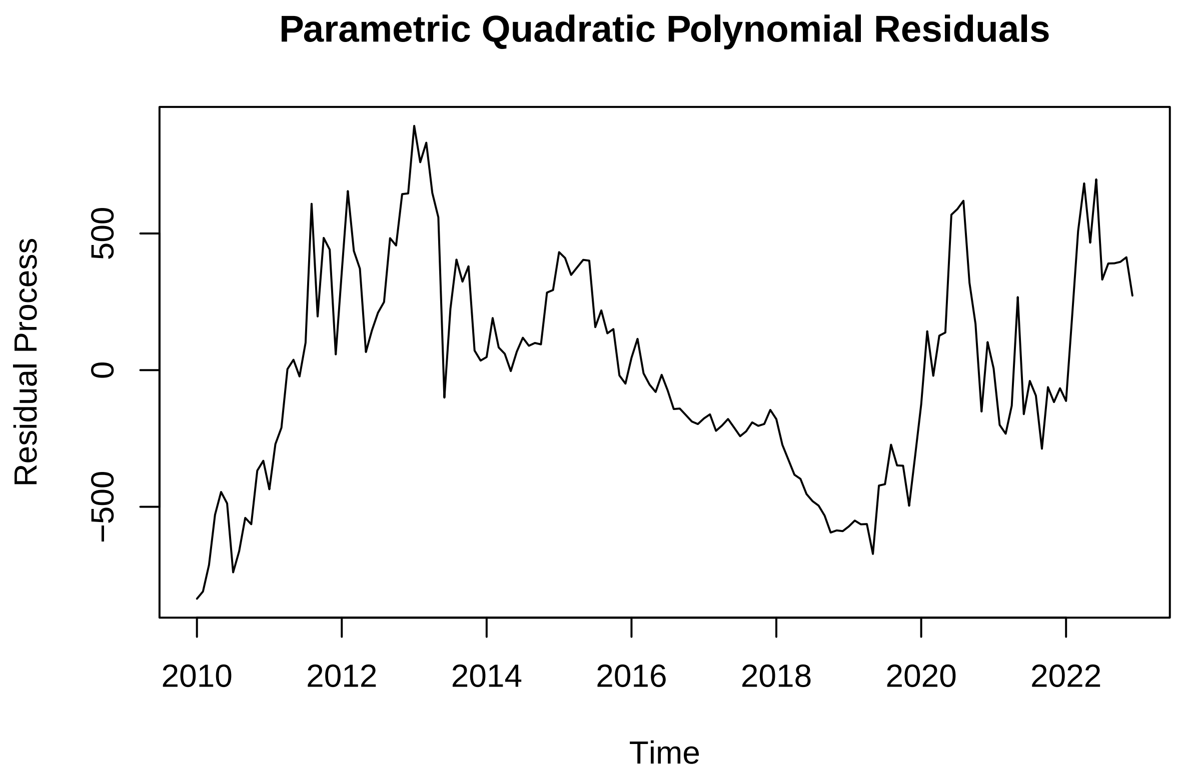
```r
dif.fit.mav = ts((fund_prices-mav.fit$y),start=c(2010, 1),frequency=12)
ts.plot(dif.fit.mav,ylab="Residual Process", main = "Moving Average Residuals")
```



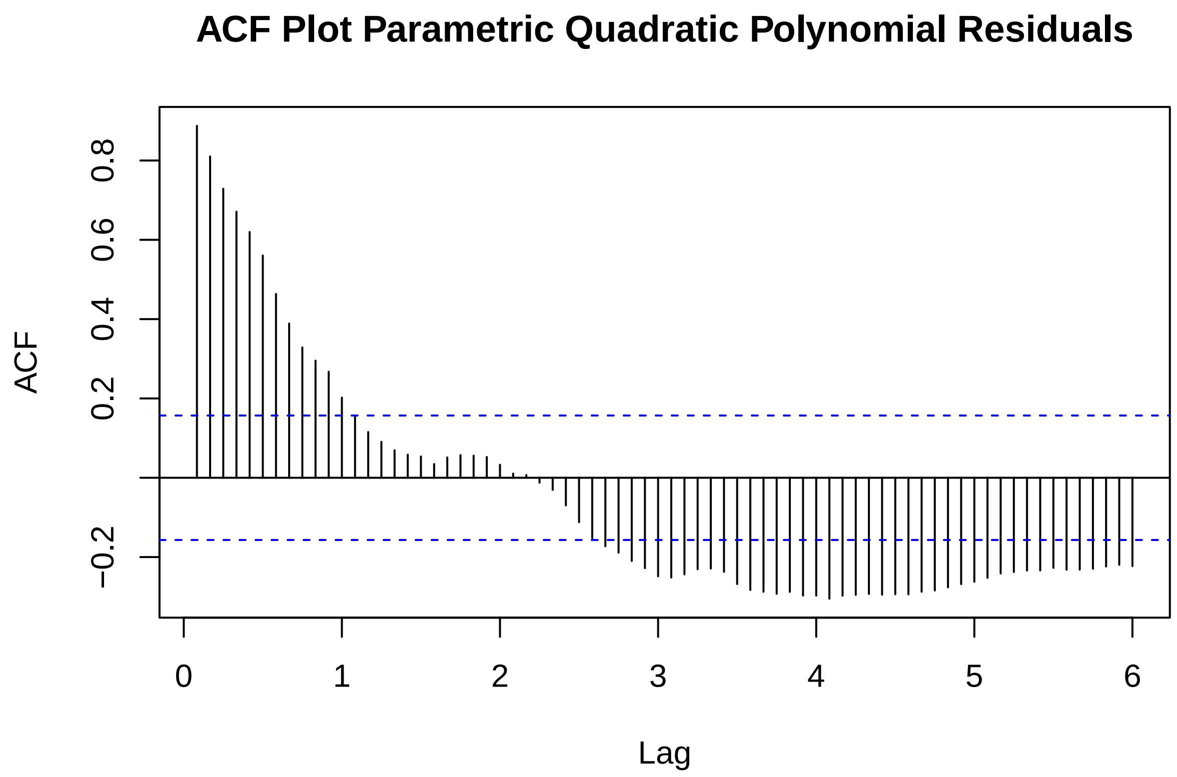
```
acf_plot_mav <- acf(dif.fit.mav, lag.max = 72, main="ACF Plot Moving Average Residuals")
```



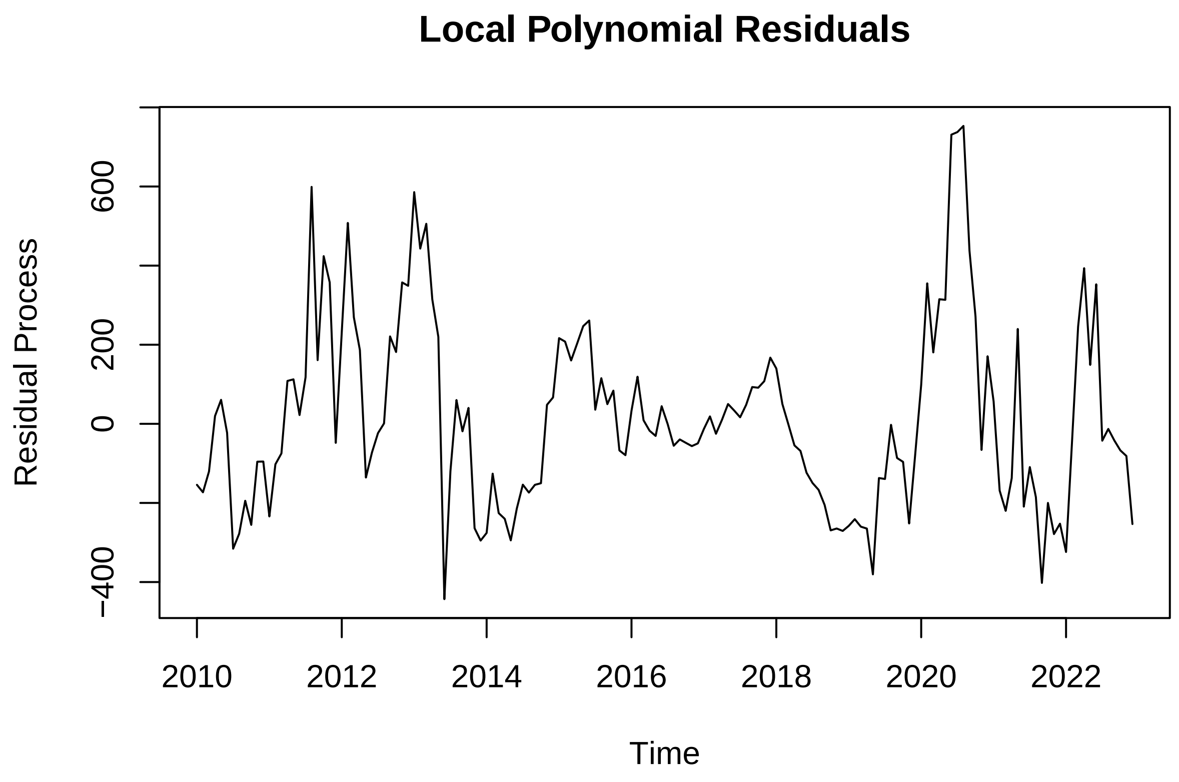
```
dif.fit.lm = ts((lm.fit$residuals),start=c(2010, 1),frequency=12)
ts.plot(dif.fit.lm,ylab="Residual Process", main = "Parametric Quadratic Polynomial Residuals")
```



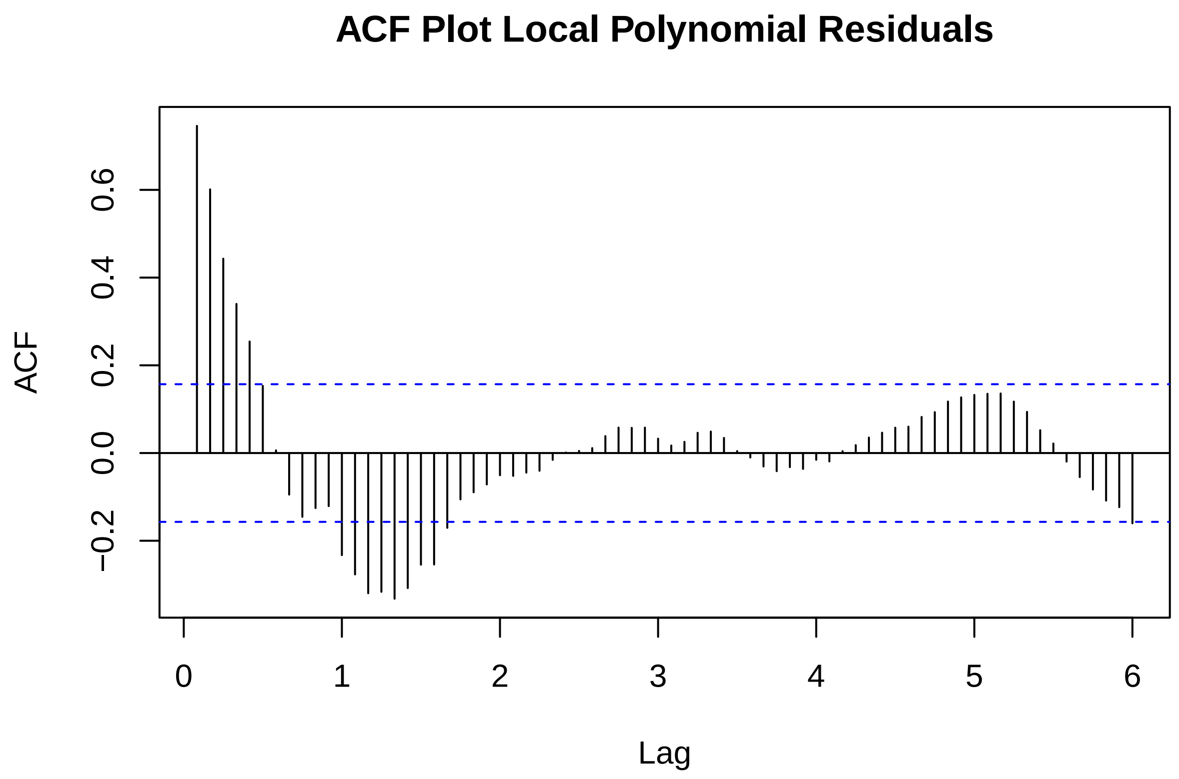
```
acf_plot_llm <- acf(dif.fit.lm, lag.max = 72, main="ACF Plot Parametric Quadratic Polynomial Residuals"
```



