ISYE 6402 Homework 2

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()`

```
setwd("C:/Users/angel/Downloads")
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

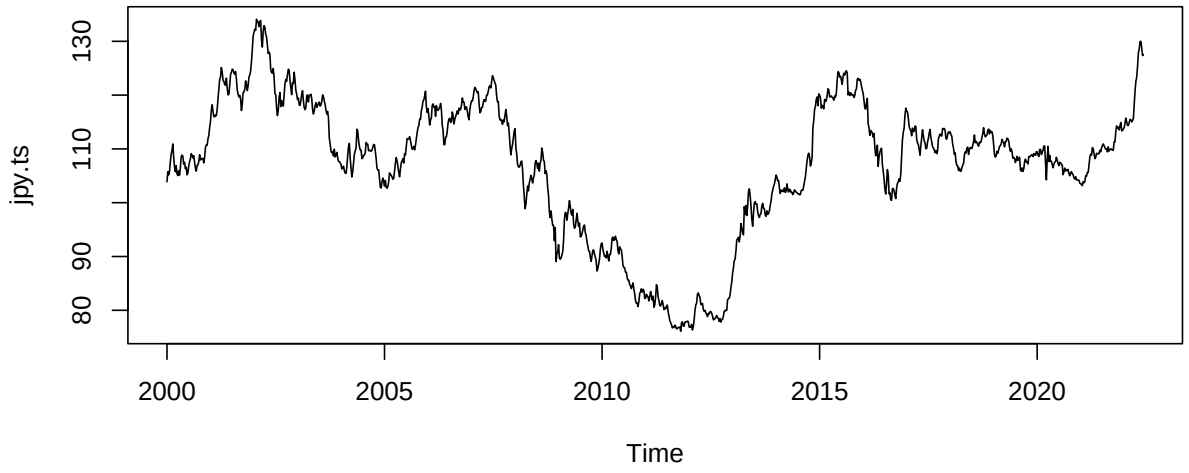
Before exploring the data, can you infer the data features from what you know about the USD-JPY currency exchange? Next plot the Time Series and ACF plots of the weekly data. Comment on the main features, and identify what (if any) assumptions of stationarity are violated.

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

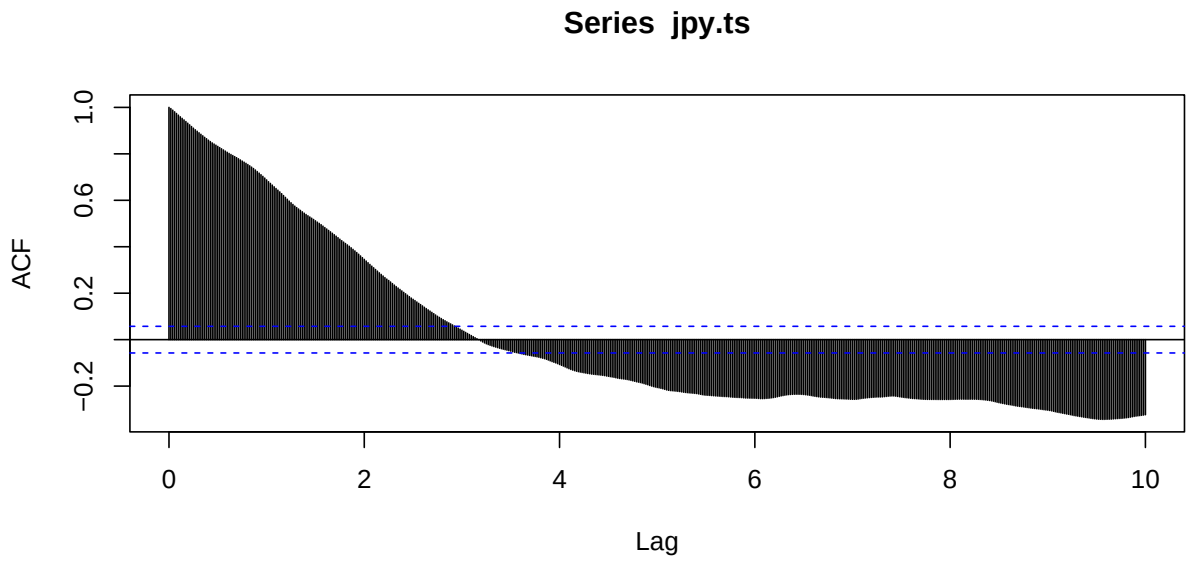
Response: General Insights on the USD-JPY Currency Rate

The USD-JPY currency exchange is the way customers can exchange one currency for another. It determines the purchasing power of the given currency and the currecy become more valuable if the demand gets higher.

