ISYE 6402 Homework 2

```
options(knitr.duplicate.label = "allow")
```

Part 1: Currency Conversion Analysis

Background

In this problem, we will study fluctuations in currency exchange rate over time.

File `USD-JPY.csv` download contains the daily exchange rate of USD/JPY from January 2000 through May 31st 2022. We will aggregate the data on a weekly basis, by taking the average rate within each week. The time series of interest is the weekly currency exchange. We will analyze this time series and its first order difference.

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()`

```
fpath <- "USD-JPY.csv"
df <- read.csv(fpath, head = TRUE)
```

Here we upload the libraries needed the this data analysis:

```
library(mgcv)
library(lubridate)
library(dplyr)
```

To prepare the data, run the following code snippet. First, aggregate by week:

```
df$date <- as.Date(df$Date, format='%Y-%m-%d')
df$week <- floor_date(df$date, "week")
df <- df[, c("week", "jpy")]
```

We now form the weekly aggrgated time series to use for data exploration! Please note that we will analyze the weekly aggregated data not the original (daily) data.

```
agg <- aggregate(x = df$jpy, by = list(df$week), FUN = mean)
colnames(agg) <- c("week", "jpy")
jpy.ts <- ts(agg$jpy, start = 2000, freq = 52)
```

Please use the `jpy` series to code and answer the following questions.

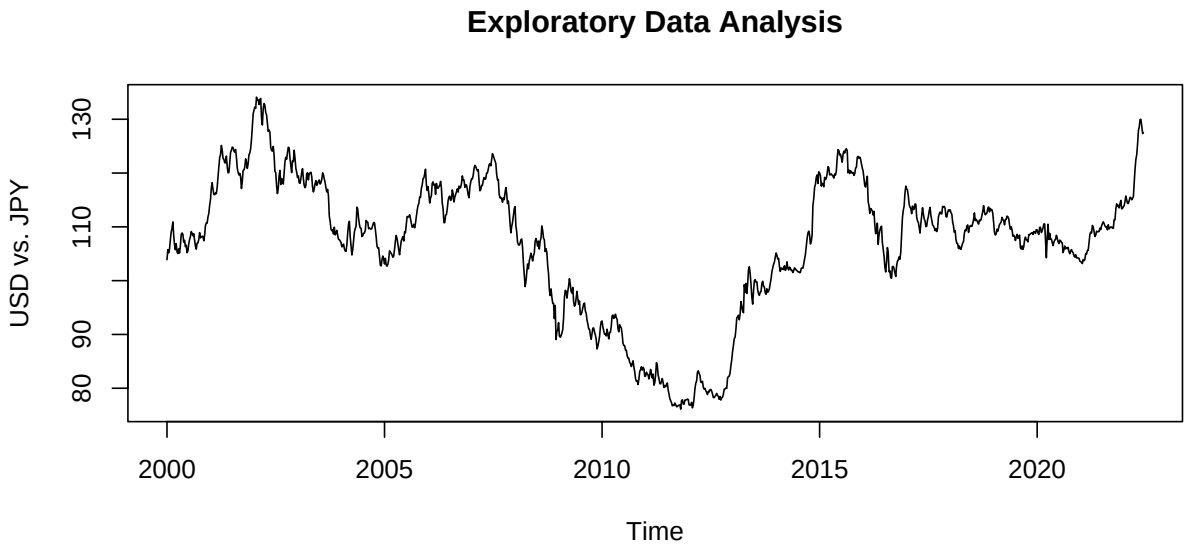
Question 1a: Exploratory Data Analysis

Before exploring the data, can you infer the data features from what you know about the USD-JPY currency exchange? Next plot the Time Series and ACF plots of the weekly data. Comment on the main features, and identify what (if any) assumptions of stationarity are violated.

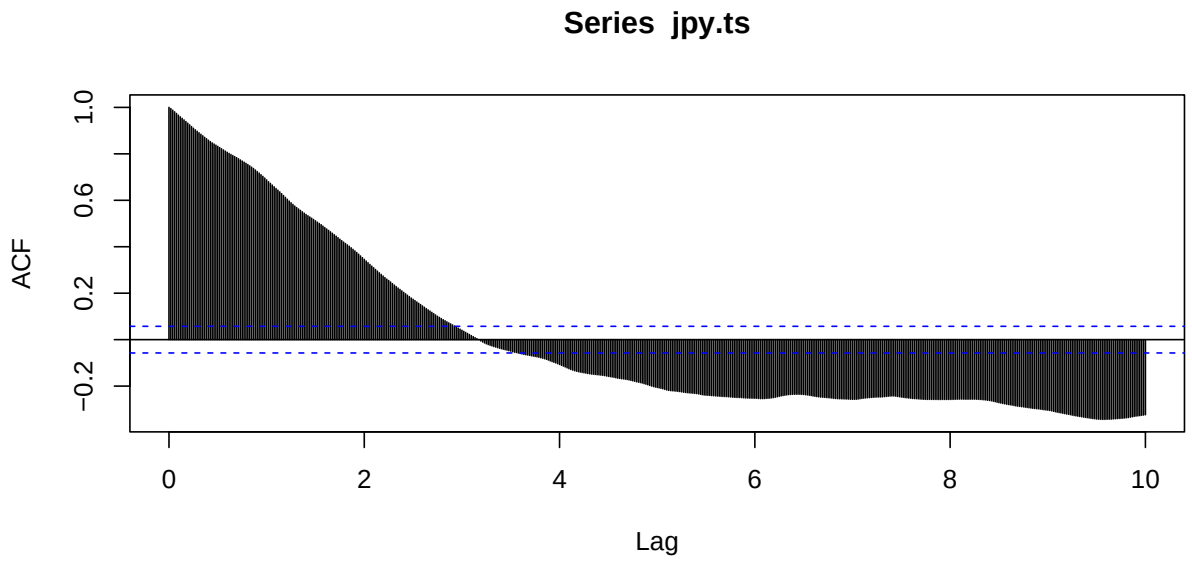
Which type of model do you think will fit the data better: the trend or seasonality fitting model? Provide details for your response.

*Response: General Insights on the USD-JPY Currency Rate*The rate at which USD currency is exchanged for JPY currency is USD-JPY Currency Rate.If demand of a certain currency than its supply, the currency becomes more valuable. Whereas, if demand of a certain currency is low and its supply is more, it will become less valuable.

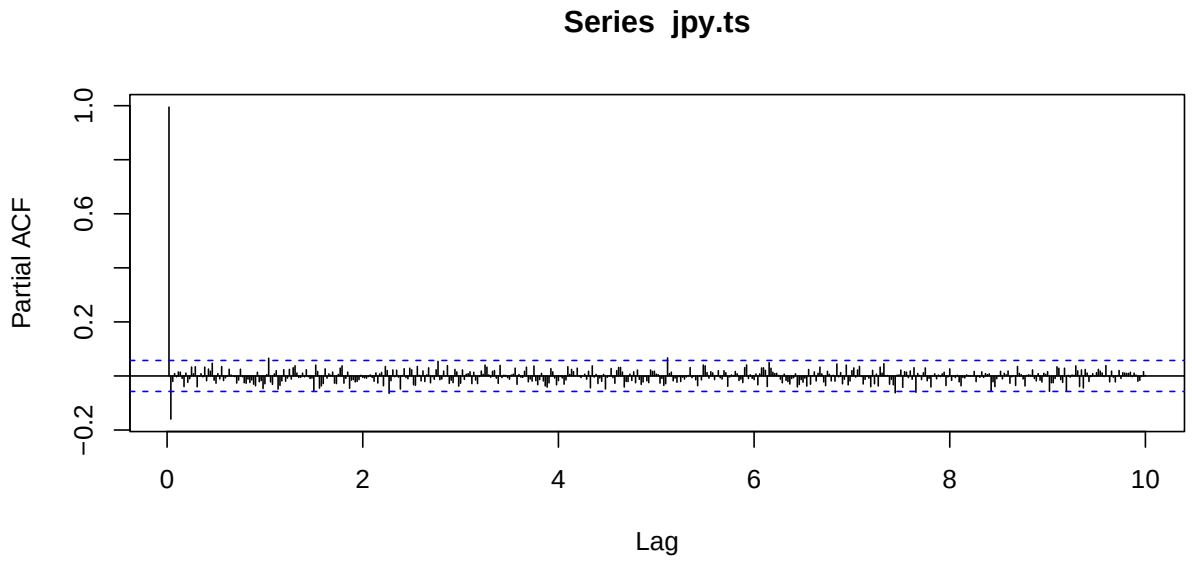
```
ts.plot(jpy.ts,main="Exploratory Data Analysis", ylab="USD vs. JPY")
```




```
#Plotting ACF
acf(jpy.ts, lag.max = 52*10)
```




```
#Plotting PACF
pacf(jpy.ts, lag.max = 52*10)
```



*Response: General Insights from the Graphical Analysis*The time series plot shows an upward trend in the beginning. It hits the highest value in Q1 2002. From 2010 Q4 to 2012 Q4, JPY struggles and hit lowest value in this time frame.I most recent time, JPY has recovered, show growth and retain to its value when the graph started recorded its values. Looking at ACF plot, it can be observed that there is significant autocorrelation found in most lags. PACF paints a different story. Most observations are shown within the limit. This can be inferred that the time series fail the stationarity assumptions of constant mean and autocorrelation structure.

Question 1b: Trend Estimation

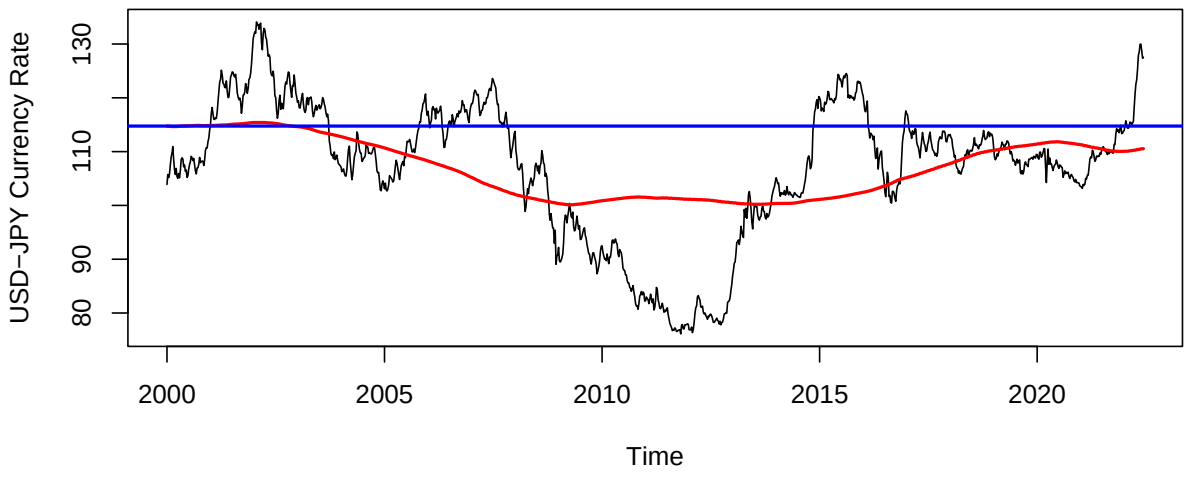
Fit the following trend estimation models:

- Moving Average
- Parametric Quadratic Polynomial
- Local Polynomial
- Splines Smoothing

Overlay the fitted values on the original time series. Plot the residuals with respect to time for each model. Plot the ACF of the residuals for each model also. Comment on the four models fit and on the appropriateness of the stationarity assumption of the residuals.

```
time.tnd = c(1:length(jpy.ts))
time.tnd = c(time.tnd - min(time.tnd))/max(time.tnd)
```

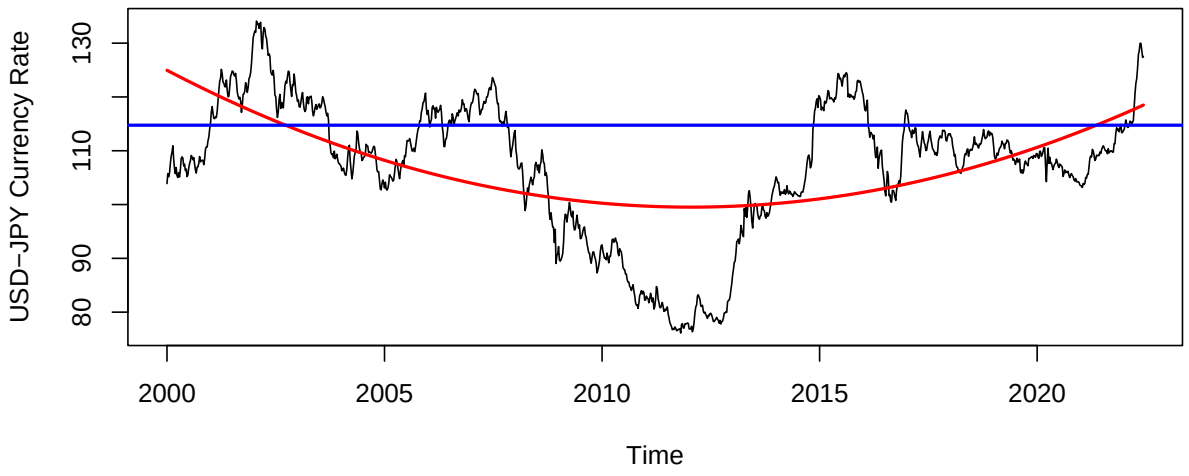
```
#Moving Average
movave.fit = ksmooth(time.tnd, jpy.ts, kernel = "box")
jpy.ts.fit.movave = ts(movave.fit$y,start=2000,frequency=52)
ts.plot(jpy.ts,ylab="USD-JPY Currency Rate")
lines(jpy.ts.fit.movave,lwd=2,col="red")
abline(jpy.ts.fit.movave[1],0,lwd=2,col="blue")
```




```
# Quadratic Polynomial Parametric
x1 = time.tnd
x2 = time.tnd^2
lm.fit = lm(jpy.ts ~ x1 + x2)
summary(lm.fit)
```