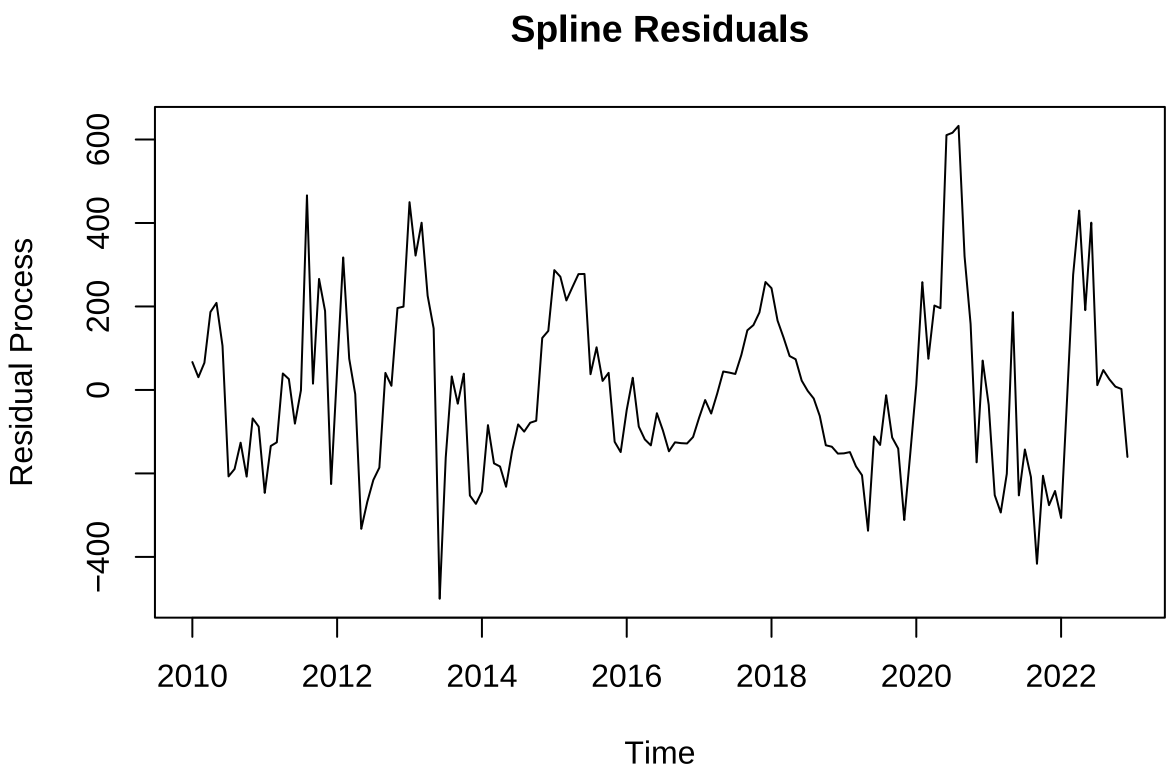
```
dif.fit.loc = ts((fund_prices-fitted(loc.fit)),start=c(2010, 1),frequency=12)
ts.plot(dif.fit.loc,ylab="Residual Process", main = "Local Polynomial Residuals")
```



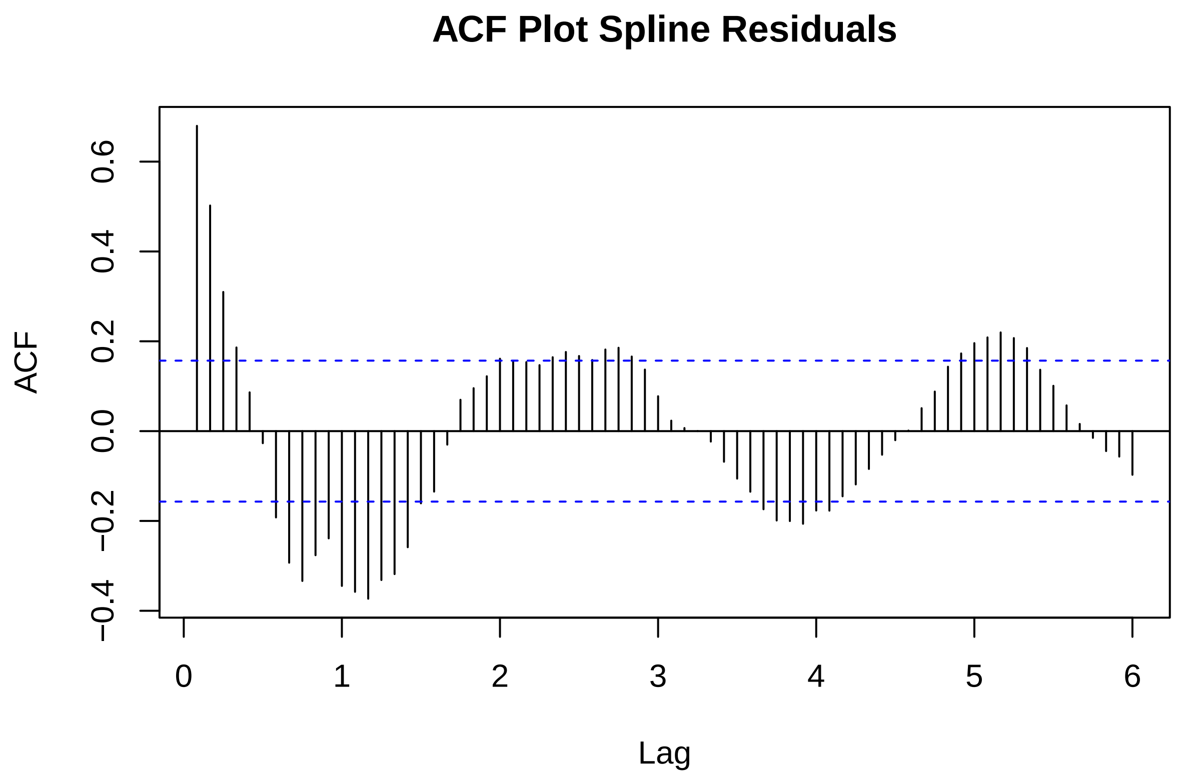
```
acf_plot_loc <- acf(dif.fit.loc, lag.max = 72, main="ACF Plot Local Polynomial Residuals")
```



```
dif.fit.gam = ts((fund_prices-fitted(gam.fit)),start=c(2010, 1),frequency=12)
ts.plot(dif.fit.gam,ylab="Residual Process", main = "Spline Residuals")
```



```
acf_plot_gam <- acf(dif.fit.gam, lag.max = 72, main="ACF Plot Spline Residuals")
```



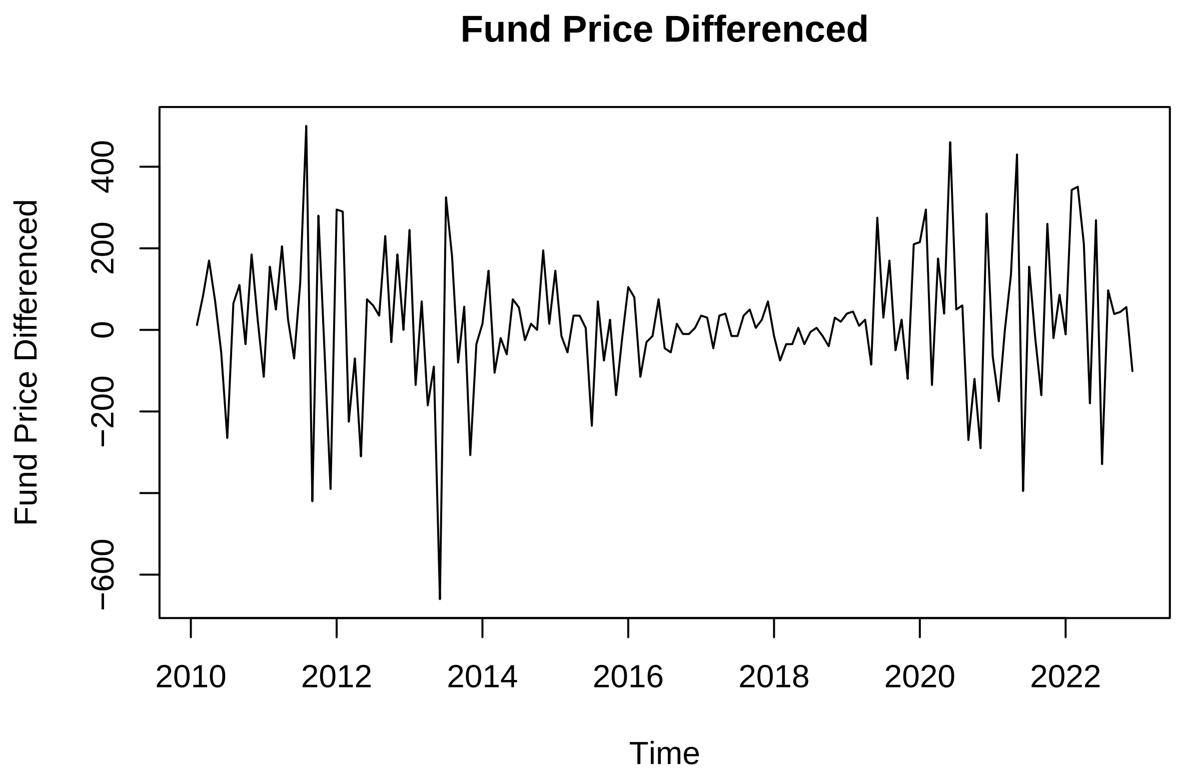
**Response:1c**

Above are the plotted residuals for each fit, as well as the ACF plot. From the residual processes, there appears to be differences between the models used. In the moving average and parametric quadratic acf plots, there appears to be consistent auto correlation exceeding the 95% confidence interval at large lag values. For the local polynomial residuals, there also appears to be some autocorrelation at lag values from 1-2 years. Finally, for the spline residuals, there could be a possible seasonality component, since acf bars exceed the confidence interval in a seasonal manner, but they are very close to it. But there is definitely an autocorrelation around the 1 year mark, similar to the local polynomial residual acf plot. This does mean that stationarity condition 3 is violated. For the first two residual plots (moving average and parametric quadratic), condition 1 also appears to be broken, as there is a very slow reduction in the ACF values, indicating a trend.

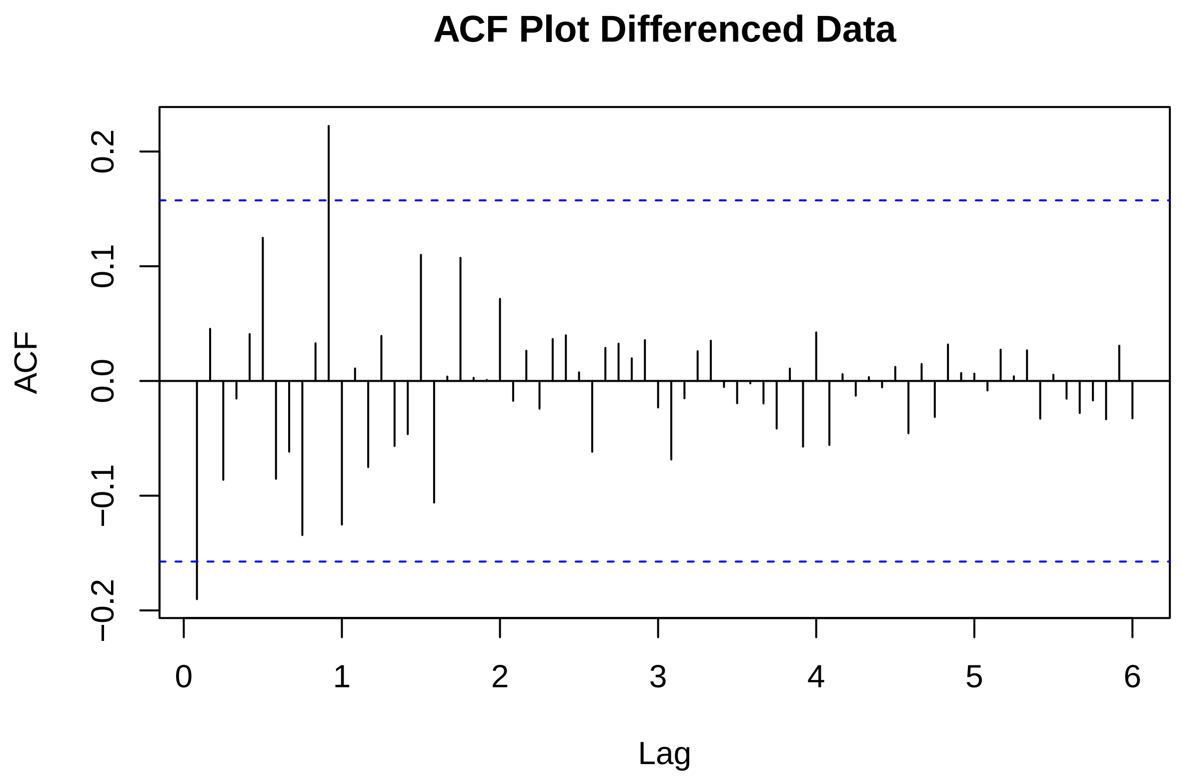
### Question 1d: Differenced Data Modeling

Now plot the difference time series and its ACF plot. Apply the four trend models in Question 1b to the differenced time series. What can you conclude about the difference data in terms of stationarity? Which model would you recommend to apply (trend removal via fitting trend vs differencing) such that to obtain a stationary process?