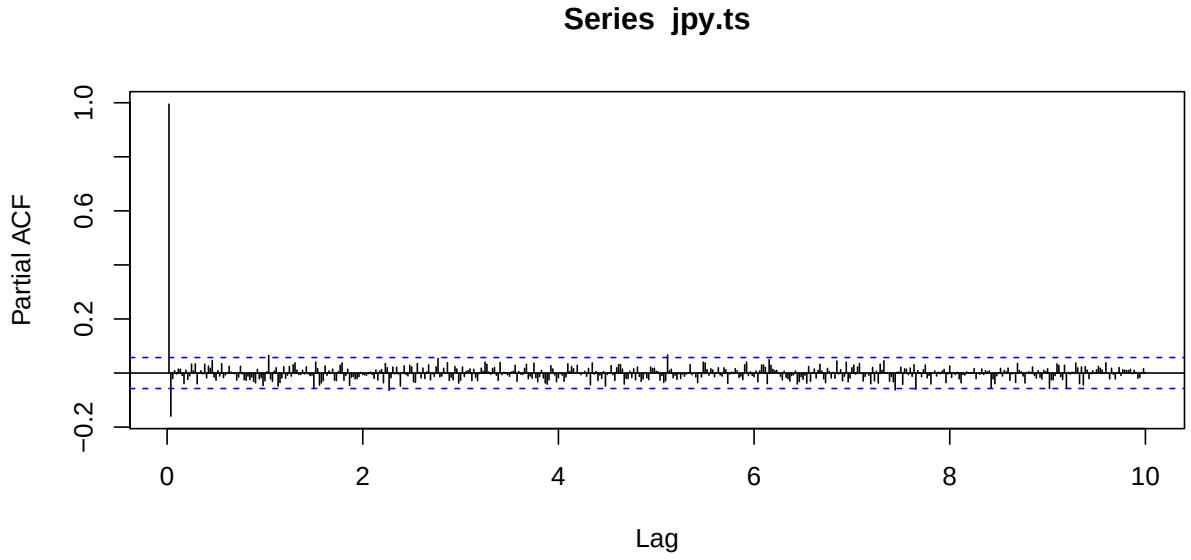
```
ts.plot(jpy.ts)
```




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




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



Response: General Insights from the Graphical Analysis The time series plot suggest that there was a decline in 2008, then a trend went upward an continued up until 2022. The graph seems to show some cyclical nature and a parabolic trend. A seasonality fitting model might work better due to the overall flat trend.