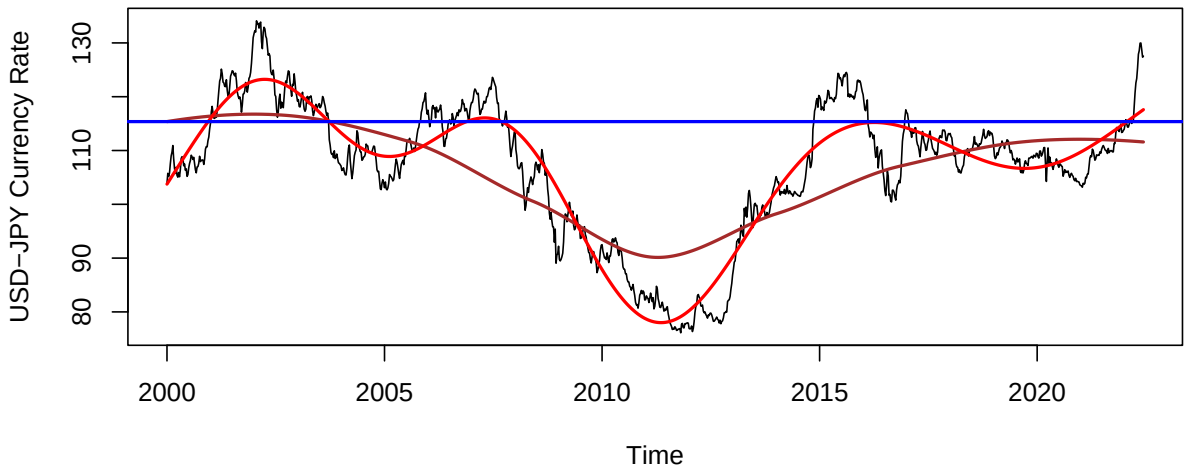
```
##
## Call:
## lm(formula = jpy.ts ~ x1 + x2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -23.4180  -5.9269   0.7775   6.9216  22.8259
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 124.9589     0.9364  133.45   <2e-16 ***
## x1          -94.8930     4.3289  -21.92   <2e-16 ***
## x2           88.4978     4.1948   21.10   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.69 on 1165 degrees of freedom
## Multiple R-squared:  0.2921, Adjusted R-squared:  0.2909
## F-statistic: 240.4 on 2 and 1165 DF,  p-value: < 2.2e-16
```

```
jpy.ts.fit.lm = ts(fitted(lm.fit),start=2000,frequency=52)
ts.plot(jpy.ts,ylab="USD-JPY Currency Rate")
lines(jpy.ts.fit.lm,lwd=2,col="red")
abline(jpy.ts.fit.movave[1],0,lwd=2,col="blue")
```

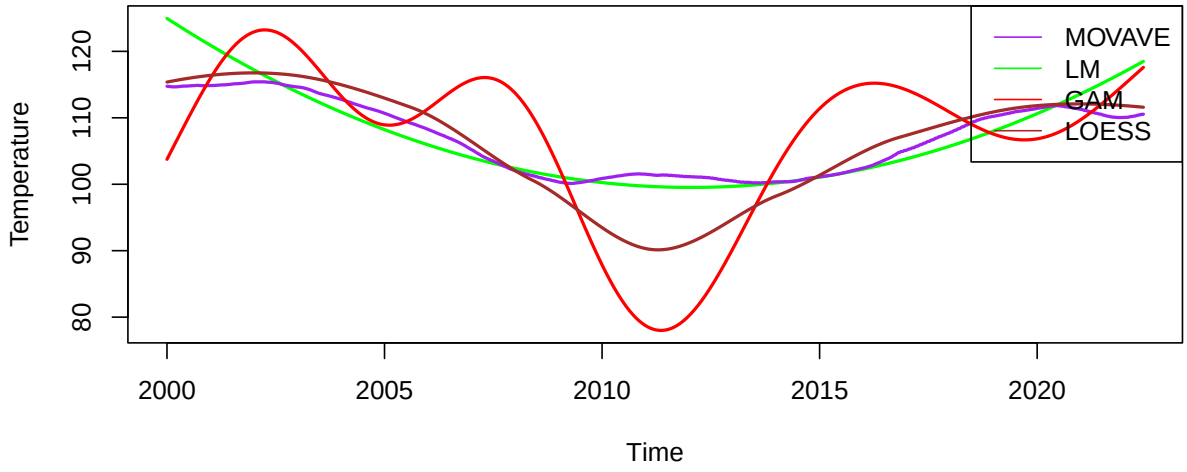



```
#Fit Trend Non-Parametric Regression
# Local Polynomial Trend Estimation
loc.fit = loess(jpy.ts~time.tnd)
jpy.ts.fit.loc =ts(fitted(loc.fit),start=2000,frequency=52)
## Splines Trend Estimation
gam.fit = gam(jpy.ts~s(time.tnd))
jpy.ts.fit.gam = ts(fitted(gam.fit),start=2000,frequency=52)
## Identifying a Trend
```

```
ts.plot(jpy.ts,ylab="USD-JPY Currency Rate")
lines(jpy.ts.fit.loc,lwd=2,col="brown")
lines(jpy.ts.fit.gam,lwd=2,col="red")
abline(jpy.ts.fit.loc[1],0,lwd=2,col="blue")
```



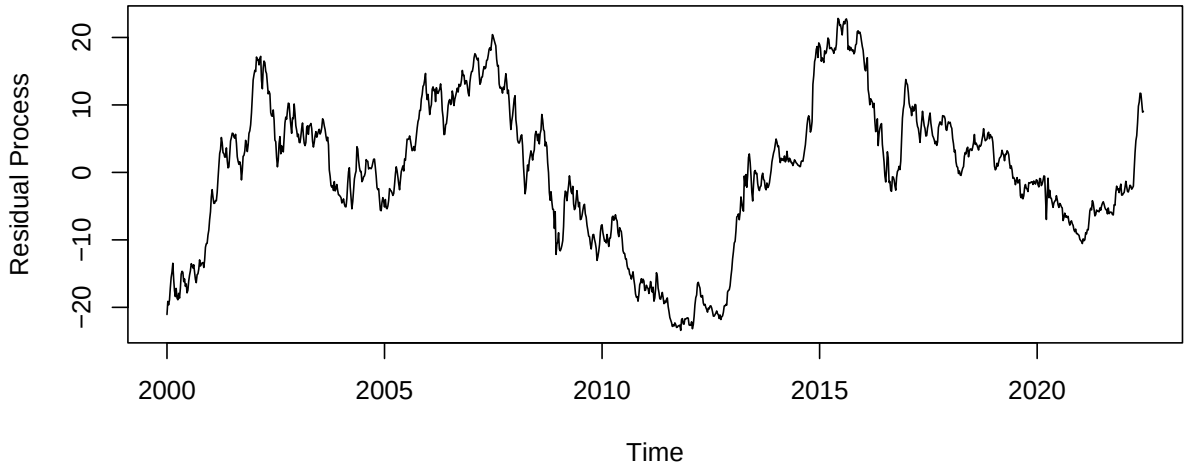

```
# Comparison all estimated trends
total = c(jpy.ts.fit.movave,jpy.ts.fit.lm,jpy.ts.fit.gam,jpy.ts.fit.loc)
ylim= c(min(total),max(total))
ts.plot(jpy.ts.fit.lm,lwd=2,col="green",ylim=ylim,ylab="Temperature")
lines(jpy.ts.fit.movave,lwd=2,col="purple")
lines(jpy.ts.fit.gam,lwd=2,col="red")
lines(jpy.ts.fit.loc,lwd=2,col="brown")
legend(x="topright",y=1.4,legend=c("MOVAVE","LM","GAM","LOESS"),lty = 1, col=c("purple","green","red","
```



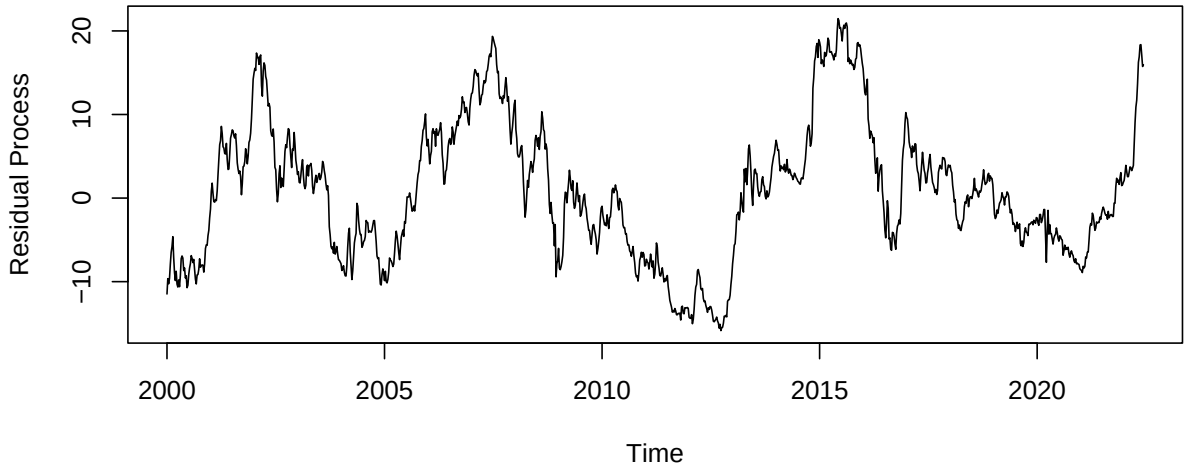
*Response: Comparison of the fitted trend models:*Most commonly, it is observed that local polynomial smoothing performs netter than kernel regression on the boundaries. For time-series analysis, local polynomial & splines regression deliver better in estimating trends in time series. Based on graph above this can be observed that the moving average slightly capturing some of the seasonality thus not a good estimate. Splines regression capture the short-term up and down trend over time. The fit using local polynomial regression shows more smooth in the trend; generally it also shows an a downward parabola pattern over time.

```
dif.fit.movave = ts((jpy.ts-movave.fit$y),start=2000,frequency=52)
ts.plot(dif.fit.movave,ylab="Residual Process")
```

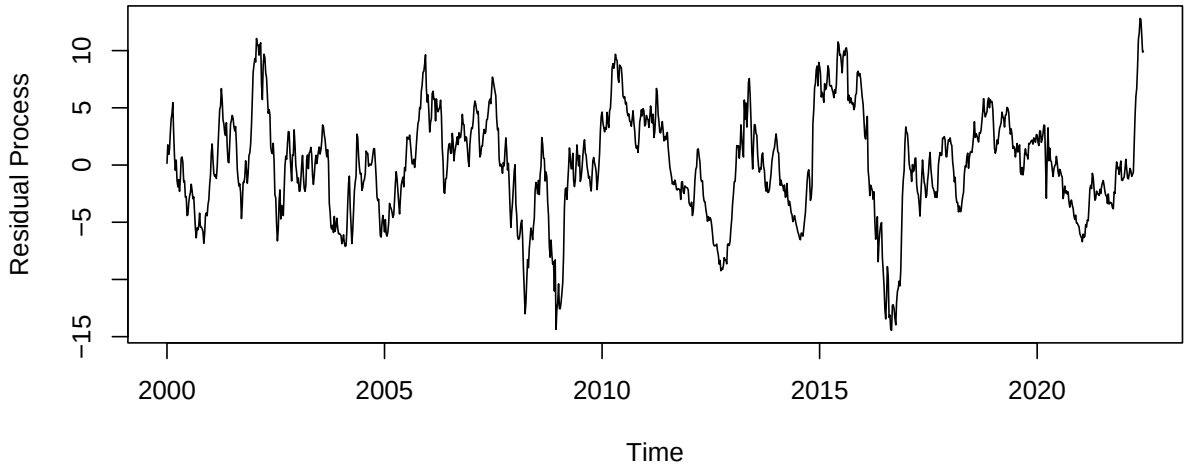

```
dif.fit.lm = ts((jpy.ts-fitted(lm.fit)),start=2000,frequency=52)
ts.plot(dif.fit.lm,ylab="Residual Process")
```