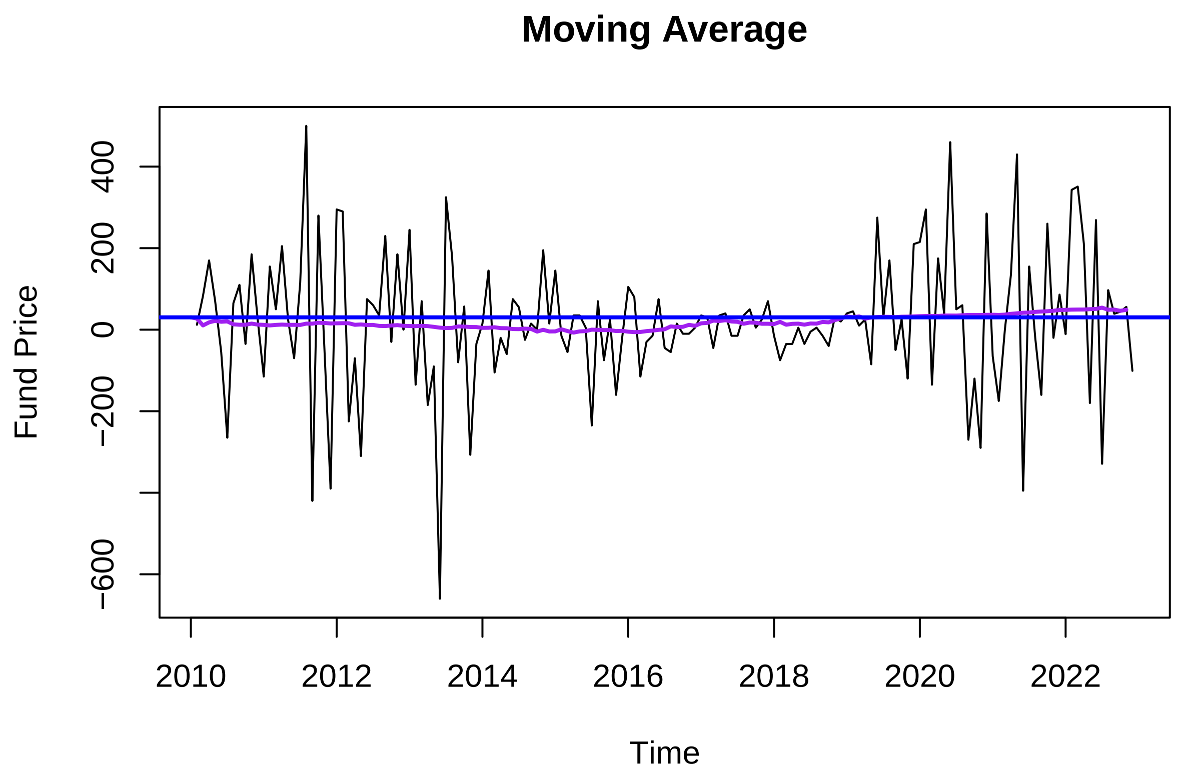
```
fund_prices_diff <- diff(fund_prices)
ts.plot(fund_prices_diff,ylab="Fund Price Differenced", main = "Fund Price Differenced")
```



```r
acf(fund_prices_diff, lag.max = 72, main="ACF Plot Differenced Data")
```



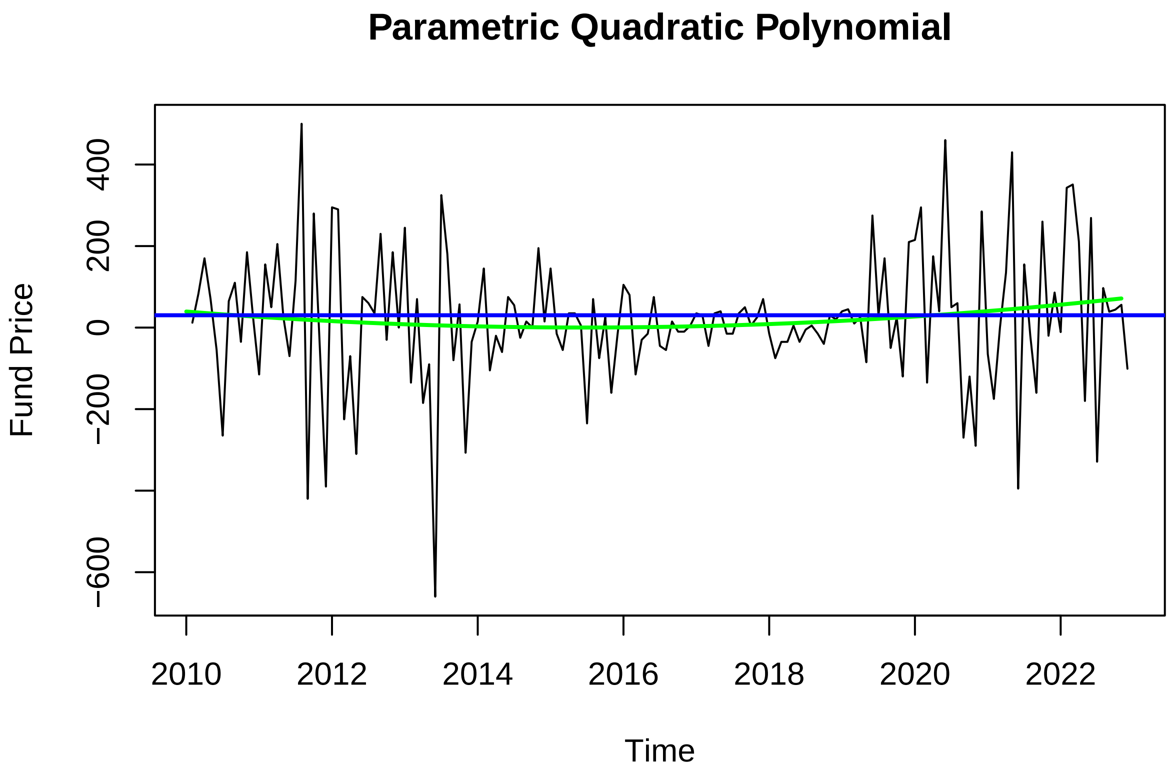
```r
## X-axis points converted to 0-1 scale, common in nonparametric regression
time.pts.diff = c(1:length(fund_prices_diff))
time.pts.diff = c(time.pts.diff - min(time.pts.diff))/max(time.pts.diff)

## Fit a moving average
mav.fit.diff = ksmooth(time.pts.diff, fund_prices_diff, kernel = "box")
fund_prices.fit.mav.diff = ts(mav.fit.diff$y,start=c(2010, 1),frequency=12)
ts.plot(fund_prices_diff,ylab="Fund Price", main="Moving Average")
lines(fund_prices.fit.mav.diff,lwd=2,col="purple")
abline(fund_prices.fit.mav.diff[1],0,lwd=2,col="blue")
```



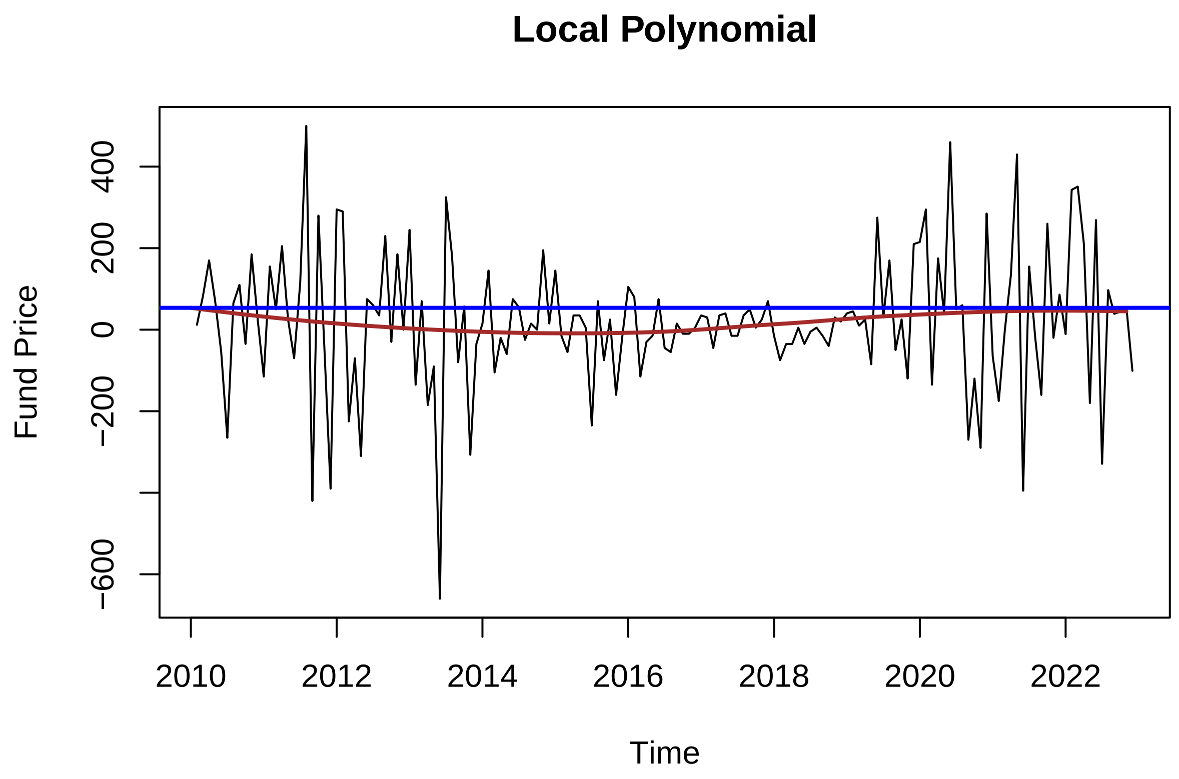
```r
## Fit a parametric quadraric polynomial
x1.diff = time.pts.diff
x2.diff = time.pts.diff^2
lm.fit.diff = lm(fund_prices_diff~x1.diff+x2.diff)
summary(lm.fit.diff)
```

```
##
## Call:
## lm(formula = fund_prices_diff ~ x1.diff + x2.diff)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -666.17  -65.62    1.61   66.10  479.17
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    39.38      39.49   0.997    0.320
## x1.diff      -185.14     183.68  -1.008    0.315
## x2.diff       218.84     178.93   1.223    0.223
##
## Residual standard error: 166 on 152 degrees of freedom
## Multiple R-squared:  0.01289,    Adjusted R-squared:  -0.0001012
## F-statistic: 0.9922 on 2 and 152 DF,  p-value: 0.3731
```

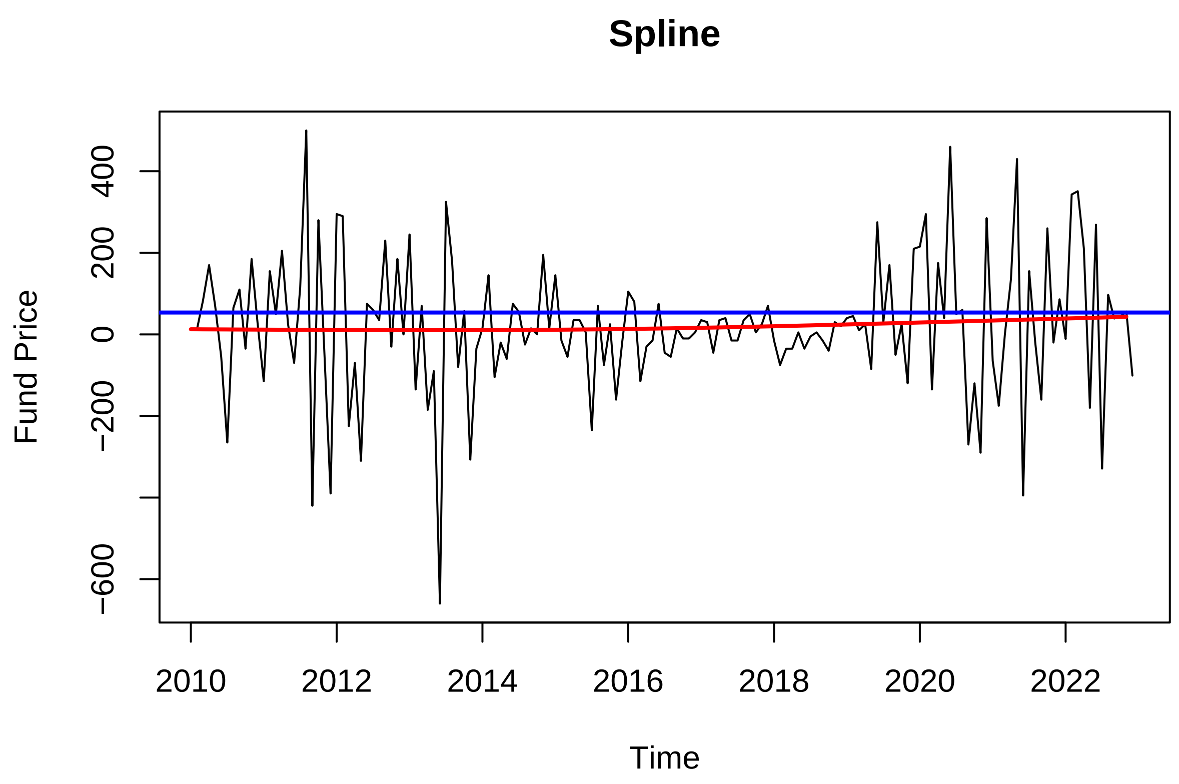
```
fund_prices.fit.lm.diff = ts(fitted(lm.fit.diff),start=c(2010, 1),frequency=12)
ts.plot(fund_prices_diff,ylab="Fund Price", main="Parametric Quadratic Polynomial")
lines(fund_prices.fit.lm.diff,lwd=2,col="green")
abline(fund_prices.fit.mav.diff[1],0,lwd=2,col="blue")
```