Question 1b: Trend Estimation

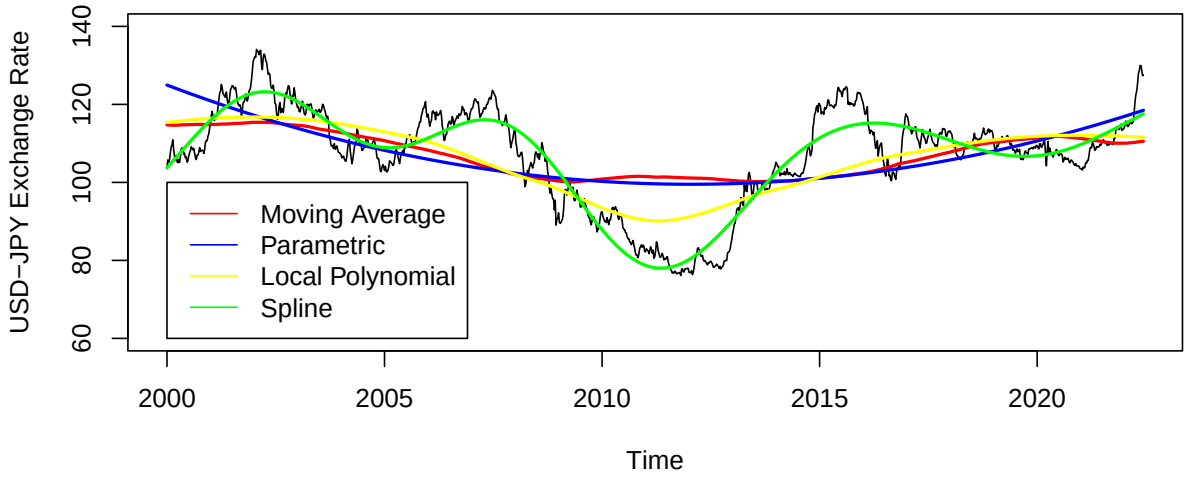
Fit the following trend estimation models:

- Moving Average

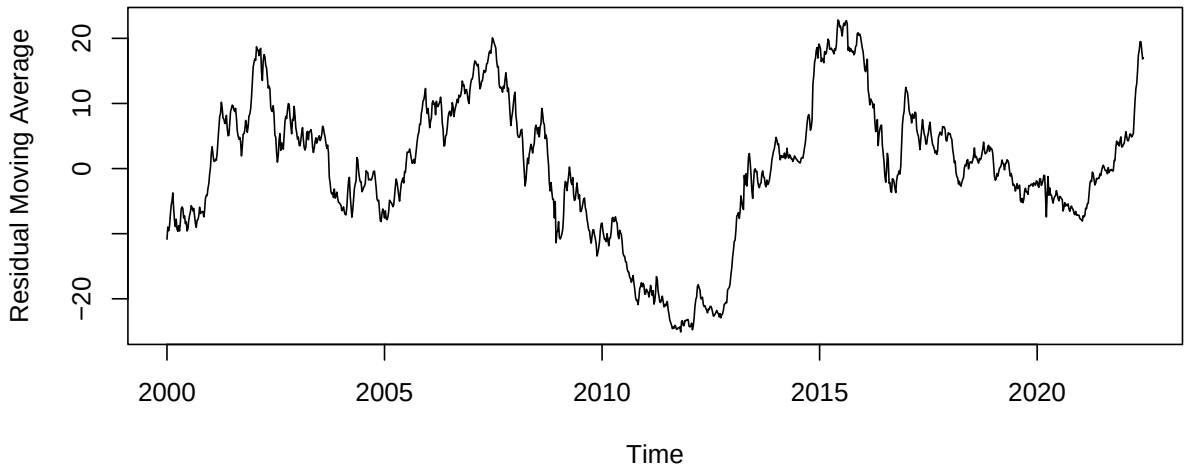
- Parametric Quadratic Polynomial
- Local Polynomial
- Splines Smoothing

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.