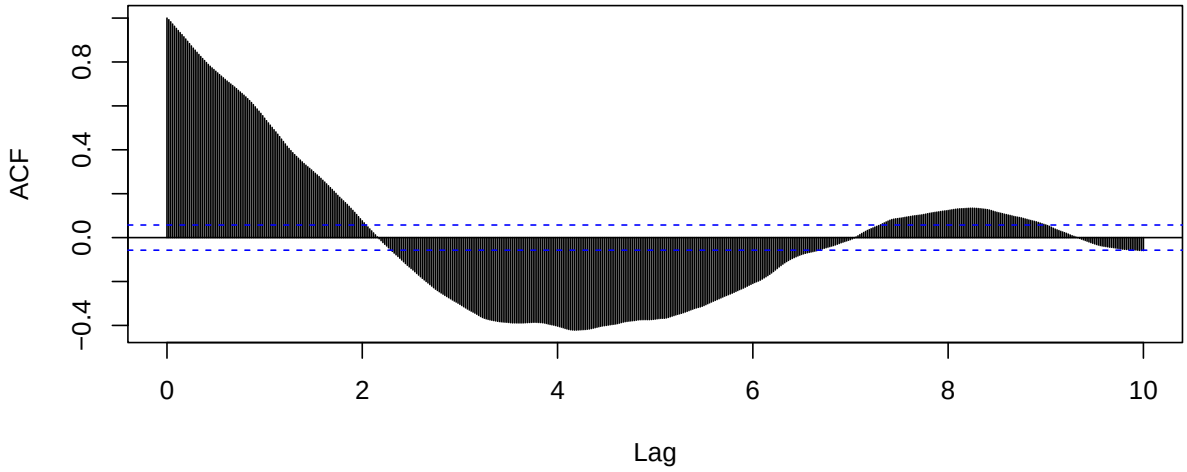

```
dif.fit.loc = ts((jpy.ts-fitted(loc.fit)),start=2000,frequency=52)
ts.plot(dif.fit.loc,ylab="Residual Process")
```



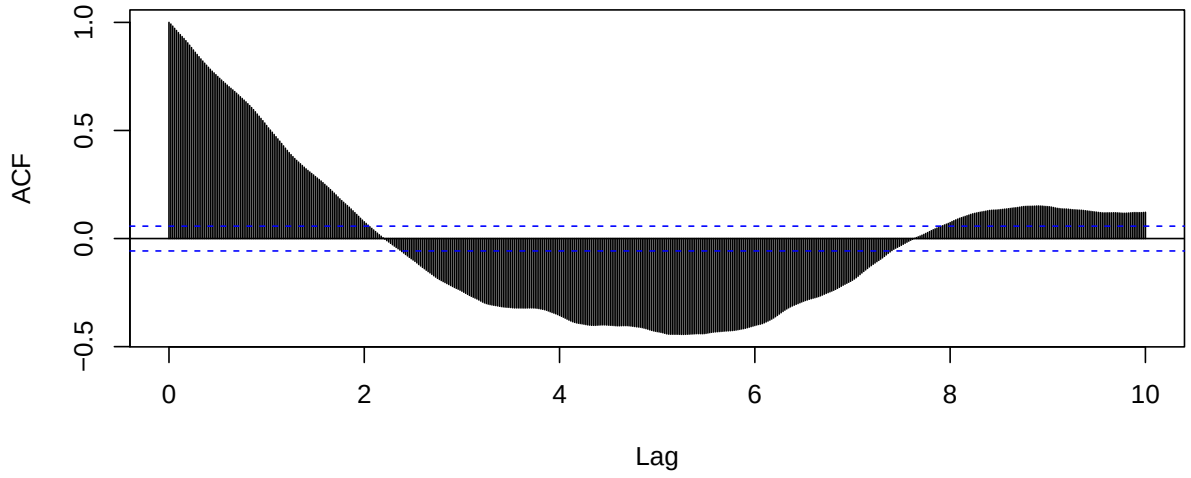

```
dif.fit.gam = ts((jpy.ts-fitted(gam.fit)),start=2000,frequency=52)
ts.plot(dif.fit.gam,ylab="Residual Process")
```



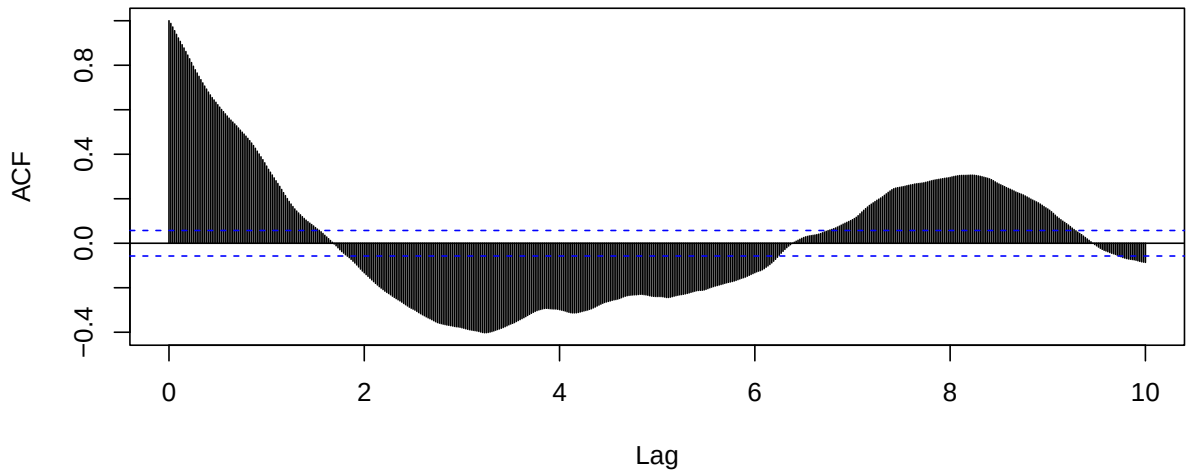

```
# Plotting ACF residuals
acf(dif.fit.movave,lag.max=52*10,main="")
```



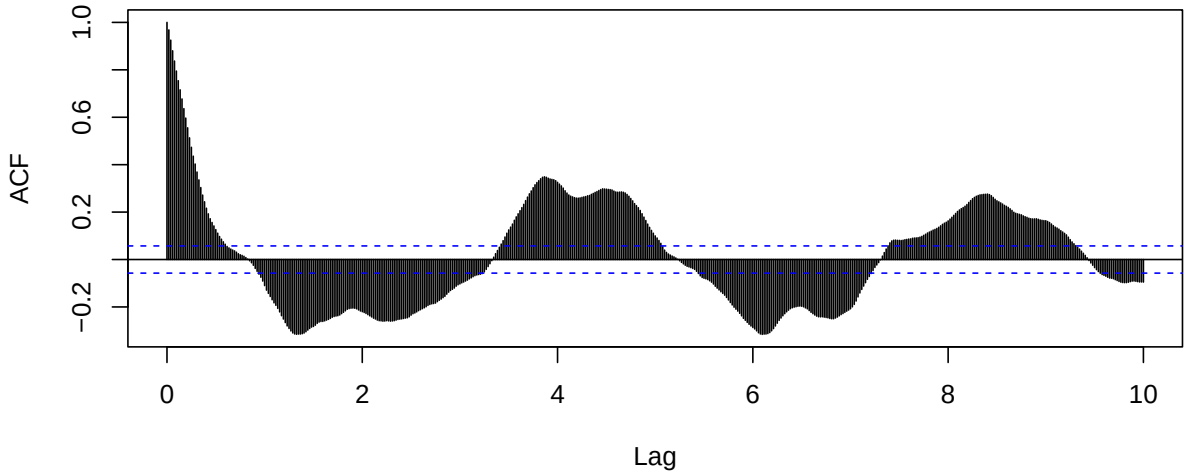

```
acf(dif.fit.lm,lag.max=52*10,main="")
```




```
acf(dif.fit.loc,lag.max=52*10,main="")
```




```
acf(dif.fit.gam,lag.max=52*10,main="")
```



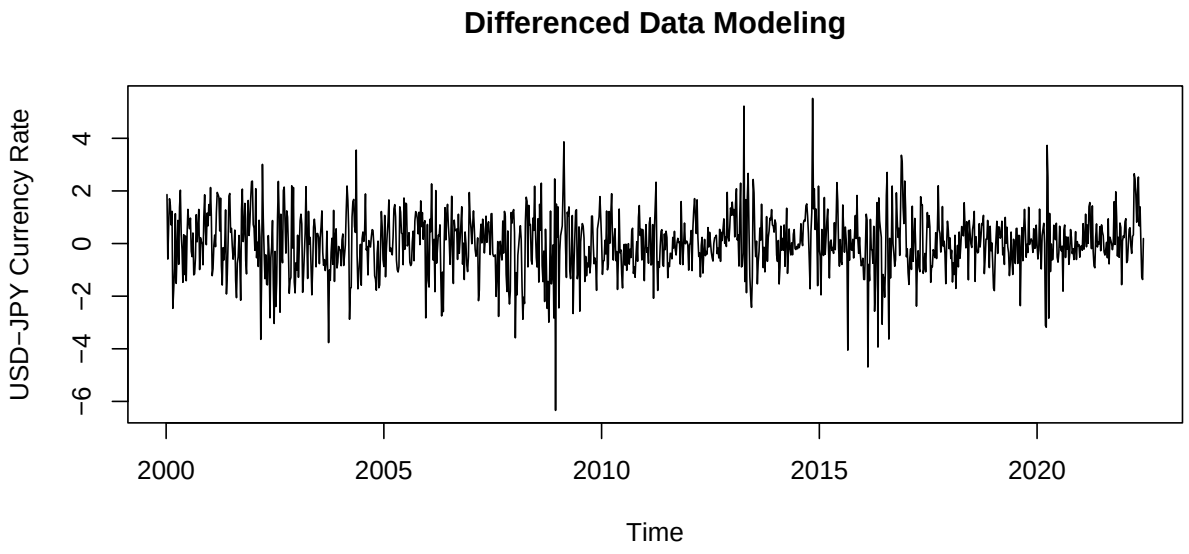
*Response: Appropriateness of the trend model for stationarity*Moving average,Parametric Quadraric Polynomial & Local Polynomial, ACF plots show minimum cyclical patterns and hence no stationarity. The ACF plot for Splines shows a slight seasonality pattern and the pattern repeats for each block of lags specified at around on the around 3, 5 & 7 on the x-axis.However, none of these observations about residual plots guarantee stationarity.

Question 1c: Differenced Data Modeling

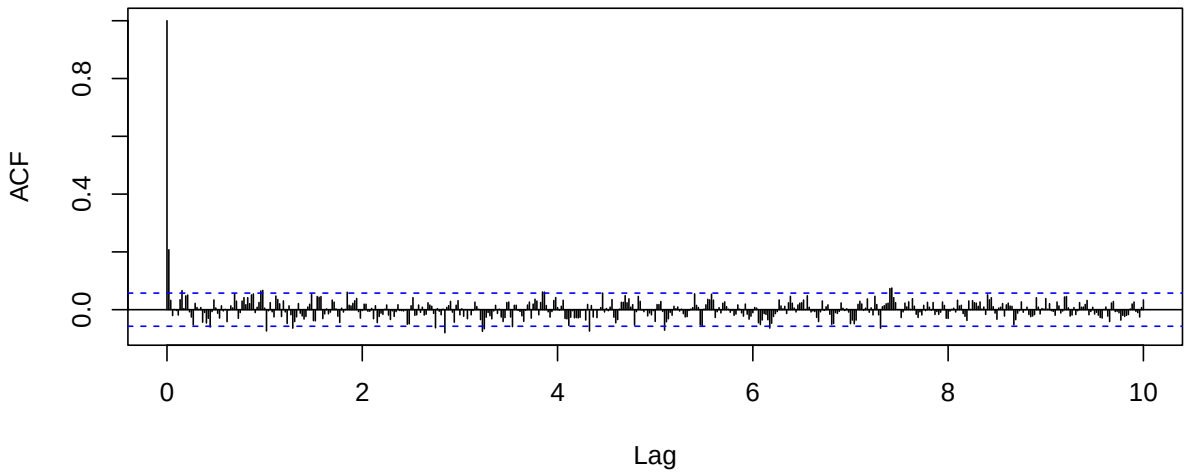
Now plot the difference time series and its ACF plot. Apply the four trend models in Question 1b to the differenced time series. What can you conclude about the difference data in terms of stationarity? Which model would you recommend to apply (trend removal via fitting trend vs differencing) such that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

Hint: When TS data are differenced, the resulting data set will have an NA in the first data element due to the differencing.