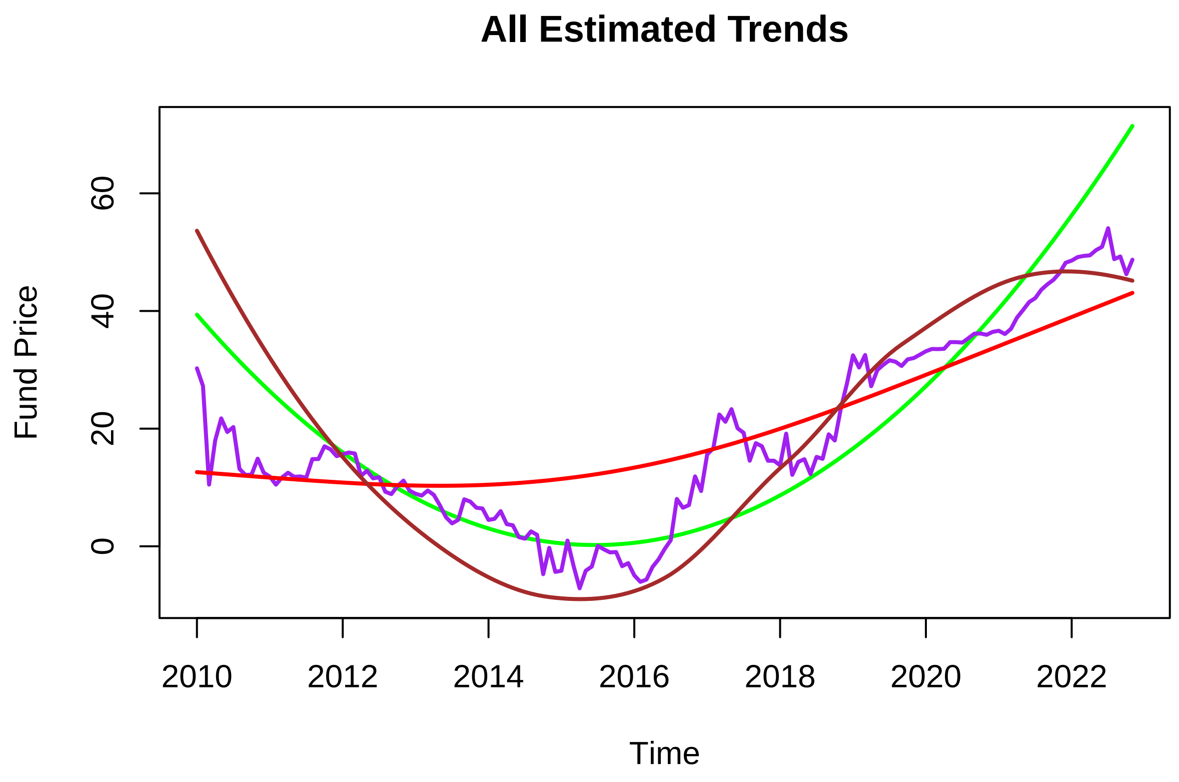
```
## Local Polynomial Trend Estimation
loc.fit.diff = loess(fund_prices_diff~time.pts.diff)
fund_prices.fit.loc.diff = ts(fitted(loc.fit.diff),start=c(2010, 1),frequency=12)
ts.plot(fund_prices_diff,ylab="Fund Price", main = "Local Polynomial")
lines(fund_prices.fit.loc.diff,lwd=2,col="brown")
abline(fund_prices.fit.loc.diff[1],0,lwd=2,col="blue")
```



```
## Splines Trend Estimation
gam.fit.diff = gam(fund_prices_diff~s(time.pts.diff))
fund_prices.fit.gam.diff = ts(fitted(gam.fit.diff),start=c(2010, 1),frequency=12)
ts.plot(fund_prices_diff,ylab="Fund Price", main = "Spline")
lines(fund_prices.fit.gam.diff,lwd=2,col="red")
abline(fund_prices.fit.loc.diff[1],0,lwd=2,col="blue")
```



```
## Compare all estimated trends
all.val.diff = c(fund_prices.fit.mav.diff, fund_prices.fit.lm.diff, fund_prices.fit.gam.diff, fund_pric
ylim.diff= c(min(all.val.diff),max(all.val.diff))
ts.plot(fund_prices.fit.lm.diff,lwd=2,col="green",ylim=ylim.diff,ylab="Fund Price", main = "All Estimat
lines(fund_prices.fit.mav.diff,lwd=2,col="purple")
lines(fund_prices.fit.gam.diff,lwd=2,col="red")
lines(fund_prices.fit.loc.diff,lwd=2,col="brown")
legend(x=2016,y=8000,legend=c("MAV","LM","GAM","LOESS"),lty = 1, col=c("purple","green","red","brown"))
```



**Response 1d**

In this case, looking at the ACF plot, it appears that there is only one bar after lag = 2, where the ACF value exceeds the 95% confidence interval. Based on this, we can say that the 3rd stationarity condition is met. Looking at the four trend fits, it appears that the first stationarity condition is also met, since there is very little variation for the mean, and none of them suggest a trend. Looking at the differenced data, there doesn't seem to be any sudden changes in the variability, so condition 2 is met.

As for recommending an approach, the differencing would be the best choice, as it effectively removed the trend, or any seasonality in the data, and achieved conditions 1 & 3 as shown by the ACF plots.

## Part 2: Temperature Analysis

## Background

In this problem, we will analyze quarterly average temperature data. The data file Temperature HW 2.csv contains average monthly temperature from a southern region from January 1980 through Dec 2016. We will aggregate the data on a quarterly basis, by taking the average rate within each quarter. We will fit the models on the data until Quarter 4 of 2015 and evaluate the predictions for Q1 to Q4 2016.

### Instructions on reading the data

To read the data in R, save the file in your working directory (make sure you have changed the directory if different from the R working directory) and read the data using the R function read.csv()

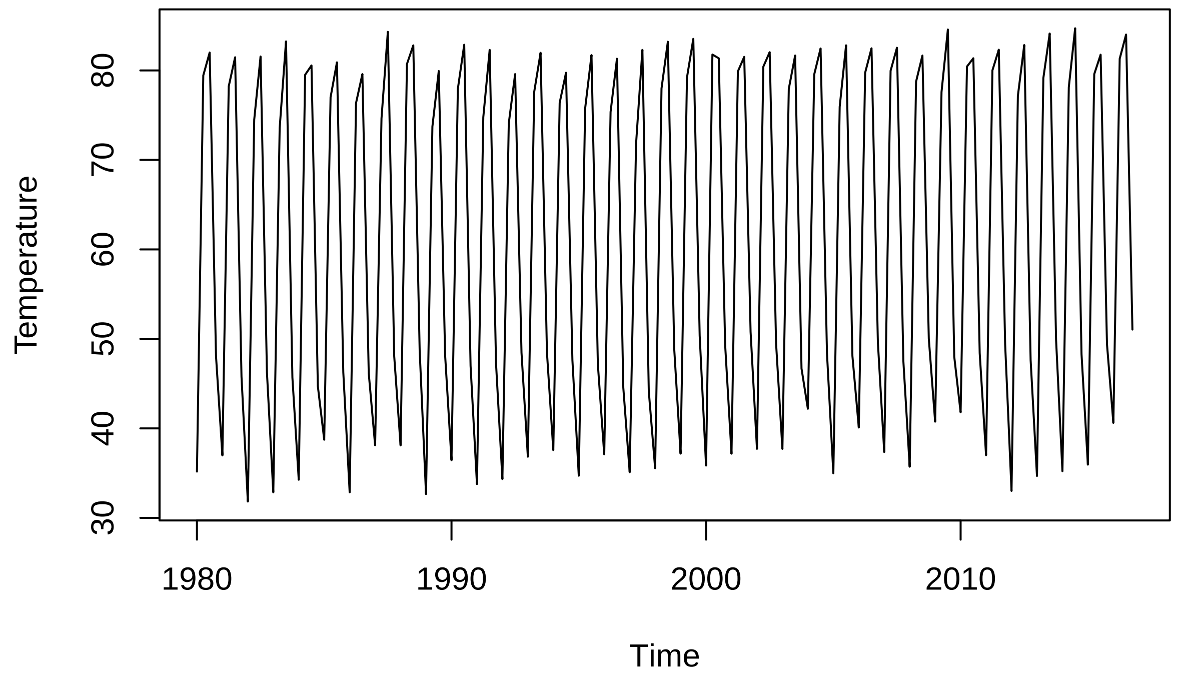
You will perform the analysis and modelling on the Temperature data column.

```r
#Run the following code to prepare the data for analysis:

df2 <- read.csv("Temperature HW 2.csv", head = TRUE)
temp <- ts(df2$Temperature, freq = 12, start = c(1980,1))
temp <- aggregate.ts(temp, nfrequency = 4)
```

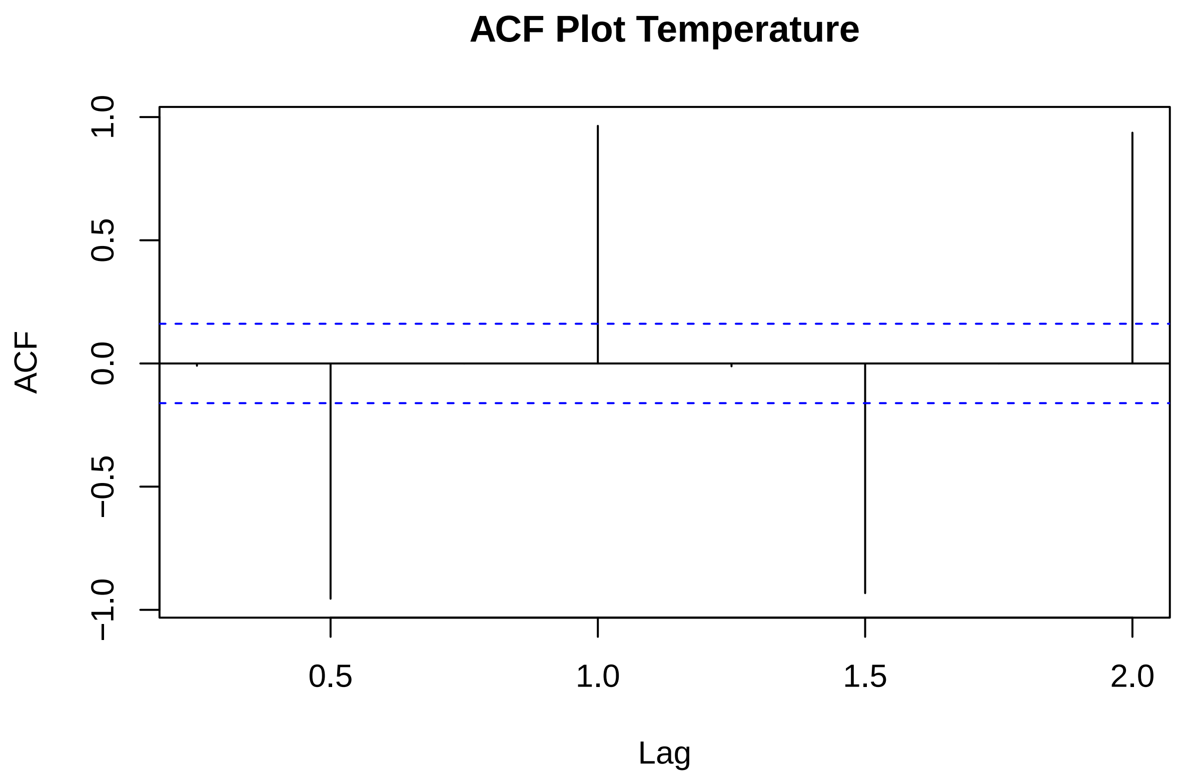
## Question 2a: Exploratory Data Analysis

Plot both the Time Series and ACF plots. Comment on the main features, and identify what (if any) assumptions of stationarity are violated. Additionally, consider the differenced data and comment on its features. Support your response with a graphical analysis.