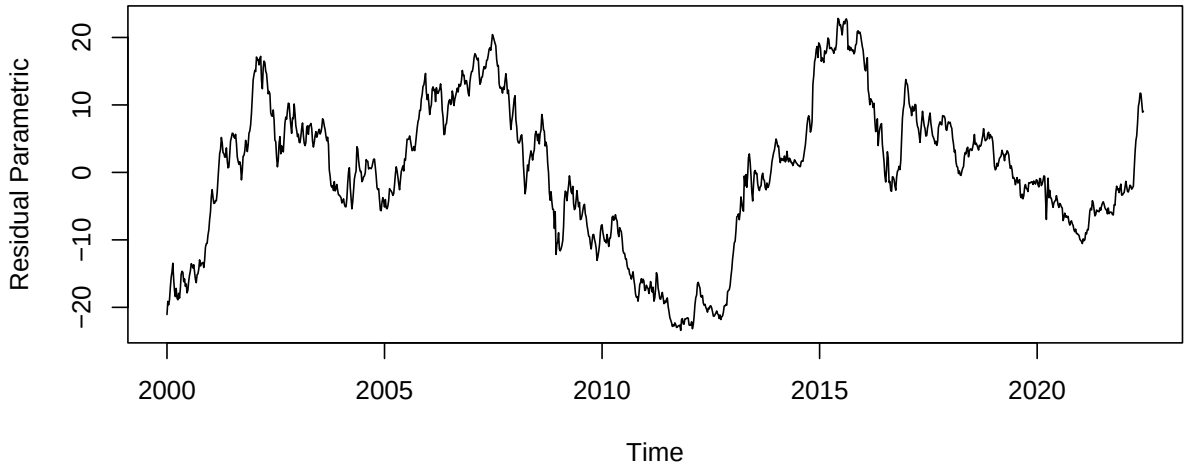
```
##Moving Average
time.pts = c(1:length(jpy.ts))
time.pts = c(time.pts-min(time.pts))/max(time.pts)
ma.fit=ksmooth(time.pts, jpy.ts, kernel="box")
price.fit.ma = ts(ma.fit$y, start=2000, frequency = 52)

##Parametric Quadratic Polynomial
x1=time.pts
x2=time.pts^2
param.fit = lm(jpy.ts~x1+x2)
price.fit.param = ts(fitted(param.fit), start=2000, frequency=52)

##Local Polynomial
local.fit = loess(jpy.ts ~ time.pts)
price.fit.local = ts(fitted(local.fit), start=2000, frequency=52)

#Spline
spline.fit = gam(jpy.ts~s(time.pts))
price.fit.spline = ts(fitted(spline.fit), start=2000, frequency=52)

##Overlay fitted values