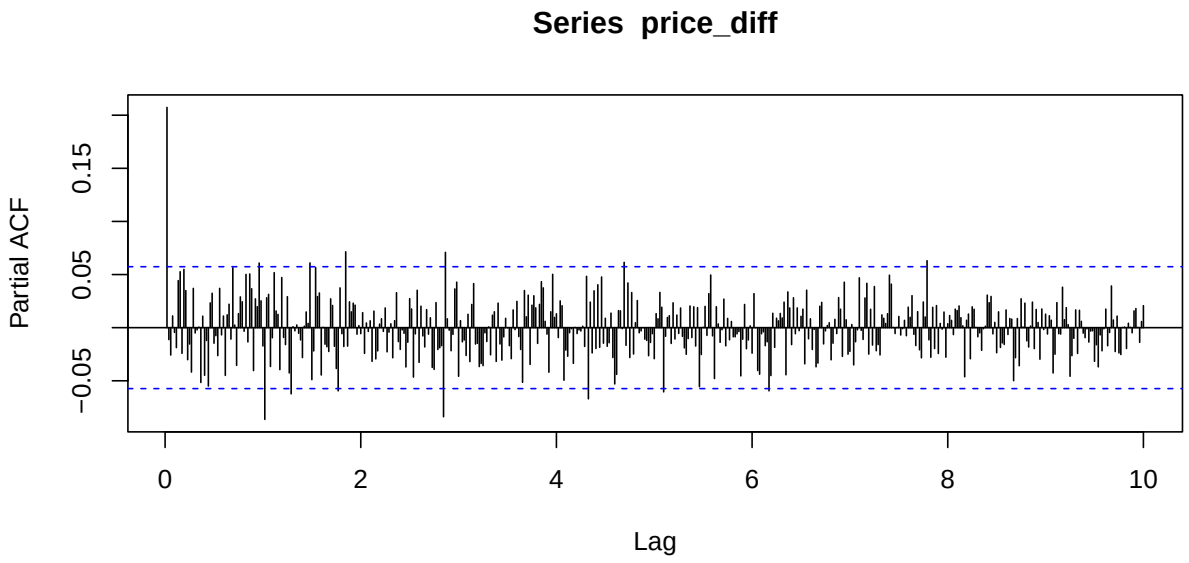
```
price_diff <- diff(jpy.ts)
ts.plot(price_diff,ylab="USD-JPY Currency Rate", main ="Differenced Data Modeling")
```




```
acf(price_diff,lag.max=52*10,main="")
```

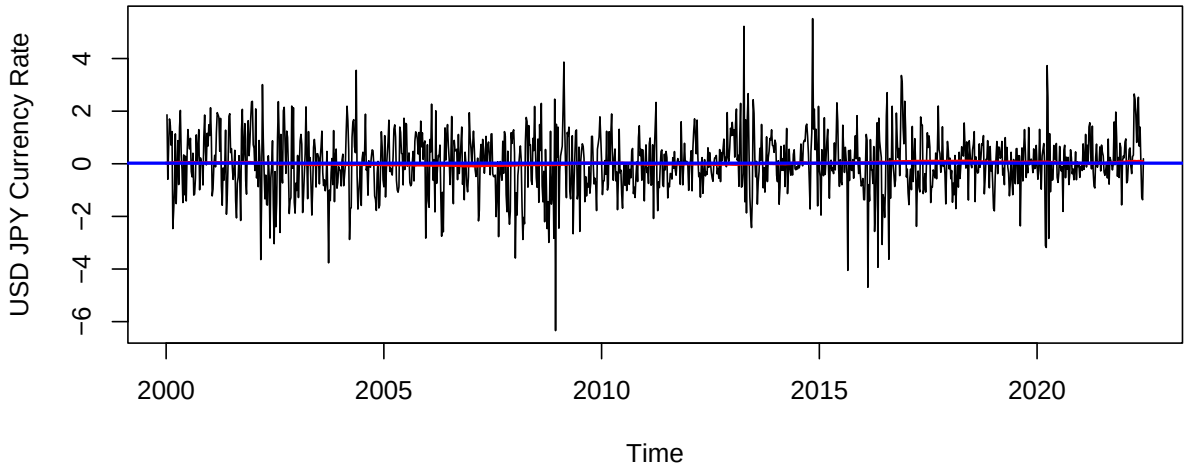



```
pacf(price_diff, lag.max = 52*10)
```




```
# Converting X-axis by scaling 0-1
time.tnd = c(1:length(price_diff))
time.tnd = c(time.tnd - min(time.tnd))/max(time.tnd)
```

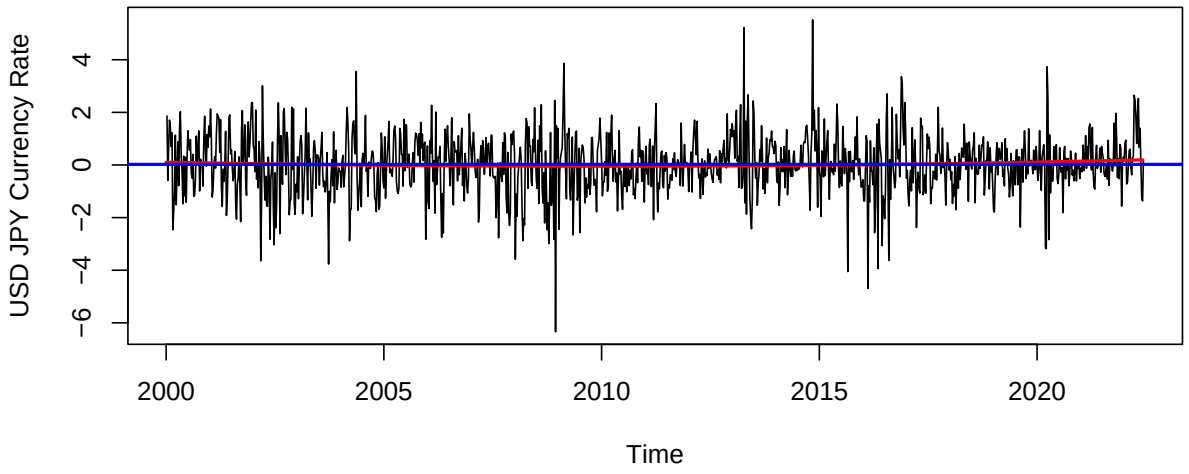
```
# Fitting Moving Average
movave.fit = ksmooth(time.tnd, price_diff, kernel = "box")
jpy.ts.fit.movave = ts(movave.fit$y,start=2000,frequency=52)
ts.plot(price_diff,ylab="USD JPY Currency Rate")
lines(jpy.ts.fit.movave,lwd=2,col="red")
abline(jpy.ts.fit.movave[1],0,lwd=2,col="blue")
```



```r
# Fitting Parametric Quadraric Polynomial
x1 = time.tnd
x2 = time.tnd^2
lm.fit = lm(price_diff ~ x1 + x2)
summary(lm.fit)
```

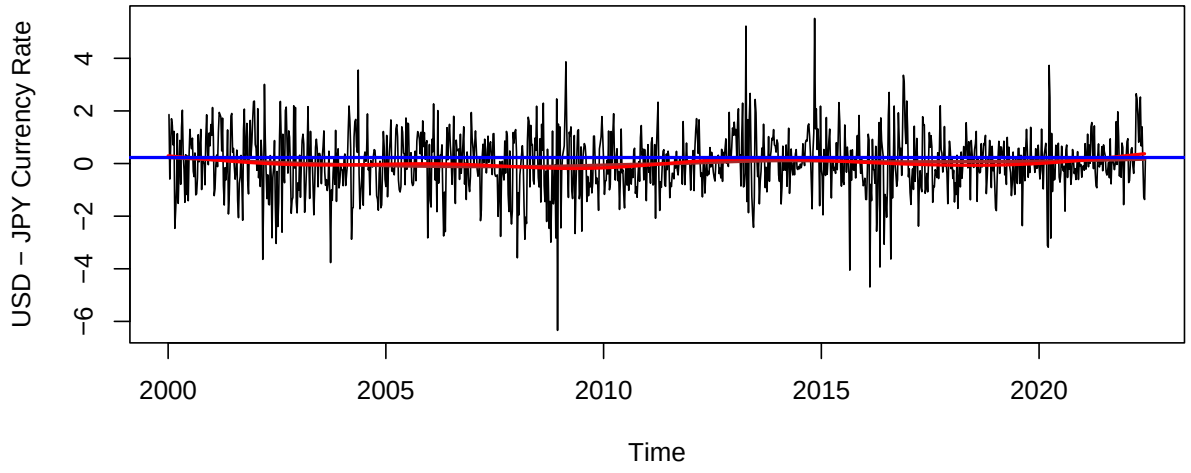
```
##
## Call:
## lm(formula = price_diff ~ x1 + x2)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.2961 -0.6619  0.0191  0.6543  5.5266
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.10450    0.09974   1.048   0.2950
## x1          -0.68566    0.46108  -1.487   0.1373
## x2           0.77587    0.44680   1.737   0.0827 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.138 on 1164 degrees of freedom
## Multiple R-squared:  0.003099,   Adjusted R-squared:  0.001386
## F-statistic: 1.809 on 2 and 1164 DF,  p-value: 0.1643
```

```r
jpy.ts.fit.lm = ts(fitted(lm.fit),start=2000,frequency=52)
ts.plot(price_diff,ylab="USD JPY Currency Rate")
lines(jpy.ts.fit.lm,lwd=2,col="red")
abline(jpy.ts.fit.movave[1],0,lwd=2,col="blue")
```

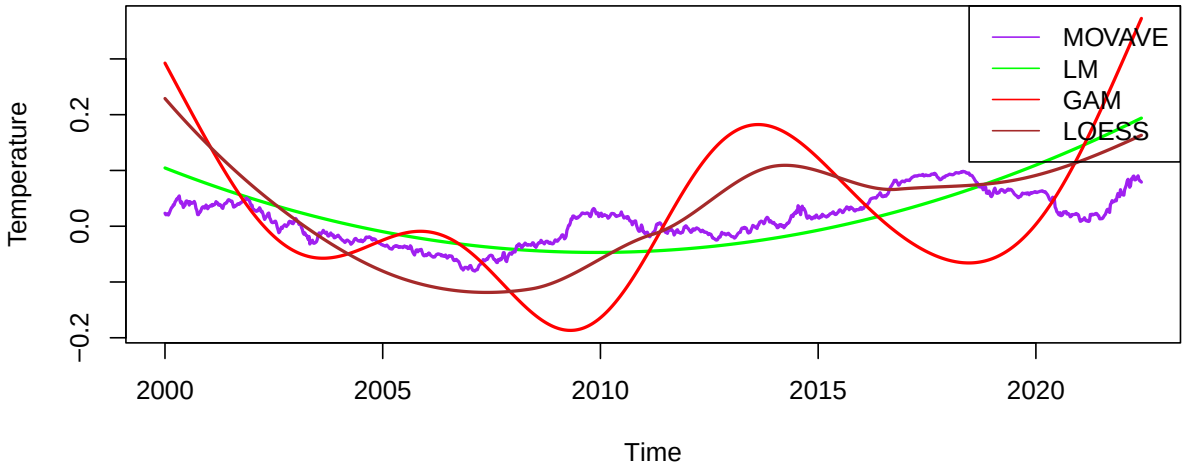


```r
# Fitting a trend using non-parametric regression
# Local Polynomial Trend Estimation
loc.fit = loess(price_diff ~ time.tnd)
jpy.ts.fit.loc =ts(fitted(loc.fit),start=2000,frequency=52)
# Splines Trend Estimation
gam.fit = gam(price_diff ~ s(time.tnd))
jpy.ts.fit.gam = ts(fitted(gam.fit),start=2000,frequency=52)
```

```r
#Identifying a Trend
ts.plot(price_diff,ylab="USD - JPY Currency Rate")
lines(jpy.ts.fit.loc,lwd=2,col="brown")
lines(jpy.ts.fit.gam,lwd=2,col="red")
abline(jpy.ts.fit.loc[1],0,lwd=2,col="blue")
```




```r
#Comparing all estimated trends
Total = c(jpy.ts.fit.movave, jpy.ts.fit.lm, jpy.ts.fit.gam, jpy.ts.fit.loc)
ylim= c(min(Total),max(Total))
ts.plot(jpy.ts.fit.lm,lwd=2,col="green",ylim=ylim,ylab="Temperature")
lines(jpy.ts.fit.movave,lwd=2,col="purple")
lines(jpy.ts.fit.gam,lwd=2,col="red")
lines(jpy.ts.fit.loc,lwd=2,col="brown")
legend(x="topright",y=1.4,legend=c("MOVAVE","LM","GAM","LOESS"),lty = 1, col=c("purple","green","red","
```



Response: Comments about the stationarity of the difference data: ACF & PACF both plots for residual plot refelects stationarity of the difference data, which is appearnce wise different from the original data initially loaded.

Part 2: Temperature Analysis

Background

In this problem, we will analyze aggregated temperature data.

Data *Everest Temp Jan-Mar 2021.csv* contains the hourly average temperature at the Mount Everest Base Camp for the months of January to March 2021. Run the following code to prepare the data for analysis:

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 `Temp` data column.

```
fpath <- "Everest Temp Jan-Mar 2021.csv"
df <- read.csv(fpath, head = TRUE)
```

Here are the libraries you will need:

```
library(mgcv)
library(TSA)
library(dynlm)
```

Run the following code to prepare the data for analysis:

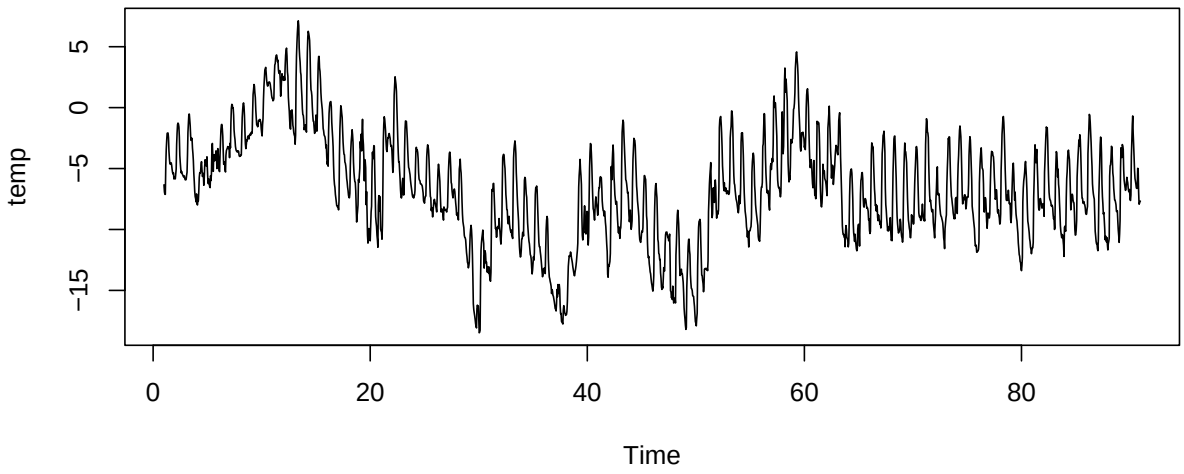
```
df$timestamp<-ymd_hms(df$timestamp)
temp <- ts(df$temp, freq = 24)

datetime<-ts(df$timestamp)
```

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, comment if you believe the differenced data is more appropriate for use in fitting the data. Support your response with a graphical analysis.