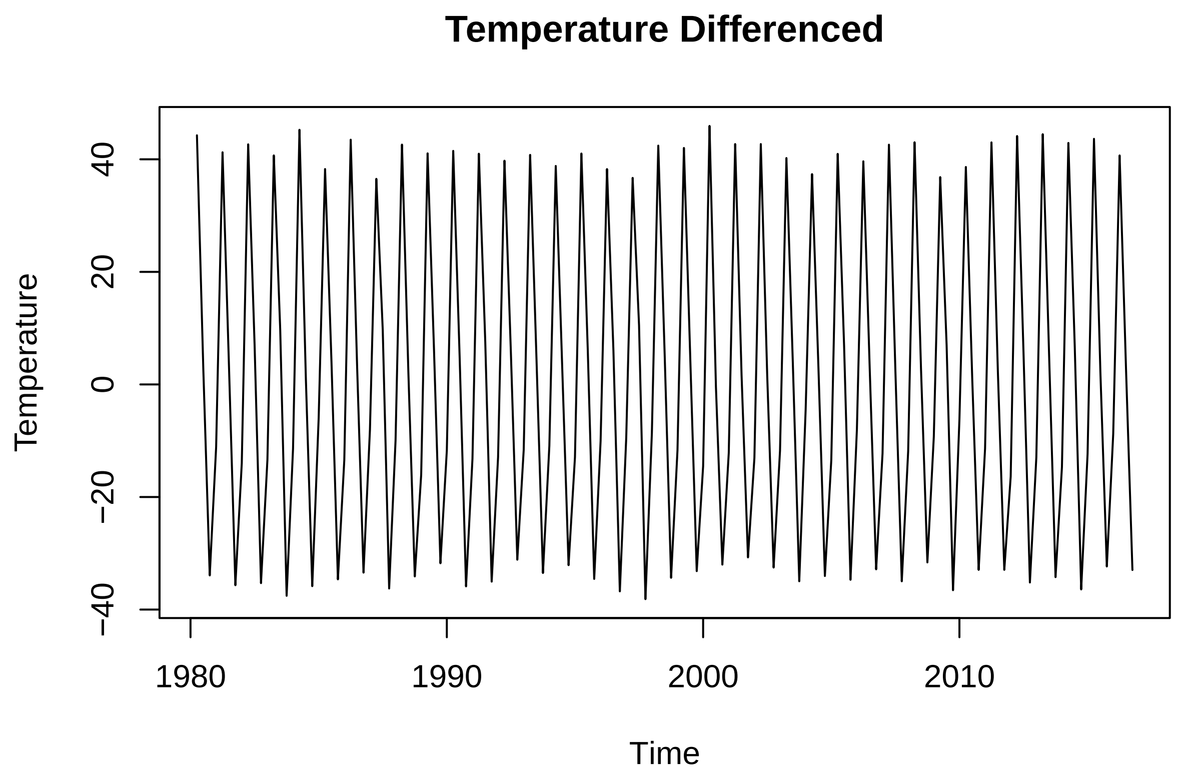
```r
## X-axis points converted to 0-1 scale, common in nonparametric regression
time.pts = c(1:length(temp))
time.pts = c(time.pts - min(time.pts))/max(time.pts)

ts.plot(temp,ylab="Temperature")
```



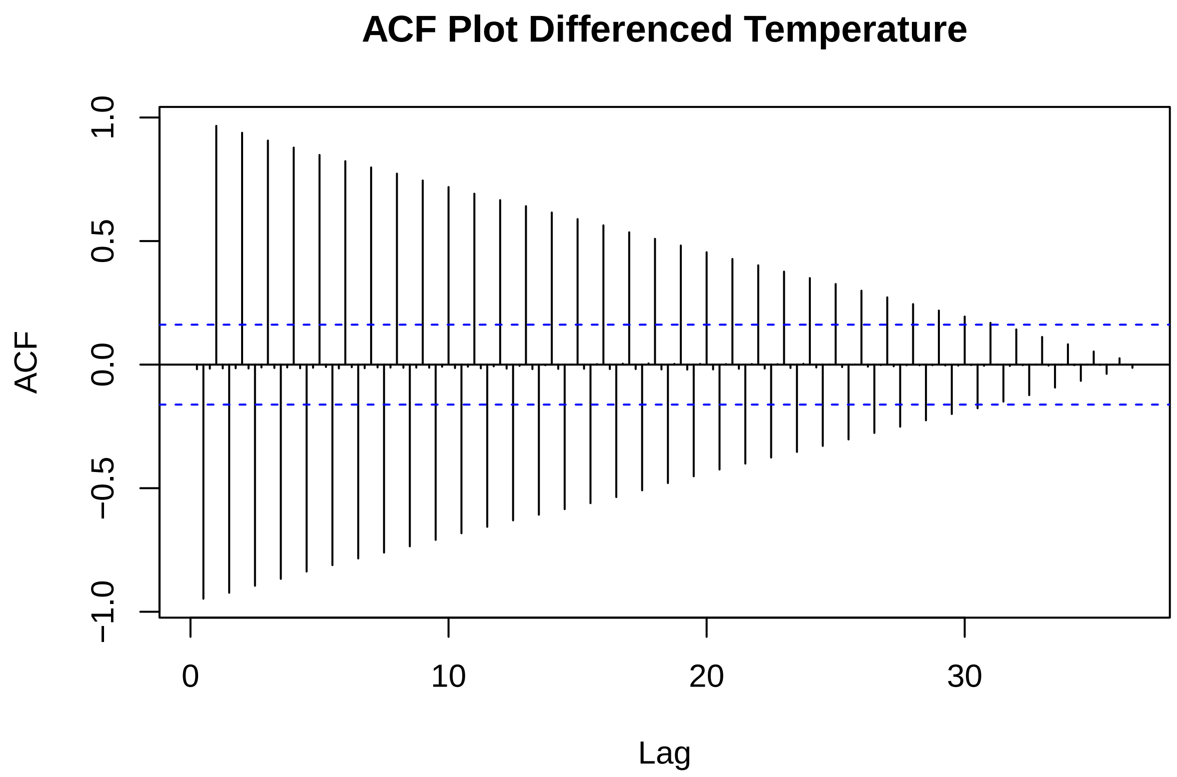
```r
acf(temp, lag.max = 8, main="ACF Plot Temperature")
```



```
temp_diff <- diff(temp)
ts.plot(temp_diff,ylab="Temperature", main = "Temperature Differenced")
```



```
acf(temp_diff, lag.max = 400, main="ACF Plot Differenced Temperature")
```



**Response: 2a**

Based on the plots above, there is clearly seasonality present in the data. For both plots, even after differencing, there is an alternating pattern in the ACF plots that persists for a long time. This is also evident in the time series plots for both the undifferenced and differenced plots, where there is a clear seasonal pattern. This is to be expected, as the average temperature of 4 months centered on January 1st of this year would be highly correlated to the same period of the previous year. It is clear that the 3rd stationarity assumption is not met as there is significant autocorrelation from the seasonality. The variance also appears to remain finite and constant with time. Lastly, at first glance, it appears that the mean does not significantly change with time.

## Question 2b: Seasonality Estimation

Separately fit a seasonality harmonic model and the ANOVA seasonality model to the temperature data. Evaluate the quality of each fit using the residual analysis. Does one model perform better than the other? Which model would you select to fit the seasonality in the data?

```
## Estimate seasonality using ANOVA approach

model1 = dynlm(temp~season(temp))
summary(model1)
```

```
##
## Time series regression with "ts" data:
## Start = 1980(1), End = 2016(4)
```

```
##
## Call:
## dynlm(formula = temp ~ season(temp))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9308 -1.2792  0.1586  1.4035  5.7621
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     36.4322     0.3410  106.85   <2e-16 ***
## season(temp)Q2  41.2919     0.4822   85.63   <2e-16 ***
## season(temp)Q3  45.7050     0.4822   94.78   <2e-16 ***
## season(temp)Q4  11.5116     0.4822   23.87   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.074 on 144 degrees of freedom
## Multiple R-squared:  0.989,  Adjusted R-squared:  0.9887
## F-statistic:  4302 on 3 and 144 DF,  p-value: < 2.2e-16
```

```
## All seasonal mean effects (model without intercept)
model2 = dynlm(temp~season(temp)-1)
summary(model2)
```

```
##
## Time series regression with "ts" data:
## Start = 1980(1), End = 2016(4)
##
## Call:
## dynlm(formula = temp ~ season(temp) - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9308 -1.2792  0.1586  1.4035  5.7621
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## season(temp)Q1   36.432      0.341   106.8   <2e-16 ***
## season(temp)Q2   77.724      0.341   227.9   <2e-16 ***
## season(temp)Q3   82.137      0.341   240.9   <2e-16 ***
## season(temp)Q4   47.944      0.341   140.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.074 on 144 degrees of freedom
## Multiple R-squared:  0.999,  Adjusted R-squared:  0.999
## F-statistic: 3.529e+04 on 4 and 144 DF,  p-value: < 2.2e-16
```

```
model3 = dynlm(temp~harmon(temp))
summary(model3)
```

```
##
```

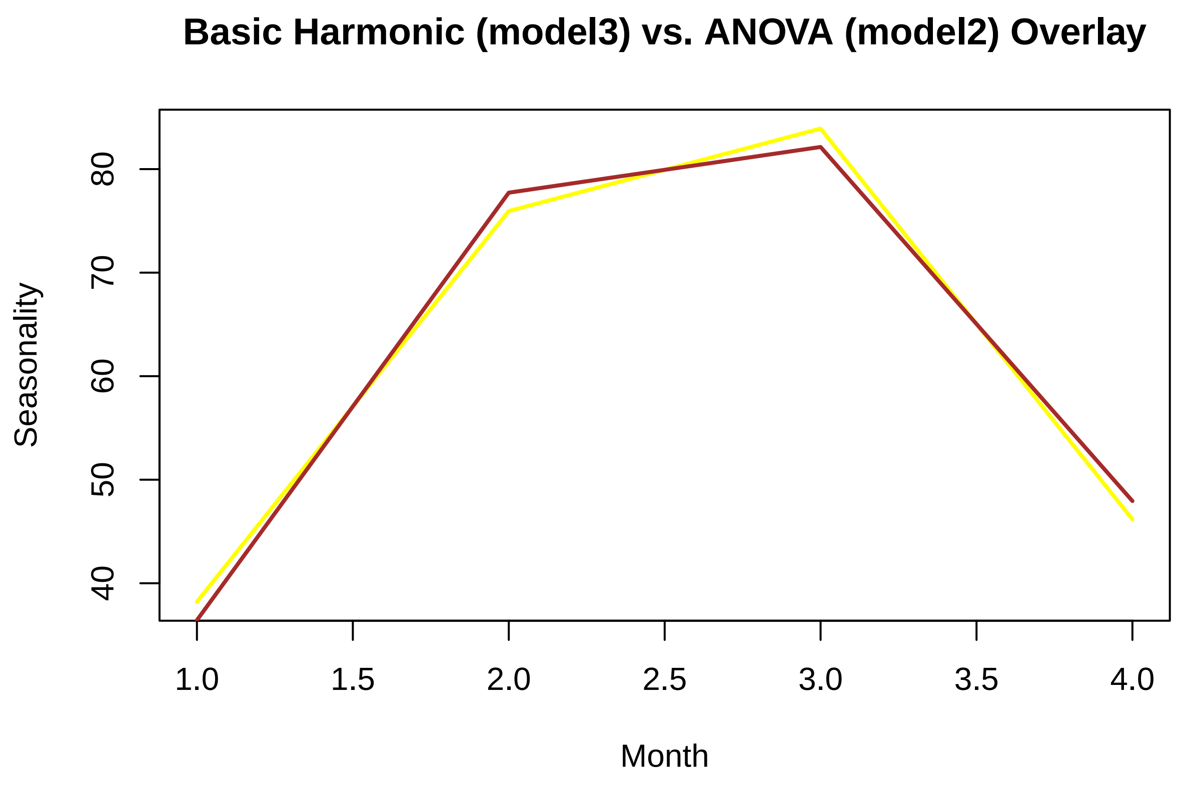
```
## Time series regression with "ts" data:
## Start = 1980(1), End = 2016(4)
##
## Call:
## dynlm(formula = temp ~ harmon(temp))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3802 -2.0522 -0.2875  2.1904  5.8253
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      61.0593     0.2249  271.48   <2e-16 ***
## harmon(temp)cos -22.8525     0.3181  -71.85   <2e-16 ***
## harmon(temp)sin  14.8901     0.3181   46.81   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.736 on 145 degrees of freedom
## Multiple R-squared:  0.9807, Adjusted R-squared:  0.9804
## F-statistic:  3677 on 2 and 145 DF,  p-value: < 2.2e-16
```

```r
## Estimate seasonality using cos-sin model
model4= dynlm(temp~harmon(temp,2))
summary(model4)
```

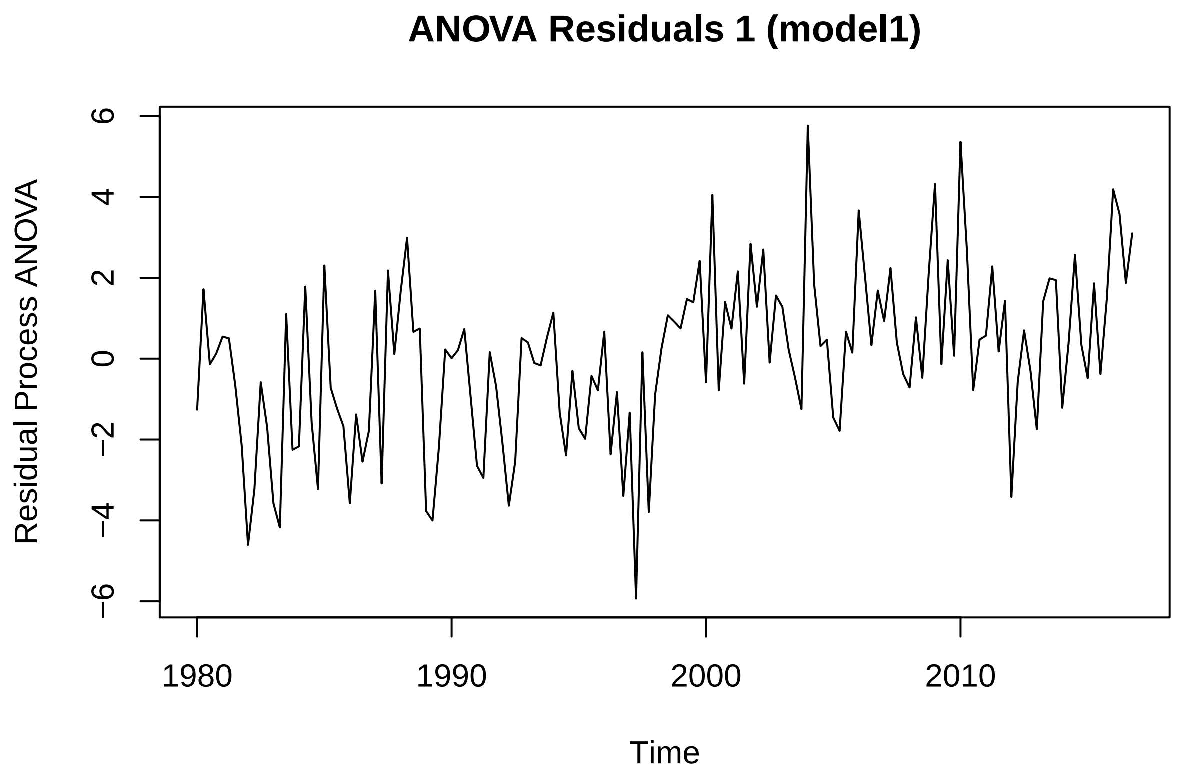
```
##
## Time series regression with "ts" data:
## Start = 1980(1), End = 2016(4)
##
## Call:
## dynlm(formula = temp ~ harmon(temp, 2))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.9308 -1.2792  0.1586  1.4035  5.7621
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)          61.0593     0.1705  358.14   <2e-16 ***
## harmon(temp, 2)cos1 -22.8525     0.2411  -94.78   <2e-16 ***
## harmon(temp, 2)cos2  -1.7746     0.1705  -10.41   <2e-16 ***
## harmon(temp, 2)sin1  14.8901     0.2411   61.76   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.074 on 144 degrees of freedom
## Multiple R-squared:  0.989,  Adjusted R-squared:  0.9887
## F-statistic:  4302 on 3 and 144 DF,  p-value: < 2.2e-16
```

```r
## Compare Seasonality Estimates
## Seasonal Means Model
st1 = coef(model2)
```