ts.plot(jpy.ts,ylab="USD-JPY Exchange Rate", ylim=c(60,140))
lines(price.fit.ma, lwd=2, col="red")
lines(price.fit.param, lwd=2, col="blue")
lines(price.fit.local, lwd=2, col="yellow")
lines(price.fit.spline, lwd=2, col="green")
legend(x=2000, y=100, legend=c("Moving Average", "Parametric", "Local Polynomial", "Spline"), lty=1,
       col=c("red","blue","yellow","green"))
```



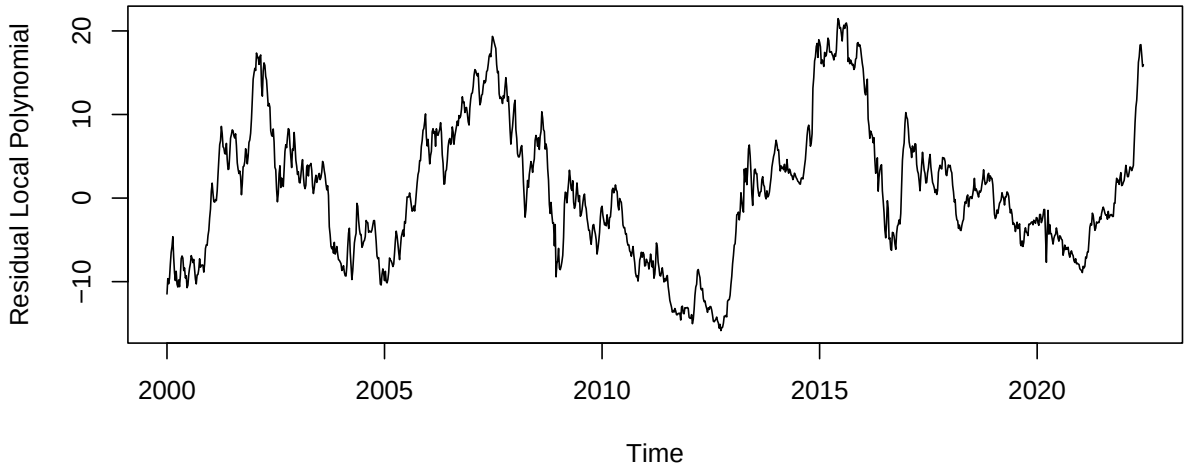

```
#Residuals
diff.fit.ma = ts((jpy.ts-ma.fit$y),start=2000,frequency=52)
ts.plot(diff.fit.ma, ylab="Residual Moving Average")
```



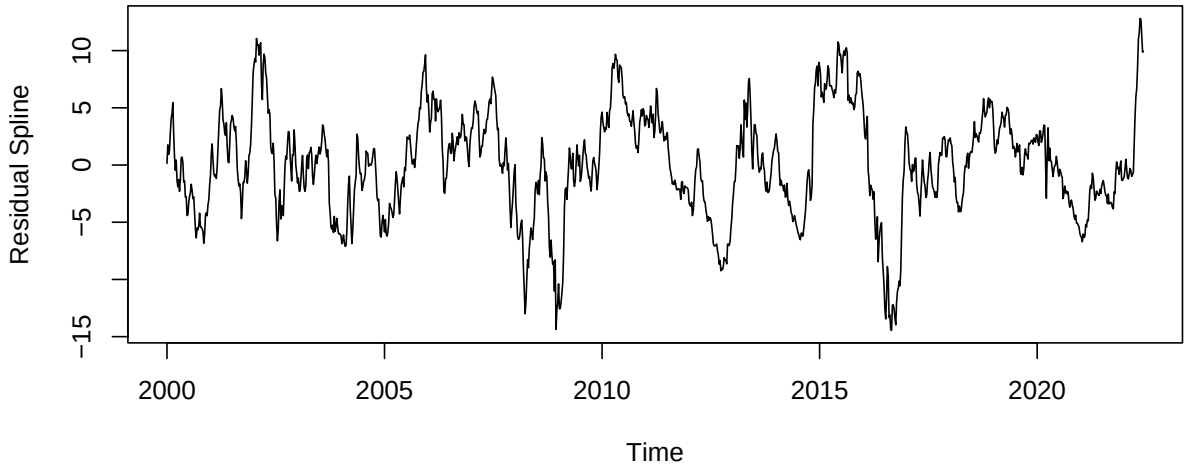

```
diff.fit.ma = ts((jpy.ts-price.fit.param),start=2000,frequency=52)