Hint: Make sure to use the appropriate differenced data.

```
ts.plot(temp)
```



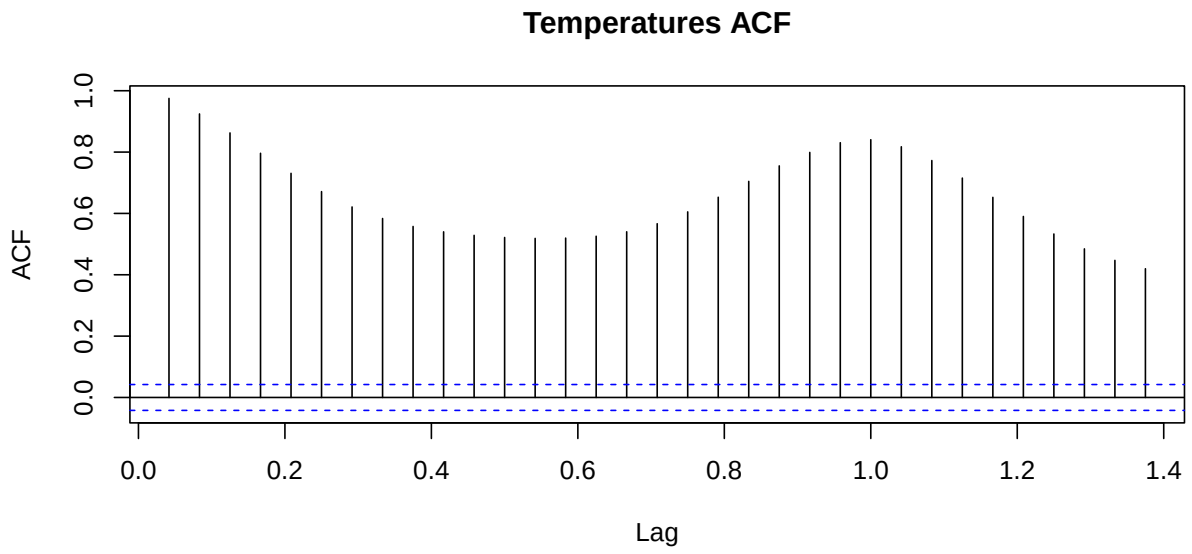

```
#Getting Max & Min Readings
time(temp)[which.max(temp)]
```

```
## [1] 13.375
```

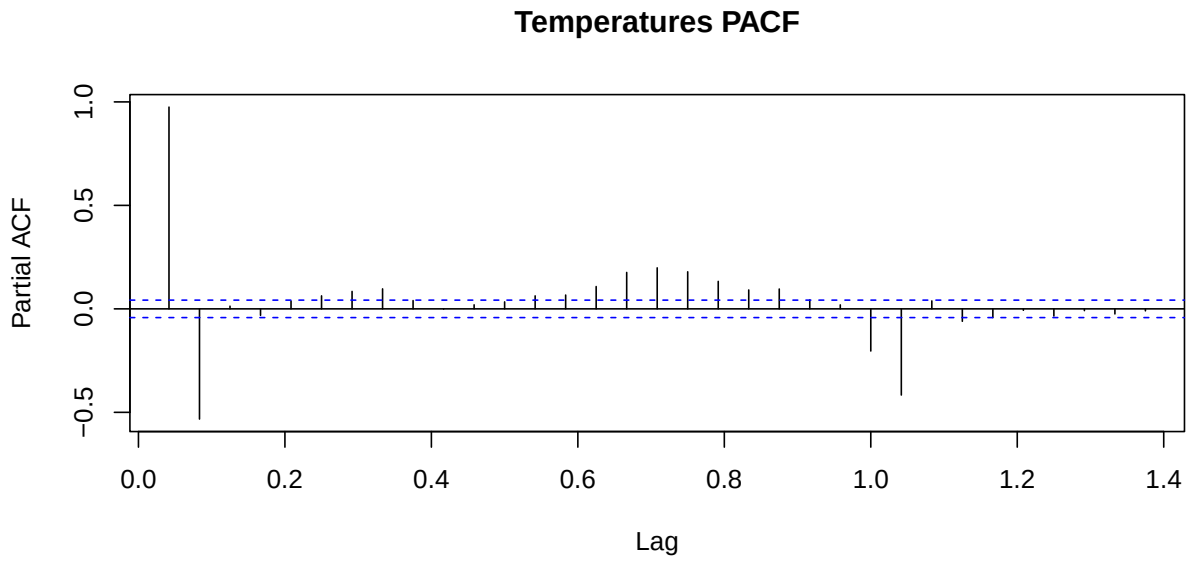
```
time(temp)[which.min(temp)]
```

```
## [1] 30.04167
```

```
#Plot the Autocorrelation Function
acf(temp,main="Temperatures ACF")
```

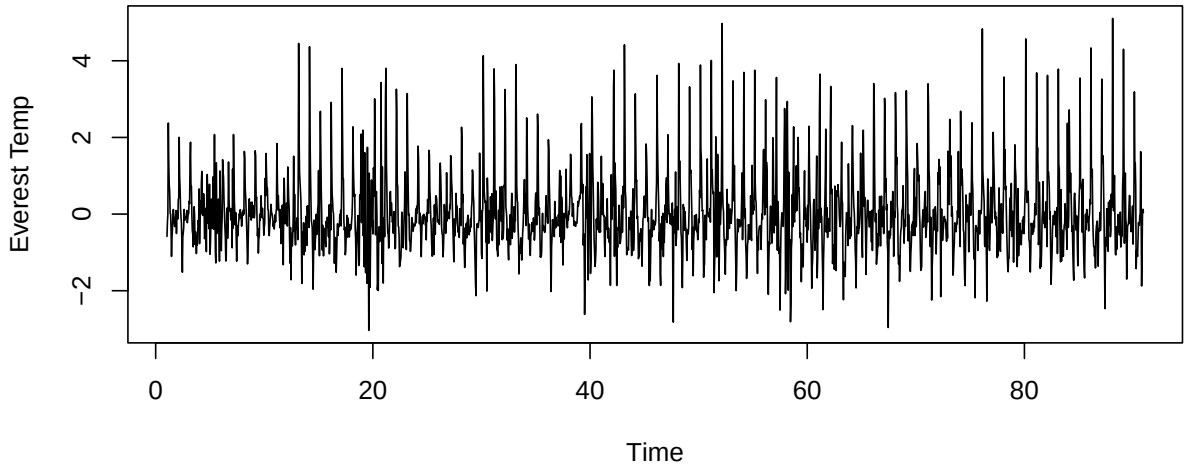



```
#Plot the Partial Autocorrelation Function
pacf(temp,main="Temperatures PACF")
```



*Response: Comments about the time series and ACF plots of the original time series*The ACF plot shows a seasonality pattern and the pattern repeats for each block of lags. The time series plot looks like it has constant mean for all time points and has a finite variance. However, the covariance function does change when shifted in time since majority of lags are out of the band in the ACF plot

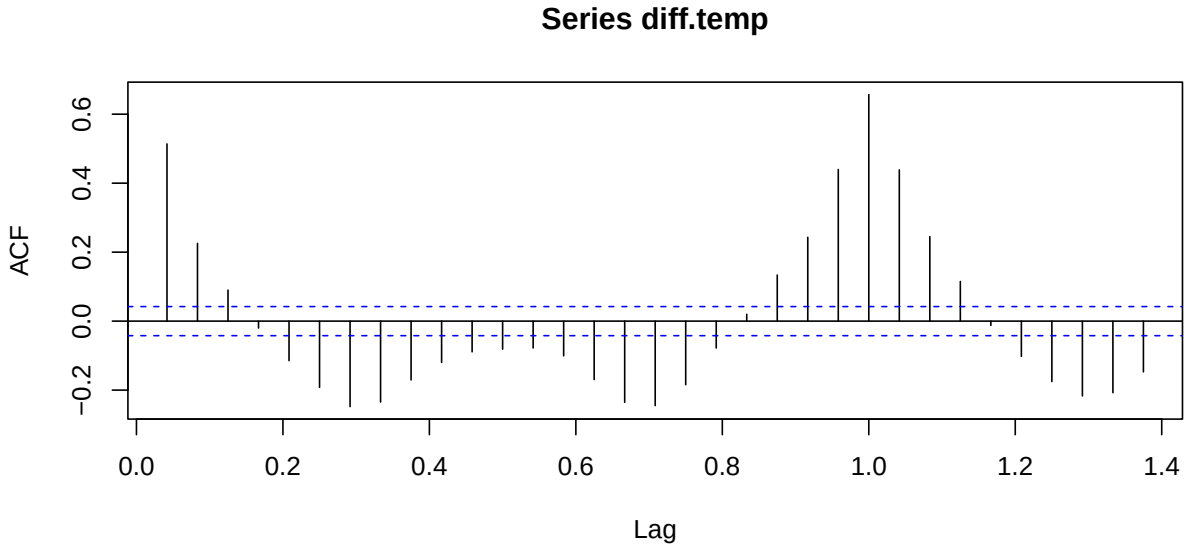
```
diff.temp = diff(temp)
diff2.temp = diff(diff(temp))
diff3.temp = diff(diff(diff(temp)))
ts.plot(diff.temp,ylab="Everest Temp")
```



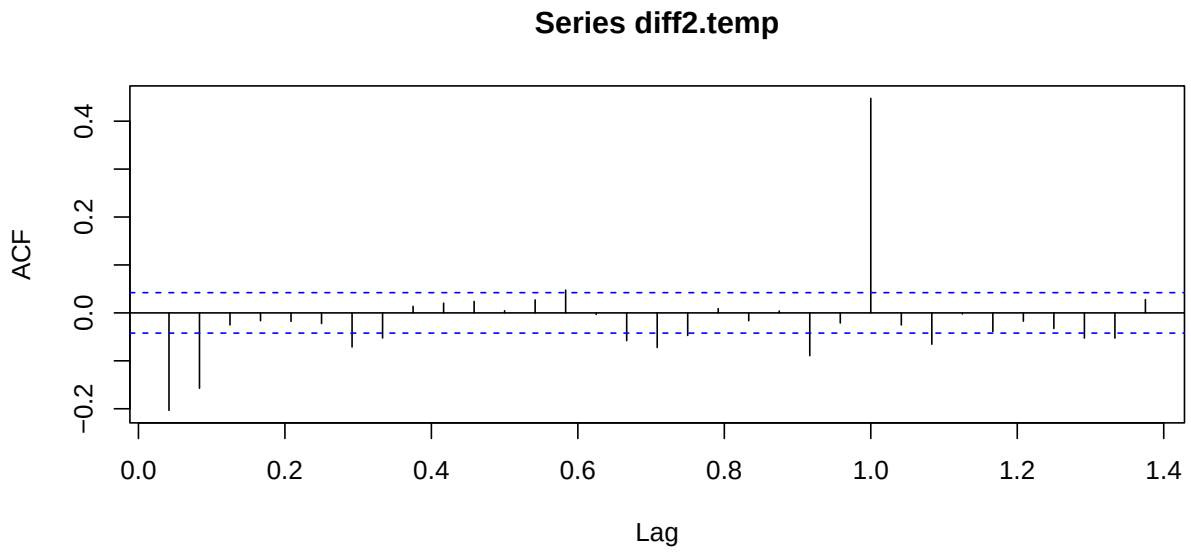

```
# Identifying Max Value
time(diff.temp)[which.max(diff.temp)]
```

```
## [1] 88.125
```

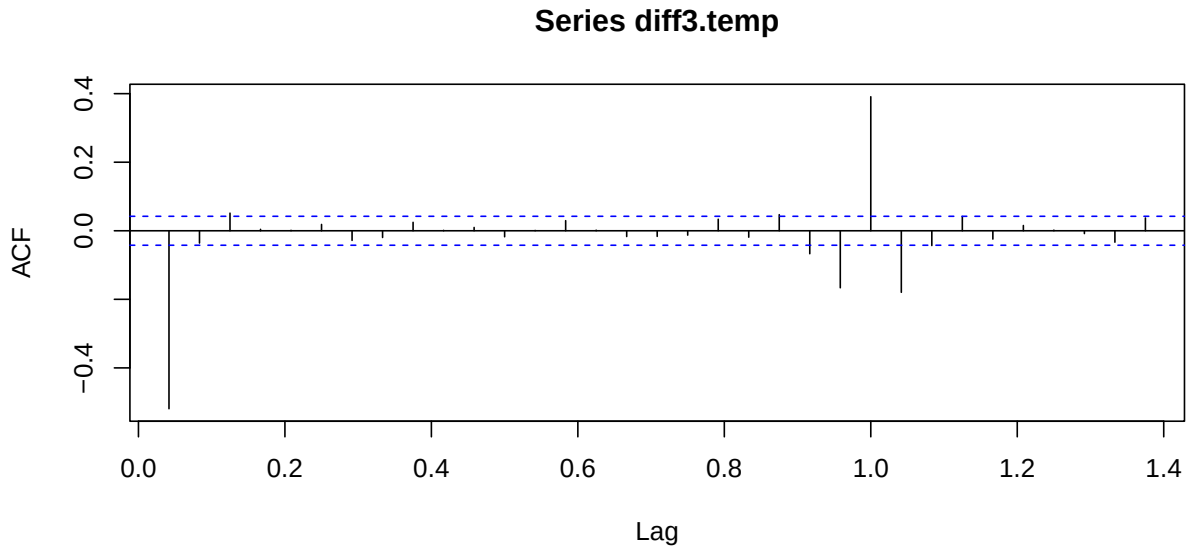
```
# Ploting Autocorrelated Function for differenced time series
acf(diff.temp)
```