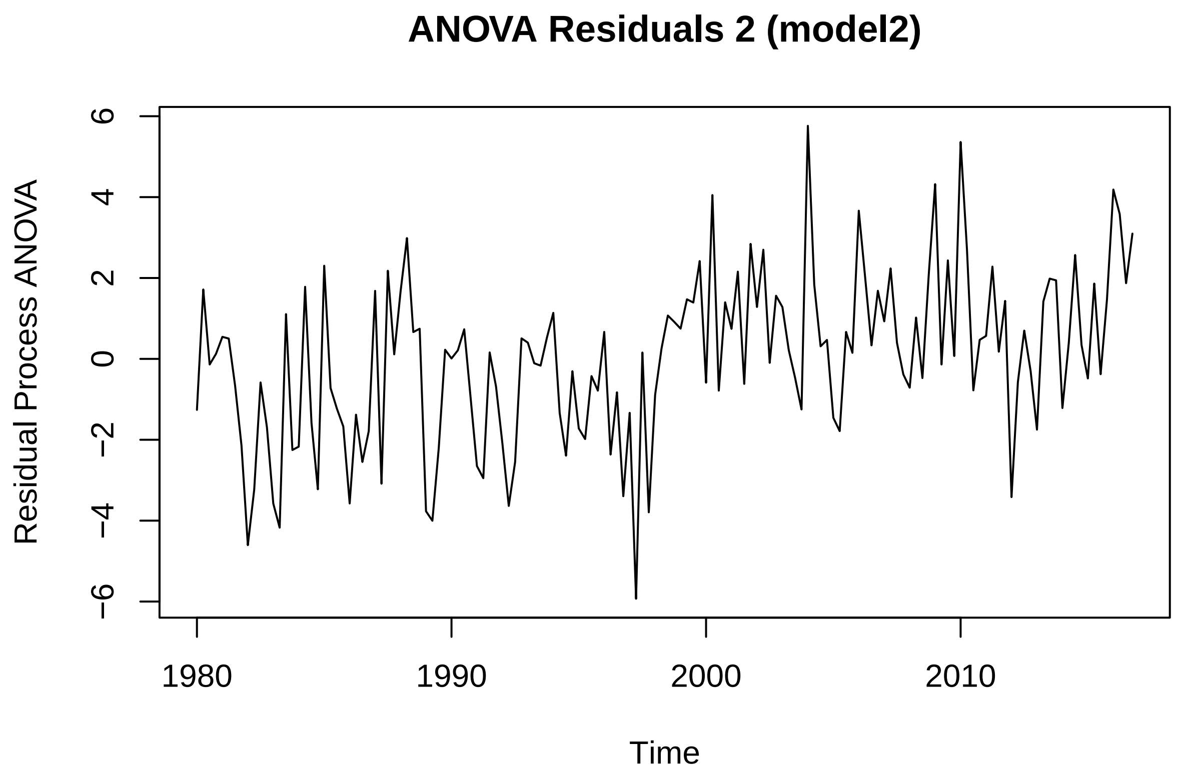
```r
## Cos-Sin Model
st2 = fitted(model3)[1:4]
plot(1:4,st2,lwd=2,type="l",xlab="Month",ylab="Seasonality", col="yellow", main = "Basic Harmonic (model
lines(1:4,st1,lwd=2, col="brown")
```



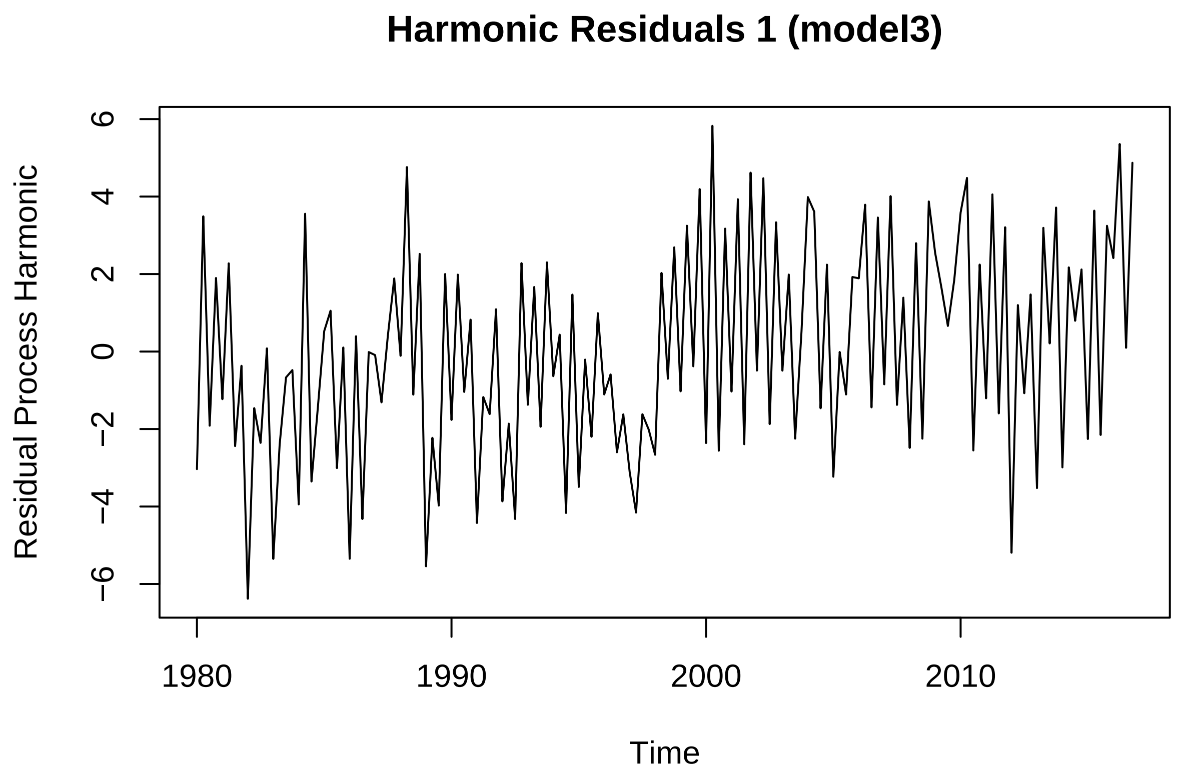
```r
dif.model1 = ts((model1$residuals),start=c(1980, 1),frequency=4)
ts.plot(dif.model1, ylab="Residual Process ANOVA", main="ANOVA Residuals 1 (model1)")
```



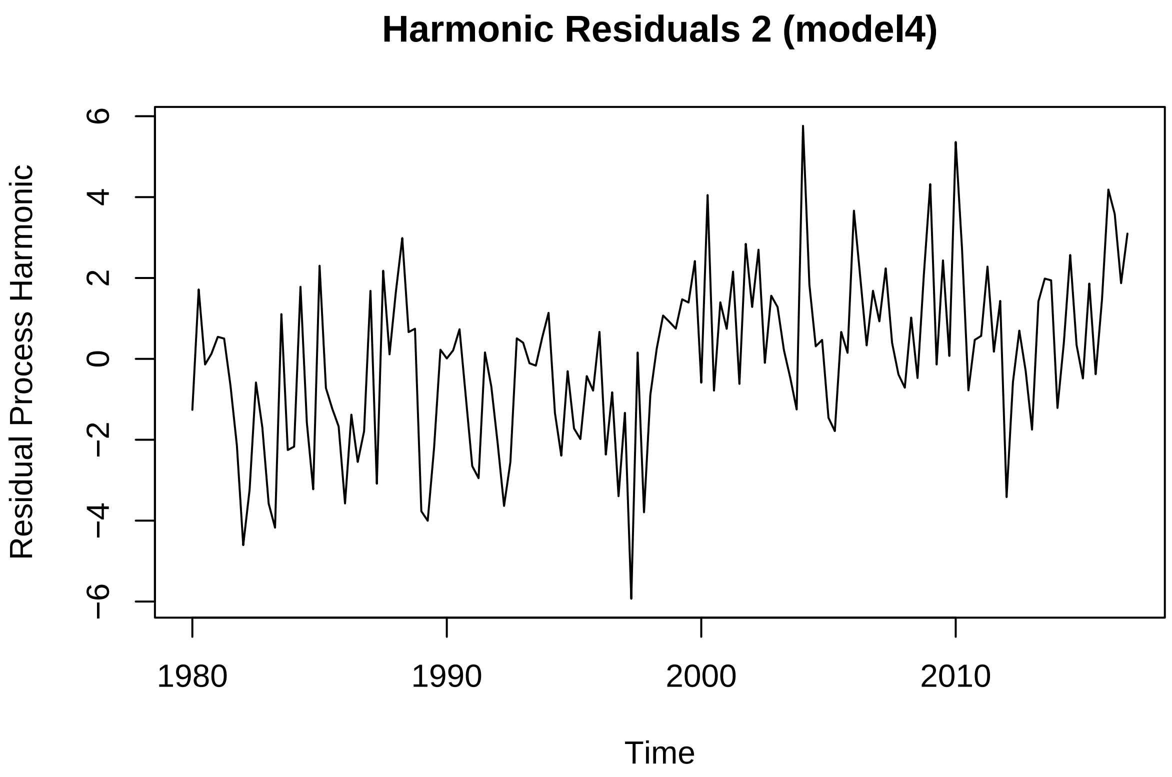
```
dif.model2 = ts((model2$residuals),start=c(1980, 1),frequency=4)
ts.plot(dif.model2, ylab="Residual Process ANOVA", main="ANOVA Residuals 2 (model2)")
```



```
dif.model3 = ts((model3$residuals),start=c(1980, 1),frequency=4)
ts.plot(dif.model3, ylab="Residual Process Harmonic", main = "Harmonic Residuals 1 (model3)")
```



```r
dif.model4 = ts((model4$residuals),start=c(1980, 1),frequency=4)
ts.plot(dif.model4, ylab="Residual Process Harmonic", main = "Harmonic Residuals 2 (model4)")
```



**Response: 2b**

It appears that both models are extremely similar when observing the residual process plots. We may need to check for a trend, since it does appear graphically that there is a small trend over the years. When looking at the summaries of each model, it appears that the ANOVA approach has a slight performance increase with a R-squared value of .999, and so it captures a slightly greater percentage of the variability in the data. The basic harmonic model (model3) has a slight difference when overlayed on the ANOVA model, and so it may not capture the seasonality as well. There is a trade-off between bias and variance. In this case, I will select the ANOVA model as the better one, even though it uses an additional regression coefficient compared to the basic harmonic model, since the increase in seasonality capture is significant.

## Question 2c: Trend-Seasonality Estimation

Using the time-series data, fit the following models to estimate the trend with seasonality fitted using ANOVA:

- Parametric quadratic polynomial
- Non-parametric model

Overlay the fitted values of the two models on the original time series. What do you conclude in terms of trend over time?