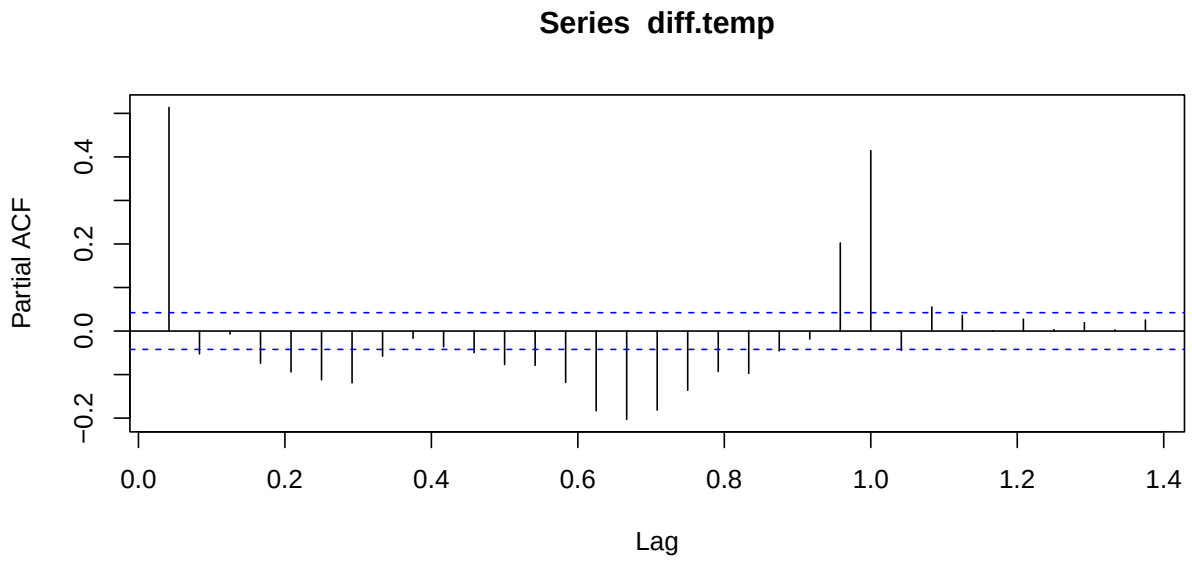

```
acf(diff2.temp)
```



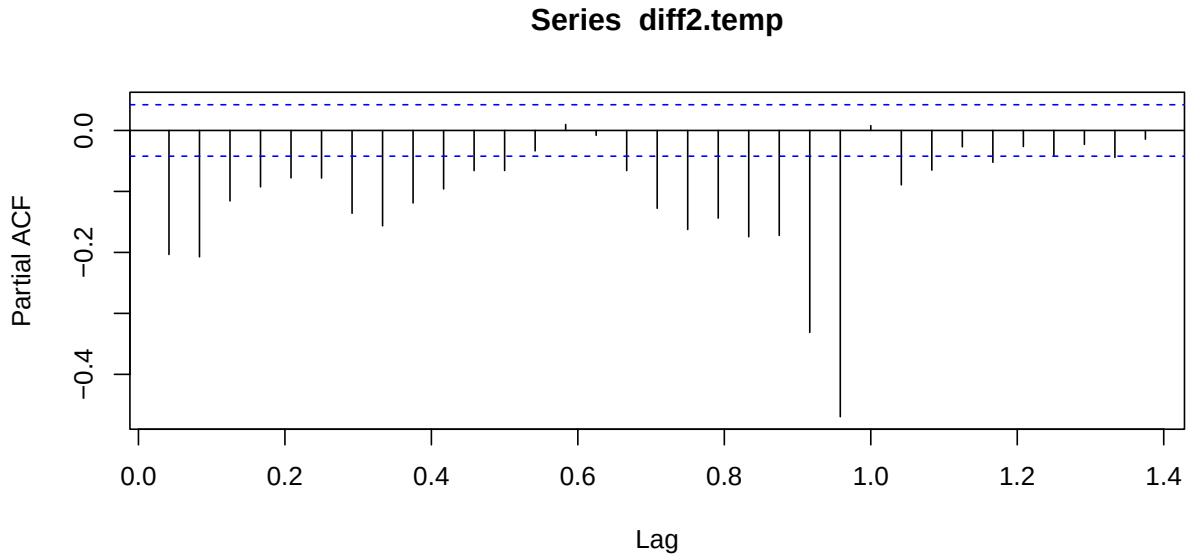

```
acf(diff3.temp)
```



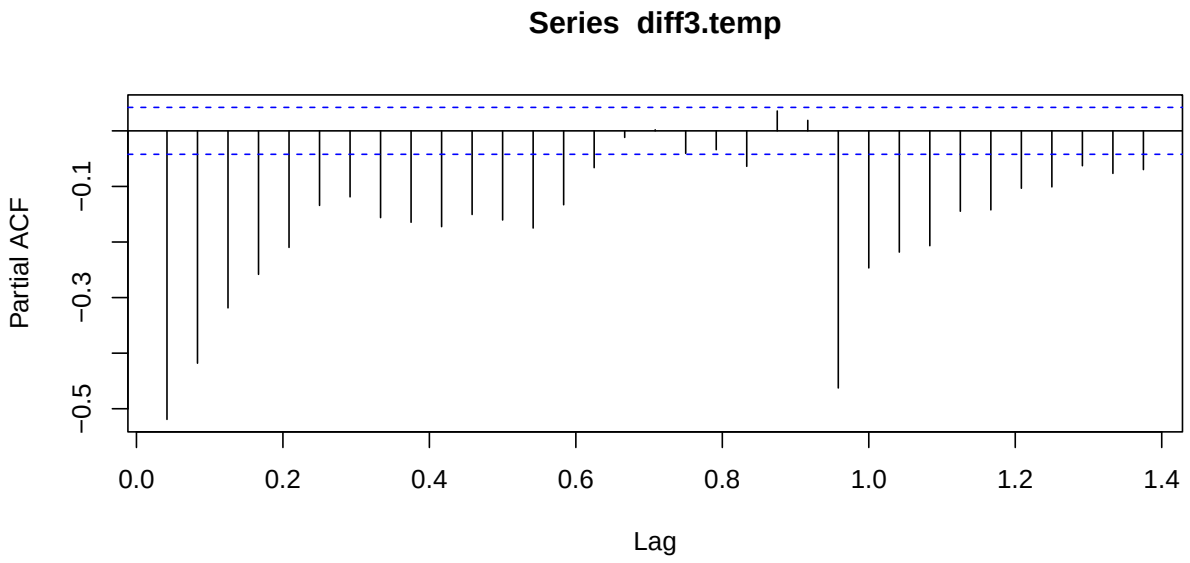

```
# Plotting Partial Autocorrelated Function for differenced data
pacf(diff.temp)
```




```
pacf(diff2.temp)
```




```
pacf(diff3.temp)
```



*Response: Comments about the time series and ACF plots of the difference time series*The plot of the differenced time series shows that there appears to be more constant mean and variance than original data loaded. The ACF and PACF plots for the last differenced data shows that after the first lag attempt there is no much autocorrelation identified as few lags being outside the confidence bands.

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 with residual analysis. Does one model perform better than the other? Which model would you select to fit the seasonality in the data?

```
# ANOVA Model
model1 = lm(temp~season(temp))
summary(model1)
```

```
##
## Call:
## lm(formula = temp ~ season(temp))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.3218  -2.1758  -0.0539   2.0218  11.2616
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -8.637733   0.419589 -20.586  < 2e-16 ***
## season(temp)Season-2    -0.089922   0.593388  -0.152 0.879564
## season(temp)Season-3     0.007367   0.593388   0.012 0.990096
## season(temp)Season-4     1.270233   0.593388   2.141 0.032416 *
## season(temp)Season-5     3.556589   0.593388   5.994 2.40e-09 ***
## season(temp)Season-6     4.840344   0.593388   8.157 5.79e-16 ***
## season(temp)Season-7     5.439567   0.593388   9.167  < 2e-16 ***
```

```
## season(temp)Season-8   5.754467   0.593388   9.698  < 2e-16 ***
## season(temp)Season-9   5.580767   0.593388   9.405  < 2e-16 ***
## season(temp)Season-10  5.180078   0.593388   8.730  < 2e-16 ***
## season(temp)Season-11  4.415444   0.593388   7.441 1.44e-13 ***
## season(temp)Season-12  3.262233   0.593388   5.498 4.31e-08 ***
## season(temp)Season-13  2.256178   0.593388   3.802 0.000147 ***
## season(temp)Season-14  1.569878   0.593388   2.646 0.008214 **
## season(temp)Season-15  1.310044   0.593388   2.208 0.027369 *
## season(temp)Season-16  1.070478   0.593388   1.804 0.071371 .
## season(temp)Season-17  0.818689   0.593388   1.380 0.167828
## season(temp)Season-18  0.701811   0.593388   1.183 0.237053
## season(temp)Season-19  0.522100   0.593388   0.880 0.379033
## season(temp)Season-20  0.408722   0.593388   0.689 0.491028
## season(temp)Season-21  0.266567   0.593388   0.449 0.653313
## season(temp)Season-22  0.158144   0.593388   0.267 0.789872
## season(temp)Season-23  0.031811   0.593388   0.054 0.957251
## season(temp)Season-24  0.083378   0.593388   0.141 0.888269
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.981 on 2136 degrees of freedom
## Multiple R-squared:  0.215,  Adjusted R-squared:  0.2065
## F-statistic: 25.43 on 23 and 2136 DF,  p-value: < 2.2e-16
```

```
# Mean Effects for all seasons
model2 = lm(temp~season(temp)-1)
summary(model2)
```

```
##
## Call:
## lm(formula = temp ~ season(temp) - 1)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.3218  -2.1758  -0.0539   2.0218  11.2616
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## season(temp)Season-1   -8.6377     0.4196 -20.586  < 2e-16 ***
## season(temp)Season-2   -8.7277     0.4196 -20.800  < 2e-16 ***
## season(temp)Season-3   -8.6304     0.4196 -20.569  < 2e-16 ***
## season(temp)Season-4   -7.3675     0.4196 -17.559  < 2e-16 ***
## season(temp)Season-5   -5.0811     0.4196 -12.110  < 2e-16 ***
## season(temp)Season-6   -3.7974     0.4196  -9.050  < 2e-16 ***
## season(temp)Season-7   -3.1982     0.4196  -7.622 3.73e-14 ***
## season(temp)Season-8   -2.8833     0.4196  -6.872 8.30e-12 ***
## season(temp)Season-9   -3.0570     0.4196  -7.286 4.48e-13 ***
## season(temp)Season-10  -3.4577     0.4196  -8.241 2.95e-16 ***
## season(temp)Season-11  -4.2223     0.4196 -10.063  < 2e-16 ***
## season(temp)Season-12  -5.3755     0.4196 -12.811  < 2e-16 ***
## season(temp)Season-13  -6.3816     0.4196 -15.209  < 2e-16 ***
## season(temp)Season-14  -7.0679     0.4196 -16.845  < 2e-16 ***
## season(temp)Season-15  -7.3277     0.4196 -17.464  < 2e-16 ***
## season(temp)Season-16  -7.5673     0.4196 -18.035  < 2e-16 ***
```

```
## season(temp)Season-17  -7.8190     0.4196 -18.635  < 2e-16 ***
## season(temp)Season-18  -7.9359     0.4196 -18.914  < 2e-16 ***
## season(temp)Season-19  -8.1156     0.4196 -19.342  < 2e-16 ***
## season(temp)Season-20  -8.2290     0.4196 -19.612  < 2e-16 ***
## season(temp)Season-21  -8.3712     0.4196 -19.951  < 2e-16 ***
## season(temp)Season-22  -8.4796     0.4196 -20.209  < 2e-16 ***
## season(temp)Season-23  -8.6059     0.4196 -20.510  < 2e-16 ***
## season(temp)Season-24  -8.5544     0.4196 -20.387  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.981 on 2136 degrees of freedom
## Multiple R-squared:  0.7544, Adjusted R-squared:  0.7516
## F-statistic: 273.3 on 24 and 2136 DF,  p-value: < 2.2e-16
```

```
## Applying cos-sin model
model3=lm(temp~harmonic(temp))
summary(model3)
```