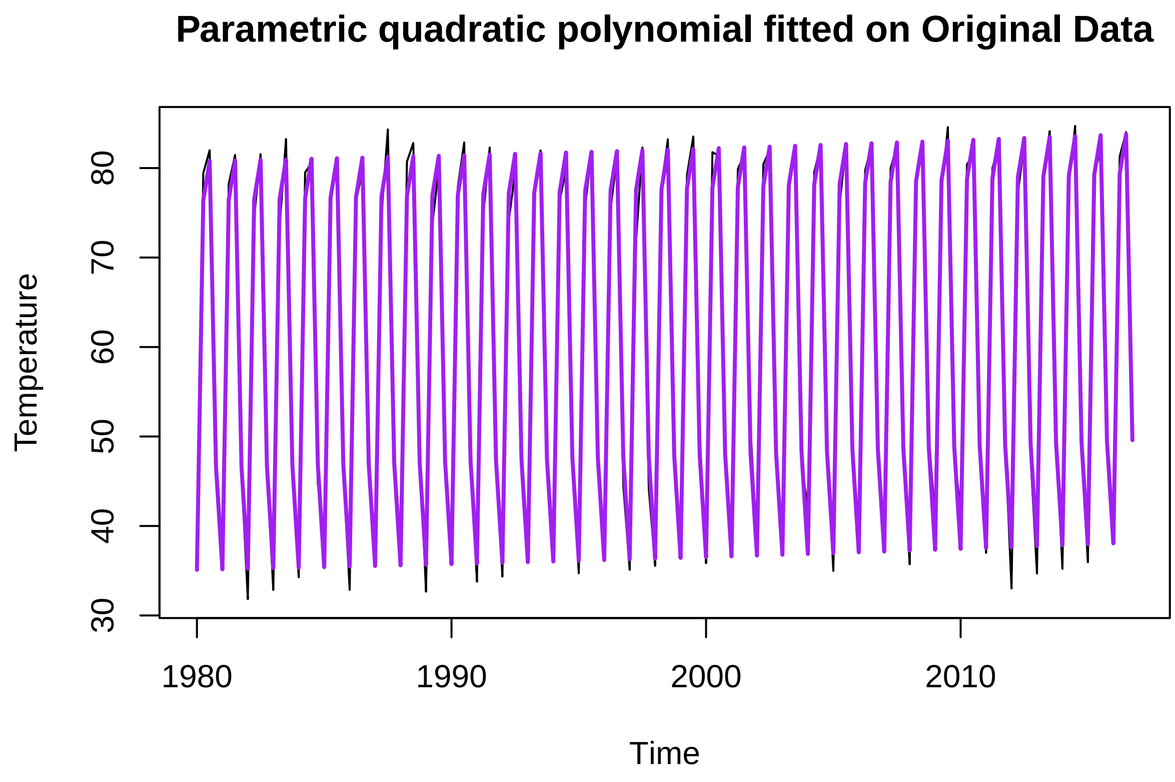
Plot the residuals with respect to time. Plot the ACF of the residuals. Comment on how the two models fit and on the appropriateness of the stationarity assumption of the residuals.

What form of modeling seems most appropriate and what implications might this have for how one might expect long term temperature data to behave? Provide explicit conclusions based on the data analysis.

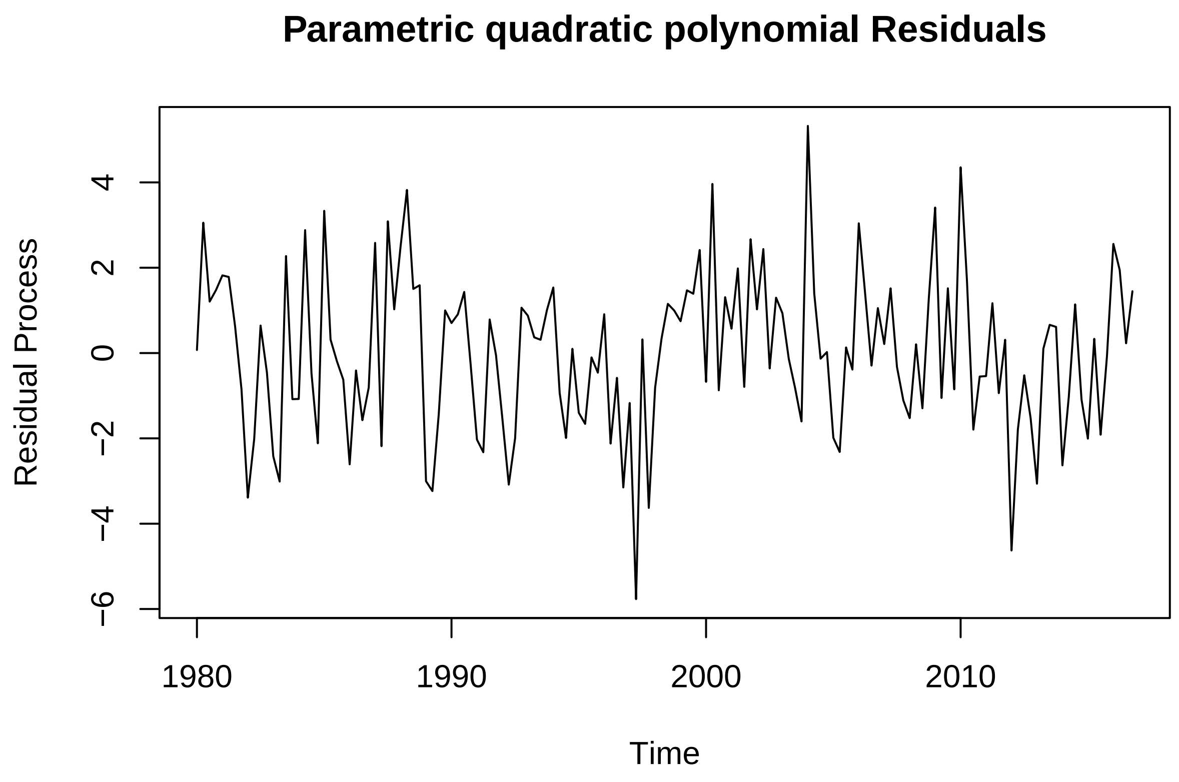
```r
x1 = time.pts
x2 = time.pts^2
lm.fit = dynlm(temp~x1+x2+season(temp)-1)
summary(lm.fit)
```

```
##
## Time series regression with "ts" data:
## Start = 1980(1), End = 2016(4)
##
## Call:
## dynlm(formula = temp ~ x1 + x2 + season(temp) - 1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.7681 -1.1263  0.0474  1.2239  5.3235
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## x1                2.1017     2.1303   0.987    0.326
## x2                0.9694     2.0757   0.467    0.641
## season(temp)Q1  35.0996     0.5278  66.501   <2e-16 ***
## season(temp)Q2  76.3709     0.5297 144.169   <2e-16 ***
## season(temp)Q3  80.7633     0.5315 151.940   <2e-16 ***
## season(temp)Q4  46.5491     0.5333  87.291   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.882 on 142 degrees of freedom
## Multiple R-squared:  0.9992, Adjusted R-squared:  0.9991
## F-statistic: 2.858e+04 on 6 and 142 DF,  p-value: < 2.2e-16
```

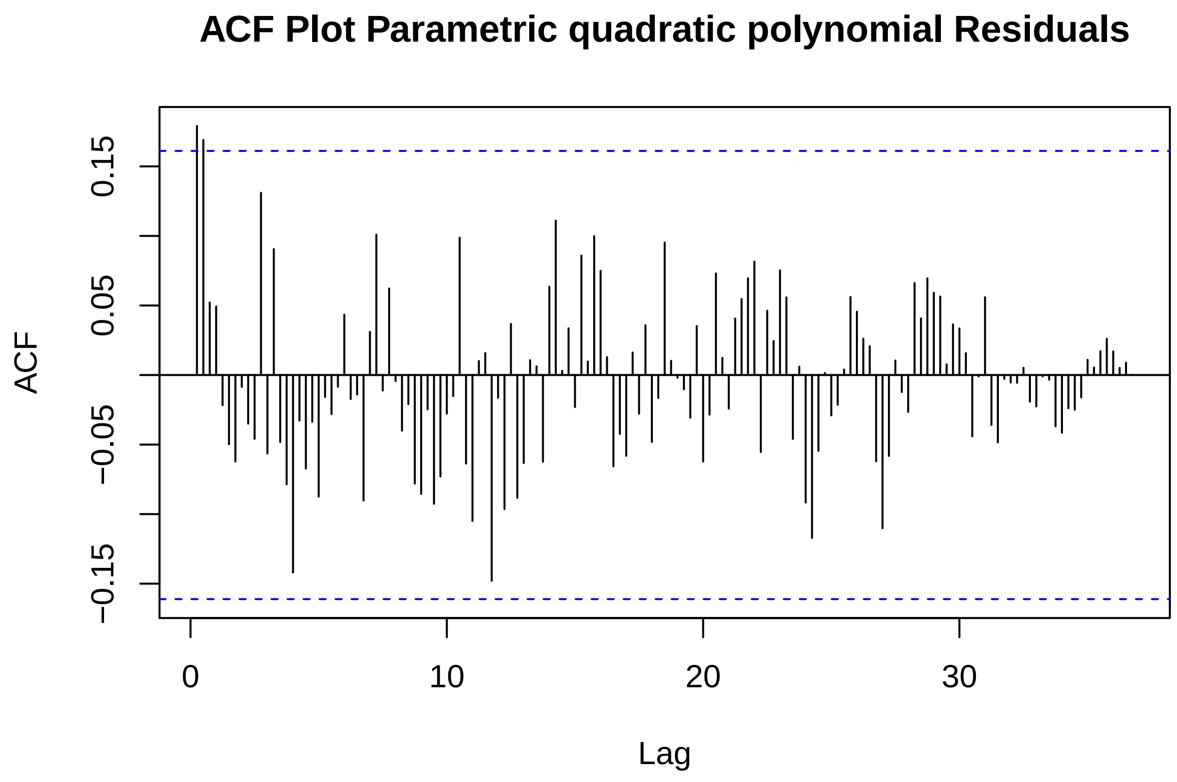
```r
lm.fit.quad = ts((fitted(lm.fit)),start=c(1980, 1),frequency=4)

ts.plot(temp,ylab="Temperature", main = "Parametric quadratic polynomial fitted on Original Data")
lines(lm.fit.quad,lwd=2,col="purple")
```



```r
dif.fit.lm = ts((temp-fitted(lm.fit)),start=c(1980, 1),frequency=4)
ts.plot(dif.fit.lm,ylab="Residual Process", main = "Parametric quadratic polynomial Residuals")
```



```r
acf(dif.fit.lm, lag.max = 400, main="ACF Plot Parametric quadratic polynomial Residuals")
```



```
anova_option = season(temp)-1
```

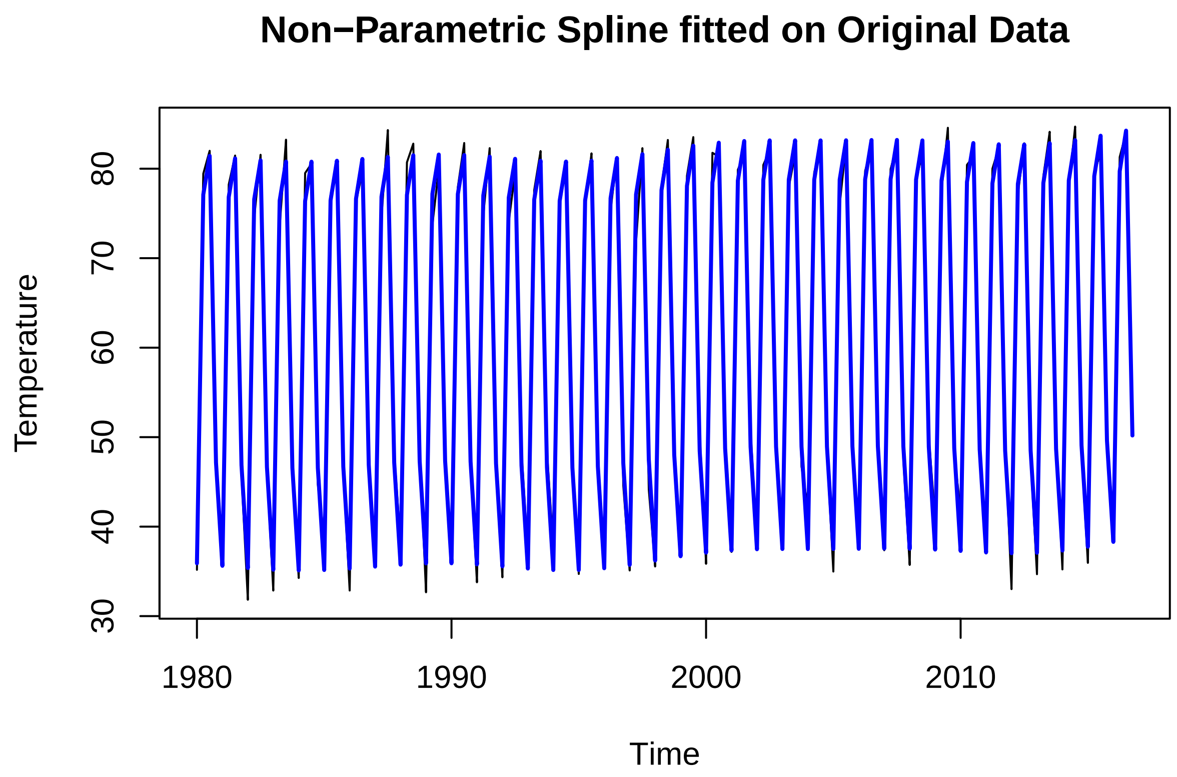
```
## Warning in Ops.factor(season(temp), 1): '-' not meaningful for factors
```

```
gam.fit = gam(temp~s(time.pts)+season(temp)-1)
summary(gam.fit)
```

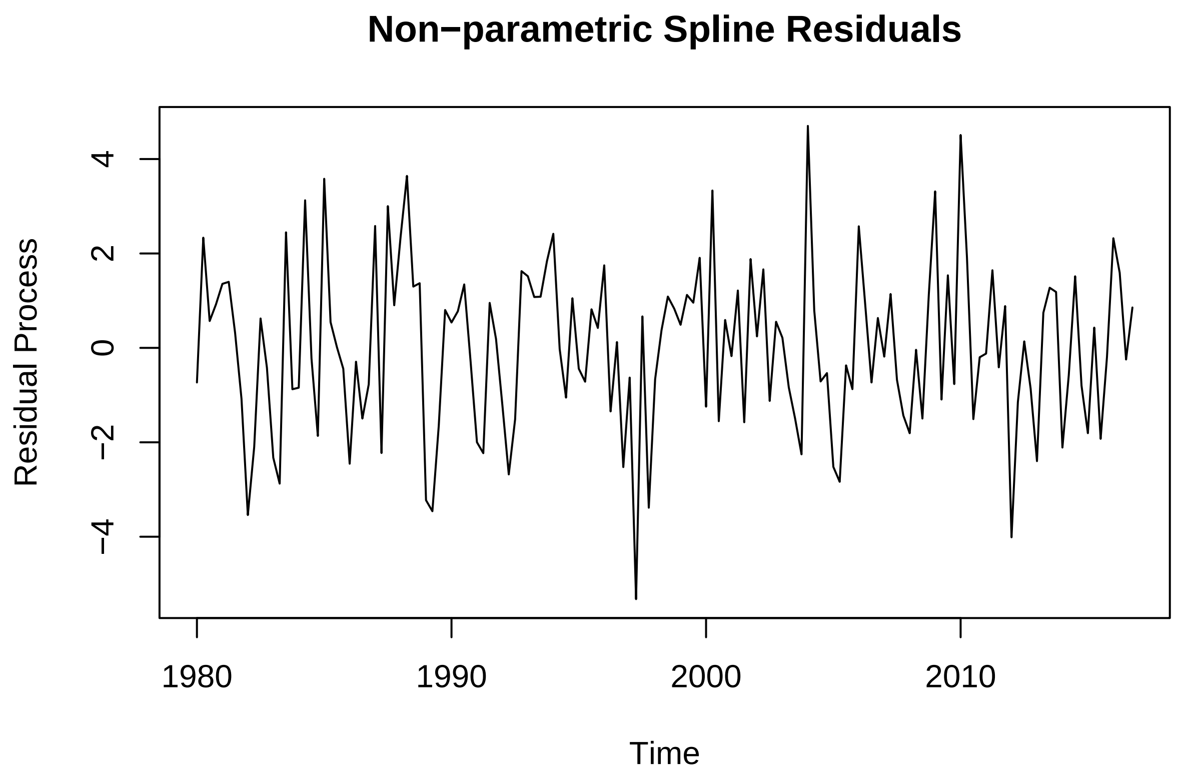
```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## temp ~ s(time.pts) + season(temp) - 1
##
## Parametric coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## season(temp)1Q  36.4605     0.2983   122.2   <2e-16 ***
## season(temp)2Q  77.7345     0.2981   260.7   <2e-16 ***
## season(temp)3Q  82.1283     0.2981   275.5   <2e-16 ***
## season(temp)4Q  47.9140     0.2983   160.6   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
```

```
##                  edf Ref.df     F  p-value
## s(time.pts) 7.227  8.249 5.955 2.28e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.991   Deviance explained = 99.9%
## GCV = 3.5582  Scale est. = 3.2883    n = 148
```

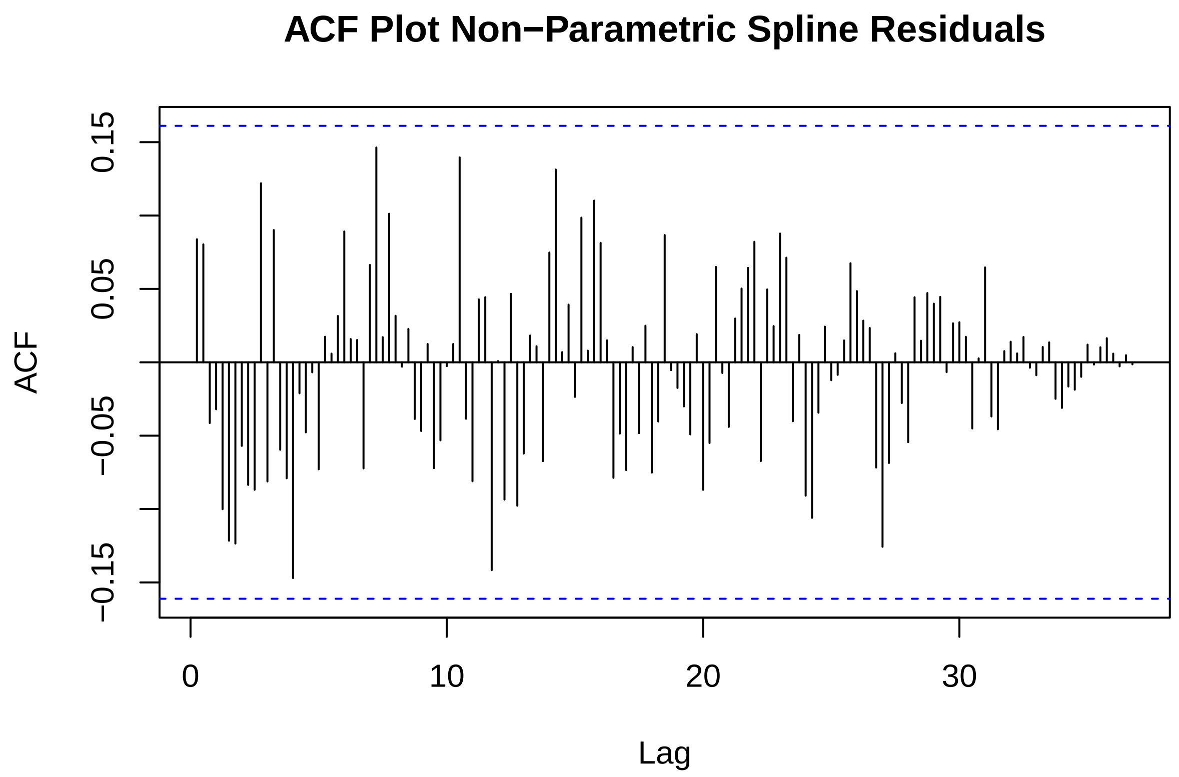
```
gam.fit.spline = ts((fitted(gam.fit)),start=c(1980, 1),frequency=4)

ts.plot(temp,ylab="Temperature", main = "Non-Parametric Spline fitted on Original Data")
lines(gam.fit.spline,lwd=2,col="blue")
```



```
dif.fit.gam = ts((temp-fitted(gam.fit)),start=c(1980, 1),frequency=4)
ts.plot(dif.fit.gam,ylab="Residual Process", main = "Non-parametric Spline Residuals")
```



```
acf(dif.fit.gam, lag.max = 400, main="ACF Plot Non-Parametric Spline Residuals")
```



**Response:2c**

When observing the fit, there does not appear to be a significant increase in the variability captured by including a trend model, whether it be parametric or non-parametric, as shown by the R^2 values in the summary output. In addition, for the parametric trend model, the coefficients were not significant, with p-values of 0.326, and 0.641.

When looking at the residuals for both plots, the variance appears to be constant, as well as the mean, indicating that stationarity conditions 1 & 2 are met. When looking at the ACF plots for both models, the autocorrelation bars rapidly decline to 0, and do not exceed the 95% confidence interval at higher lag values, so condition 3 is also met. The residual plots are plausibly stationary.

The model that seems to fit the best based on the current data would be the seasonality-only model, using the ANOVA approach. However, this would also mean that we would expect the long-term temperature trend to be non-existent. If we were to make long term predictions based on this model, it could be problematic, as the trend could change in the future, and the average temperature over time could increase or decrease.

## Question 2d: Prediction

Using the trend-seasonality models, predict the temperature data for 4 quarters ahead (Q1 to Q4 2016). Apply both one step ahead rolling predictions as well as predictions 8 steps ahead. Note that we have had an unusual heated summer this year. How do your predictions compare with the observed average temperature?

Hints:

- Keep in mind that modeling factors may require extra steps on the data preparation.

- To predict, you may want to rename the columns of your training data, you could use: setnames(your_data, old = c(), new = c()).
- You can use predict, or predict.gam for your predictions.

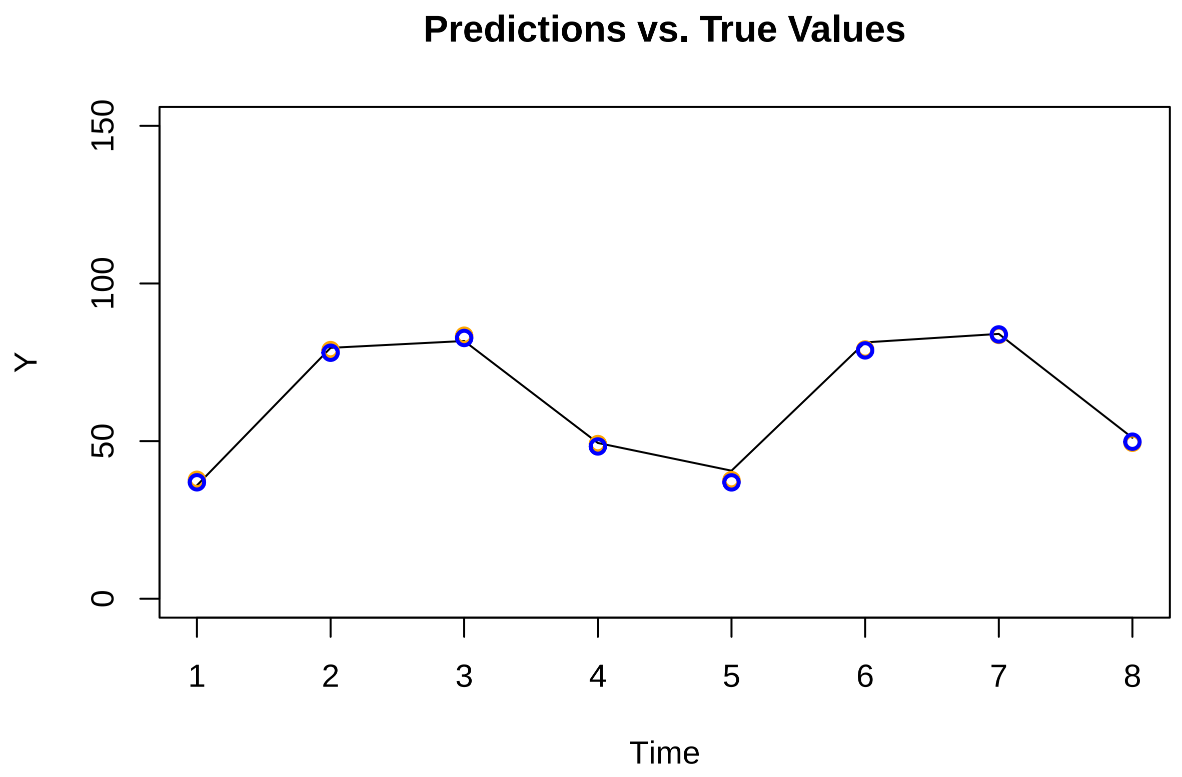
```
factors=season(temp)

season = createDummyFeatures(factors)

setnames(season, old = c('1Q', '2Q', '3Q', '4Q'), new = c('spring', 'summer', 'fall', 'winter'))

train_season=season

time.pts = c(1:length(temp))
time.pts = c(time.pts - min(time.pts))/max(time.pts)
x1_train = time.pts
x2_train = time.pts^2

train_data=cbind(x1_train,x2_train,train_season[,-1])

# One step ahead prediction
nfore = 8
lm.pred_n1 = NULL
gam.pred_n1 = NULL
for(f in 1: nfore)
{

model1 = dynlm(ts(temp[1:(length(temp)-(8-(f-1)))], frequency = 4, start = 1980)~., data=train_data[1:(
#you may add s()
model2 = gam(ts(temp[1:(length(temp)-(8-(f-1)))], frequency = 4, start = 1980)~s(x1_train)+summer+fall+

#Forecast

lm.pred_n1 = c(lm.pred_n1, predict(model1, train_data[(length(temp)-(7-(f-1))):(length(temp)-(7-(f-1)))
gam.pred_n1 = c(gam.pred_n1, predict(model2, train_data[(length(temp)-(7-(f-1))):(length(temp)-(7-(f-1)

}
lm.pred_n1
```

```
##      141      142      143      144      145      146      147      148
## 37.73480 78.87631 83.44009 49.06306 37.62125 79.09152 83.67453 49.46921
```

```
gam.pred_n1
```

```
##      141      142      143      144      145      146      147      148
## 36.93537 78.03178 82.71367 48.30231 36.91234 78.78495 83.84990 49.82106
```

```
plot(temp[(length(temp)-7):(length(temp))],type="l",ylab="Y",xlab="Time",
     main="Predictions vs. True Values",
     ylim=c(0,150))
points(lm.pred_n1,lwd=2,col="orange")
points(gam.pred_n1,lwd=2,col="blue")
```



**Response: 2d**

When observing the predictions vs true values plot, it appears that both predictions from using both methods were quite accurate compared to the actual values.