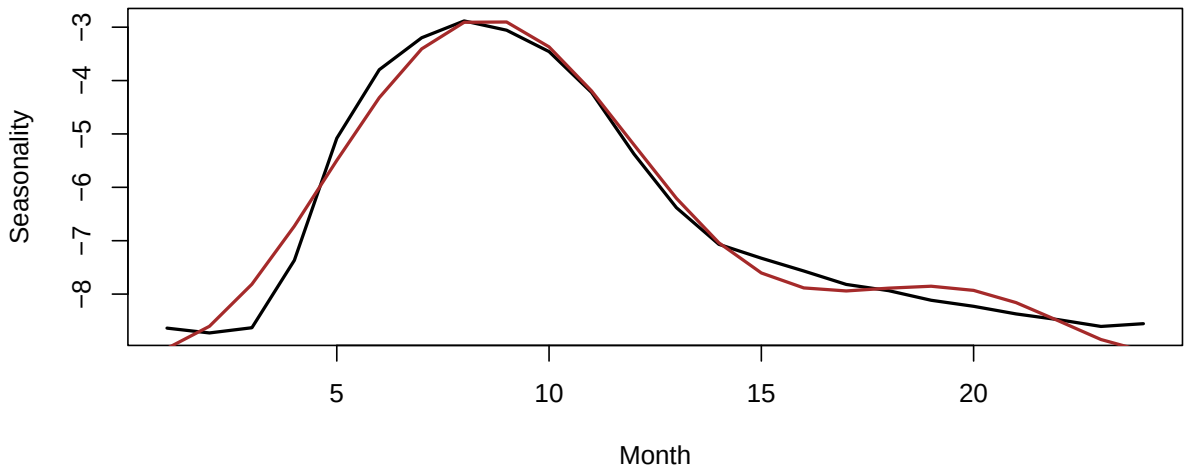
```
##
## Call:
## lm(formula = temp ~ harmonic(temp))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.2278  -2.3106  -0.0262   2.2835  11.9682
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               -6.62044    0.08746  -75.69   <2e-16 ***
## harmonic(temp)cos(2*pi*t) -1.40540    0.12369  -11.36   <2e-16 ***
## harmonic(temp)sin(2*pi*t)  2.22315    0.12369   17.97   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.065 on 2157 degrees of freedom
## Multiple R-squared:  0.1733, Adjusted R-squared:  0.1725
## F-statistic: 226.1 on 2 and 2157 DF,  p-value: < 2.2e-16
```

```
model4=lm(temp~harmonic(temp,2))
summary(model4)
```

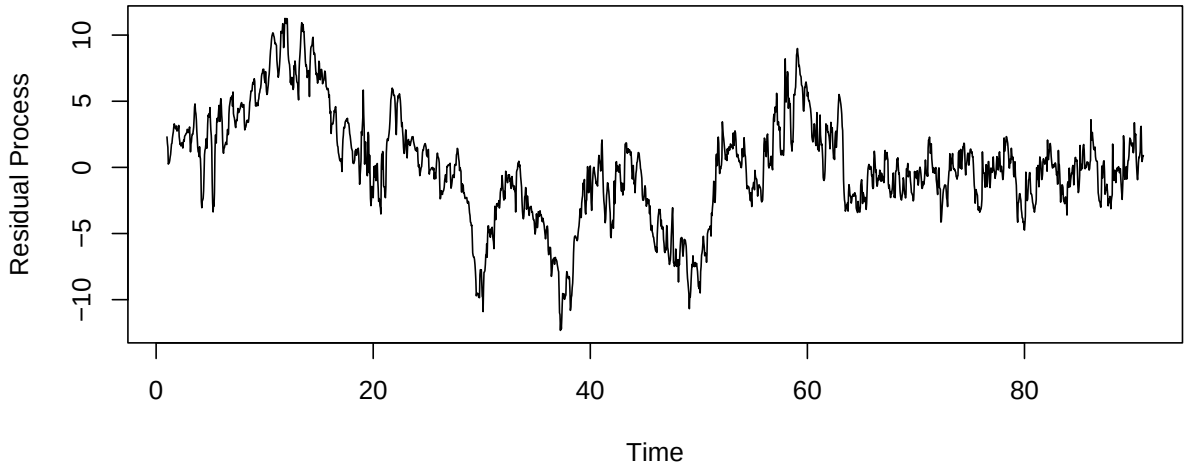
```
##
## Call:
## lm(formula = temp ~ harmonic(temp, 2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.3568  -2.1744  -0.0314   2.0073  11.4989
##
## Coefficients:
##                             Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 -6.62044    0.08556 -77.380  < 2e-16 ***
```

```
## harmonic(temp, 2)cos(2*pi*t) -1.40540    0.12100 -11.615  < 2e-16 ***
## harmonic(temp, 2)cos(4*pi*t) -0.99104    0.12100  -8.191 4.40e-16 ***
## harmonic(temp, 2)sin(2*pi*t)  2.22315    0.12100  18.374  < 2e-16 ***
## harmonic(temp, 2)sin(4*pi*t) -0.68552    0.12100  -5.666 1.66e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.976 on 2155 degrees of freedom
## Multiple R-squared:  0.2097, Adjusted R-squared:  0.2082
## F-statistic: 142.9 on 4 and 2155 DF,  p-value: < 2.2e-16
```

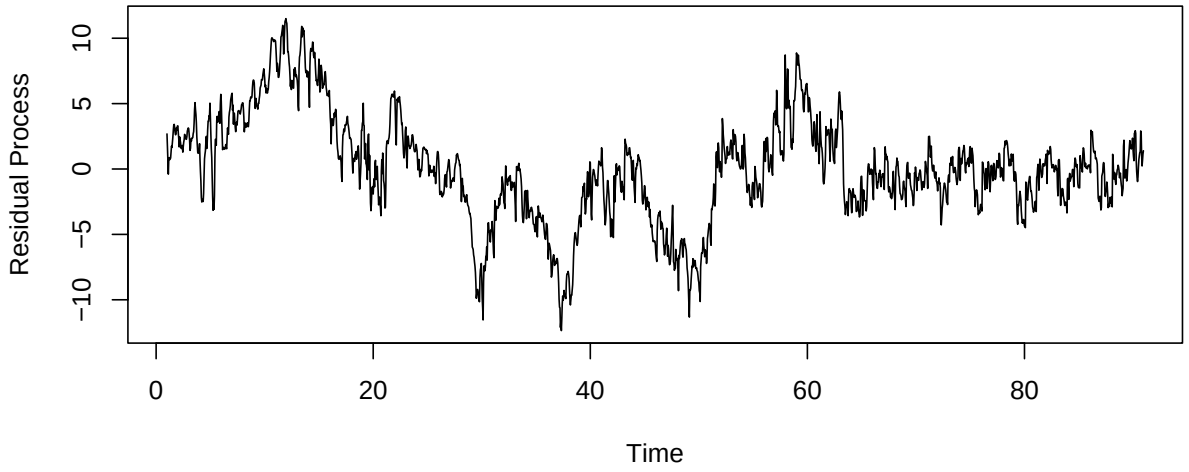
```
# Comparing Seasonality Estimates with Seasonal Means Model
st = coef(model2)
## Cos-Sin Model
st_1 = fitted(model4)
plot(st,lwd=2,type="l",xlab="Month",ylab="Seasonality")
lines(st_1,lwd=2, col="brown")
```



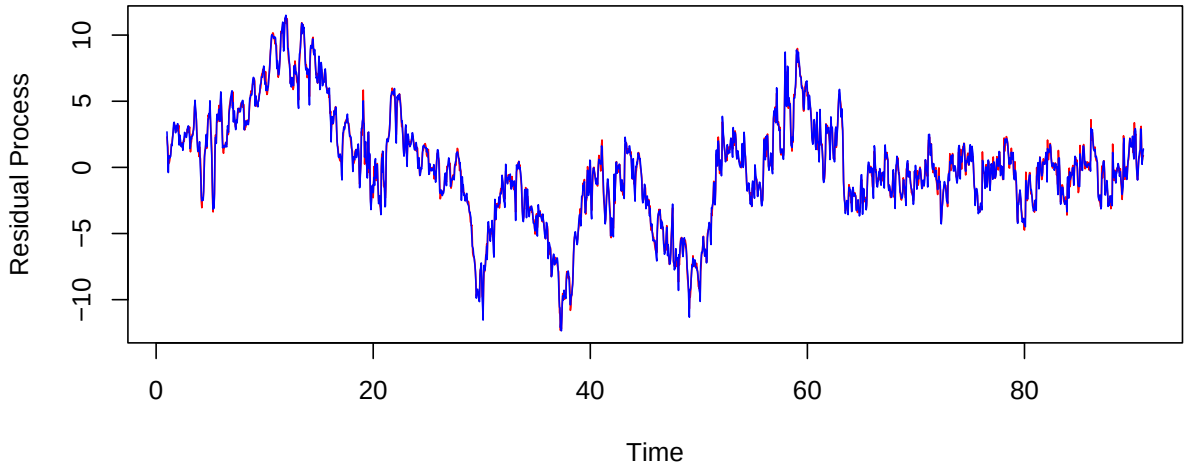

```
## Evaluating the fittness with residual analysis
dif.fit.ANOVA =ts((temp-fitted(model2)),frequency=24)
ts.plot(dif.fit.ANOVA,ylab="Residual Process")
```



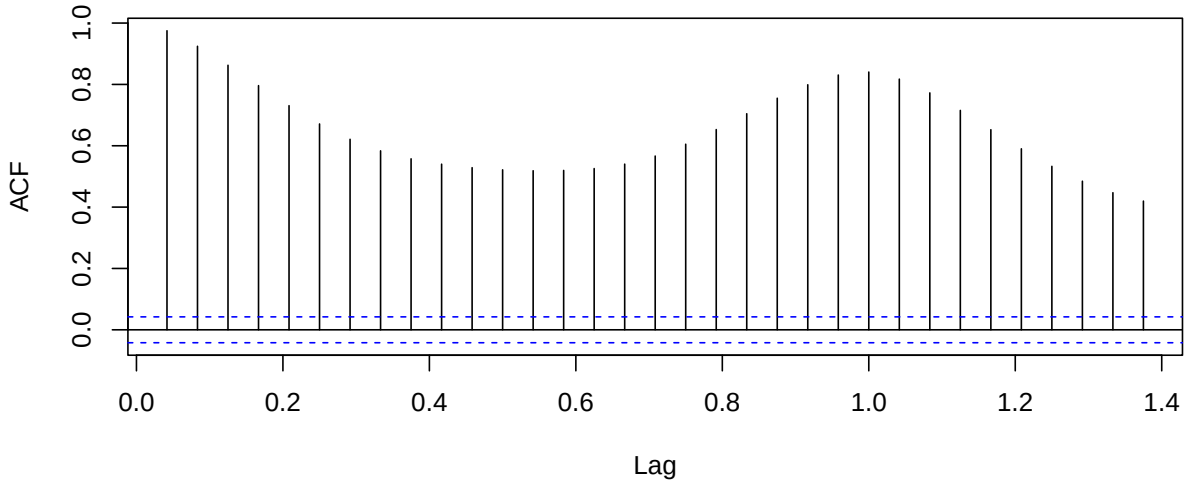

```
dif.fit.cs =ts((temp-fitted(model4)),frequency=24)
ts.plot(dif.fit.cs,ylab="Residual Process")
```



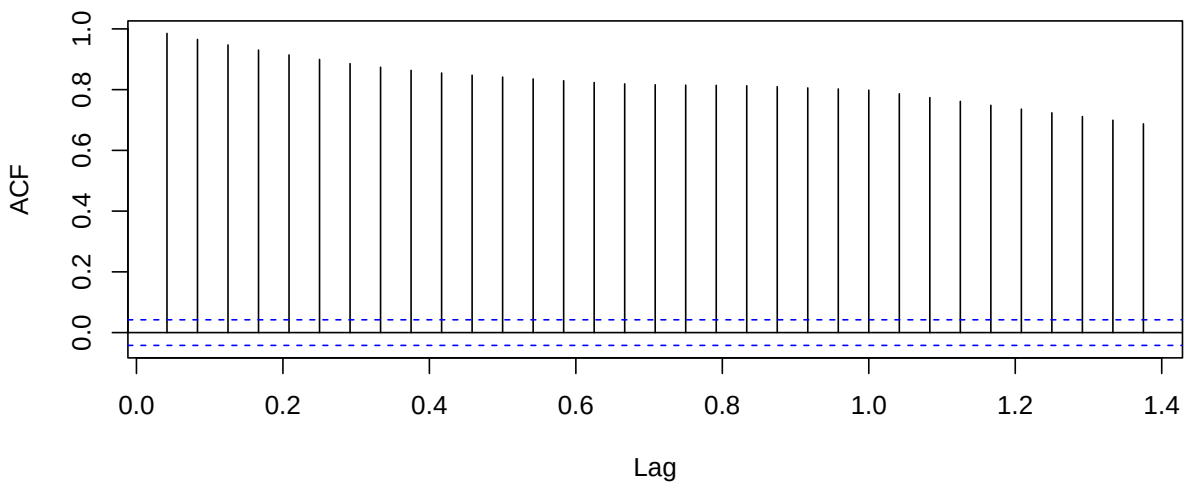

```
## Drawing Comparisons
ts.plot(dif.fit.ANOVA,ylab="Residual Process",col="red")
lines(dif.fit.cs,col="blue")
```




```
acf(temp,main="")
```




```
acf(dif.fit.ANOVA,main="")
```



Response: Compare Seasonality Models: In fitting plots,both models show similar seasonality. It is evident that ANOVA model performs better than Cos-Sin Model.Seasonality in the data by using ANOVA will likely be a better approach

Question 2c: Trend-Seasonality Estimation

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

- Parametric Polynomial Regression
- Non-parametric model

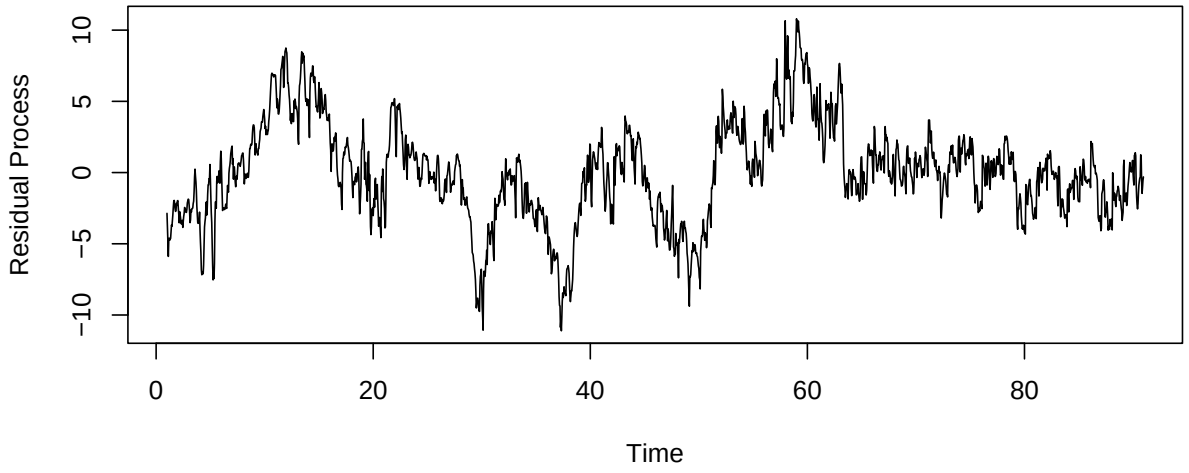
Overlay the fitted values on the original time series. Plot the residuals with respect to time. Plot the ACF of the residuals. Comment on how the two models fit and on the appropriateness of the stationarity assumption of the residuals.

What form of modelling 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.

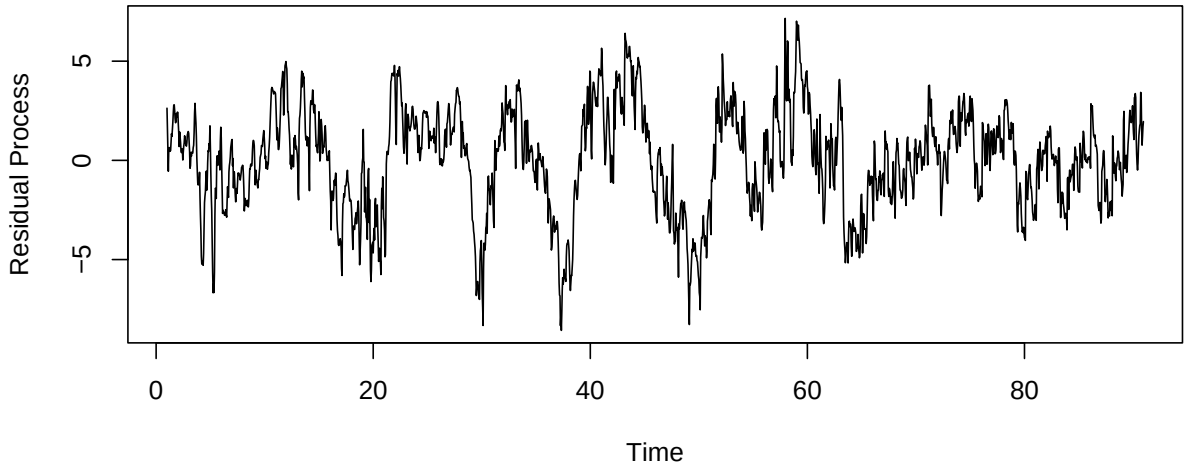
```
## X-axis points converted to 0-1 scale, common in nonparametric regression
time.tnd = c(1:length(temp))
time.tnd = c(time.tnd - min(time.tnd))/max(time.tnd)
```

```
# Fitting parametric model for both trend and seasonality bu using Linear Regression

# Linear trend

lm.fit.lin = dynlm(temp~trend(temp)+harmon(temp,2))

## Quadratic trend
```

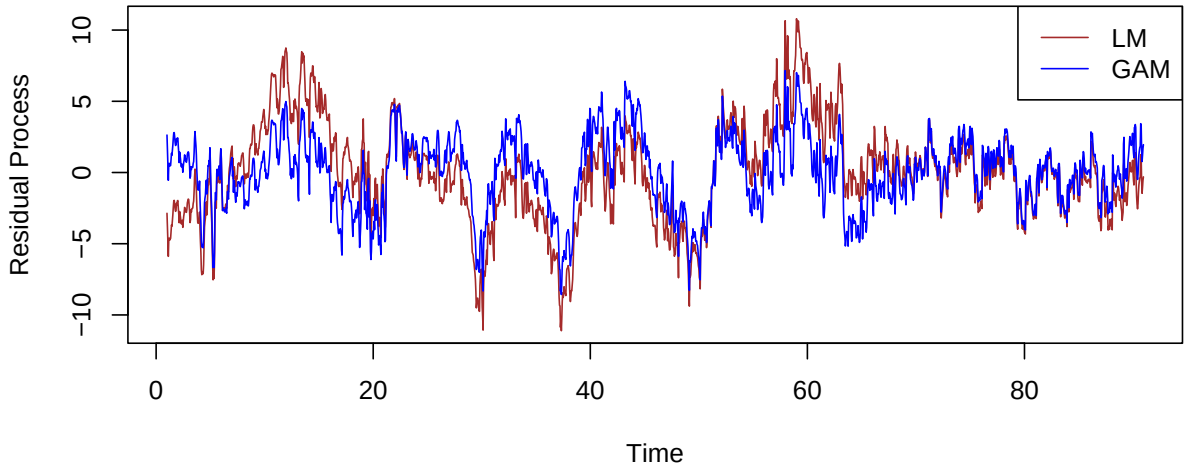
```
x1 = time.tnd
x2 = time.tnd^2
lm.fit = dynlm(temp ~ x1 + x2 + harmon(temp,2))

#summary(lm.fit)

dif.fit.lm = ts((temp-fitted(lm.fit)),frequency=24)
ts.plot(dif.fit.lm,ylab="Residual Process")
```



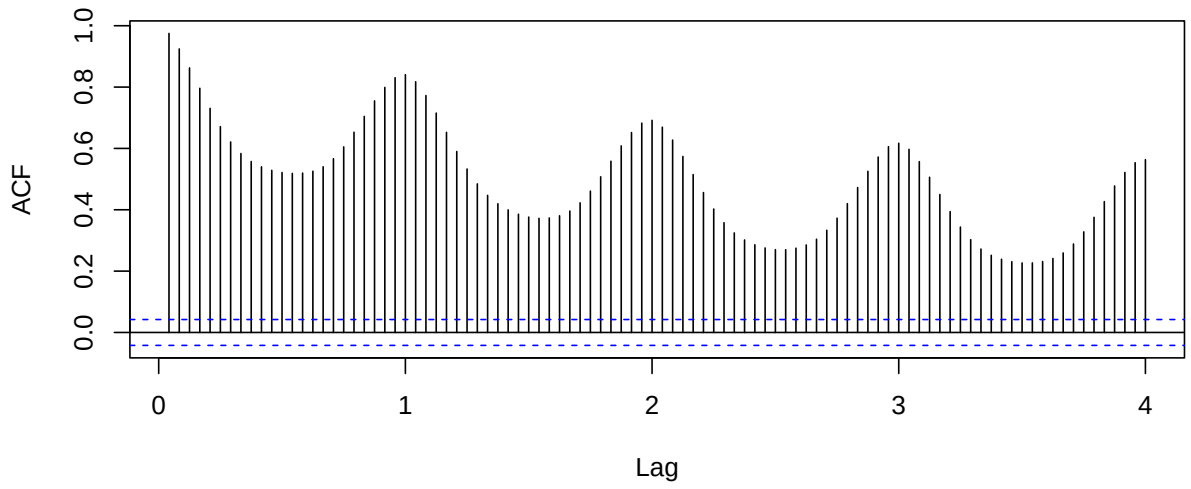

```
# Fitting a non-parametric model for trend & Linear Model for Seasonality

har_2 = harmonic(temp,2)
gam.fit = gam(temp ~ s(time.tnd)+har_2)
dif.fit.gam = ts((temp-fitted(gam.fit)),frequency=24)
ts.plot(dif.fit.gam,ylab="Residual Process")
```



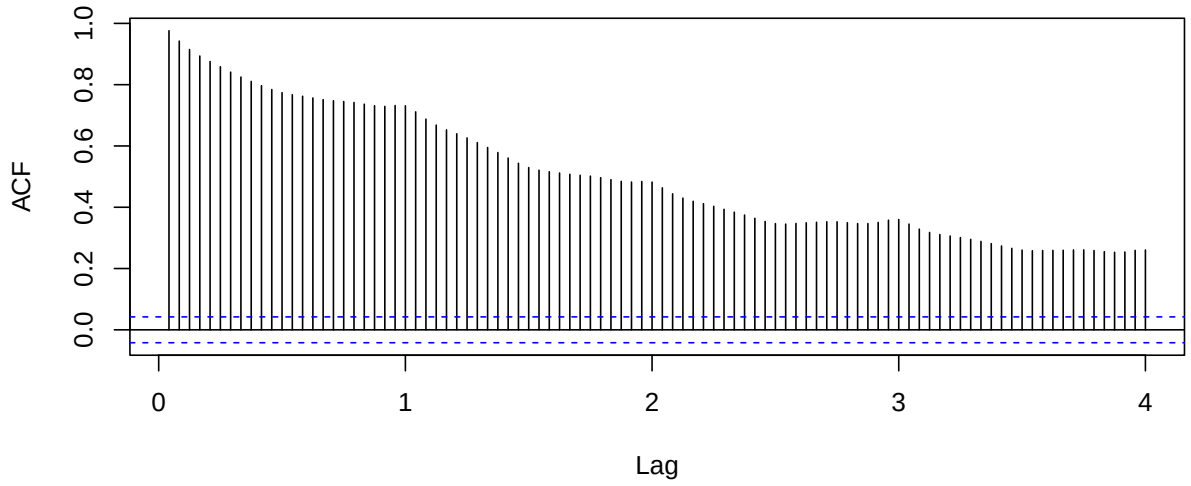

```
## Drawing comparison

ts.plot(dif.fit.lm,ylab="Residual Process",col="brown")
lines(dif.fit.gam,col="blue")
legend(x="topright",y=1.4,legend=c("LM","GAM"),lty = 1,col=c("brown","blue"))
```



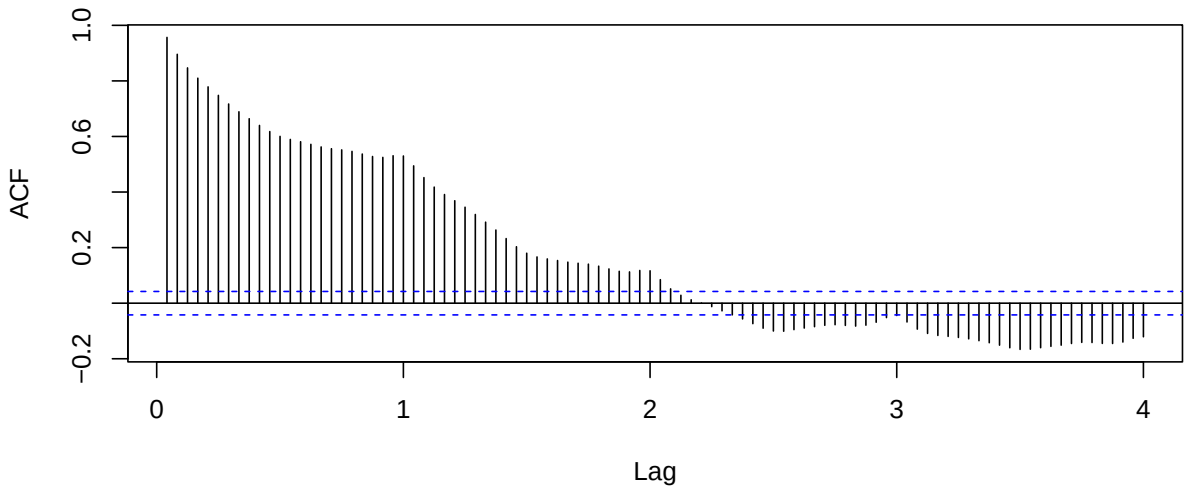

```
acf(temp,lag.max=24*4,main="")
```




```
acf(dif.fit.lm,lag.max=24*4,main="")
```




```
acf(dif.fit.gam,lag.max=24*4,main="")
```



*Response: Model Comparison*From Time Series, it is evident there is prominent trend and because of this the non-constant mean and zero autocorrelation assumptions are both violated for the original data. ACF plot for original data shows the presence of consistent seasonal pattern and these pattern can be seen repetitively for each block of lags.However, the ACF plot for non-parametric model and parametric model shows there are not significant autocorrelation, with only some lags being outside the confidence bands.