ts.plot(diff.fit.ma, ylab="Residual Parametric")
```




```
diff.fit.ma = ts((jpy.ts-price.fit.local),start=2000,frequency=52)
ts.plot(diff.fit.ma, ylab="Residual Local Polynomial")
```

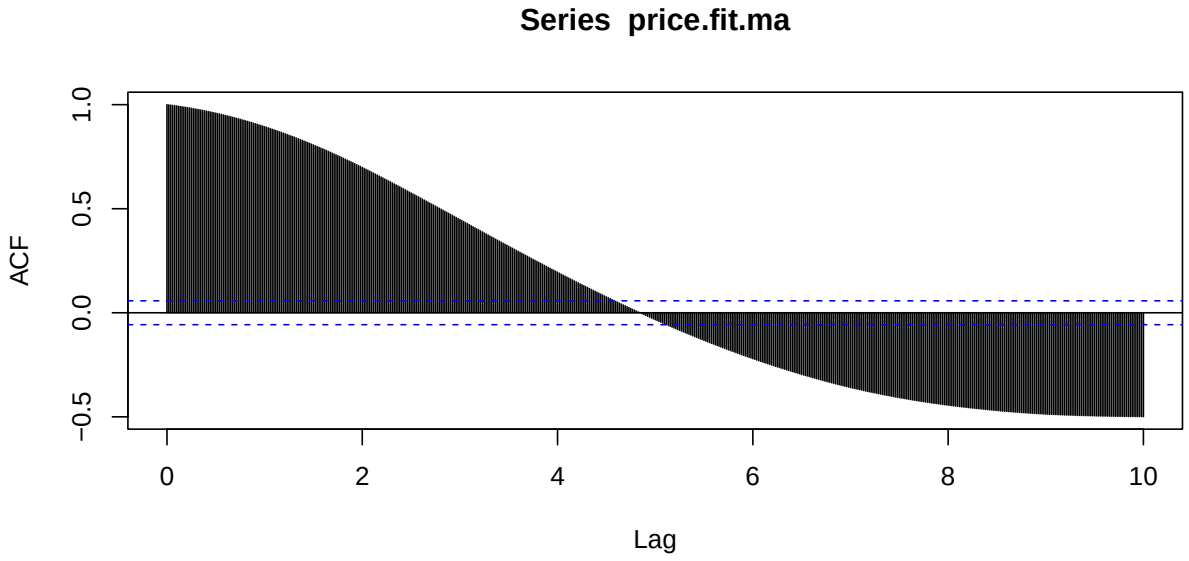



```
diff.fit.ma = ts((jpy.ts-price.fit.spline),start=2000,frequency=52)
ts.plot(diff.fit.ma, ylab="Residual Spline")
```

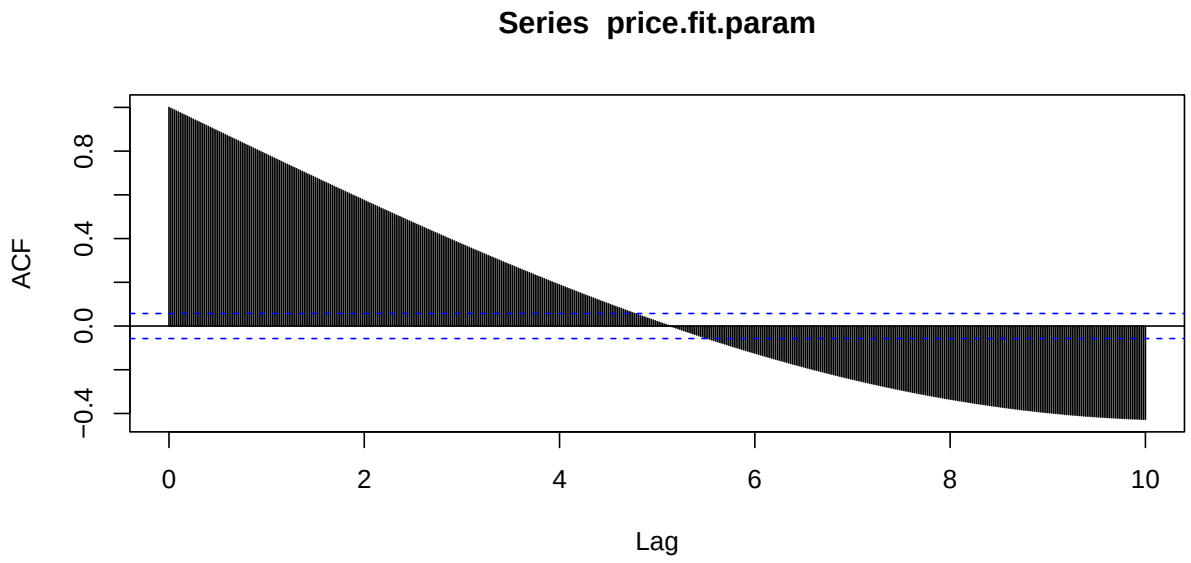


Response: Comparison of the fitted trend models: Based on the models, the spline model performed the best. The residuals are smaller for spline, therefore it performed the beset. The local polynomial performed the second best at explaining the variance.

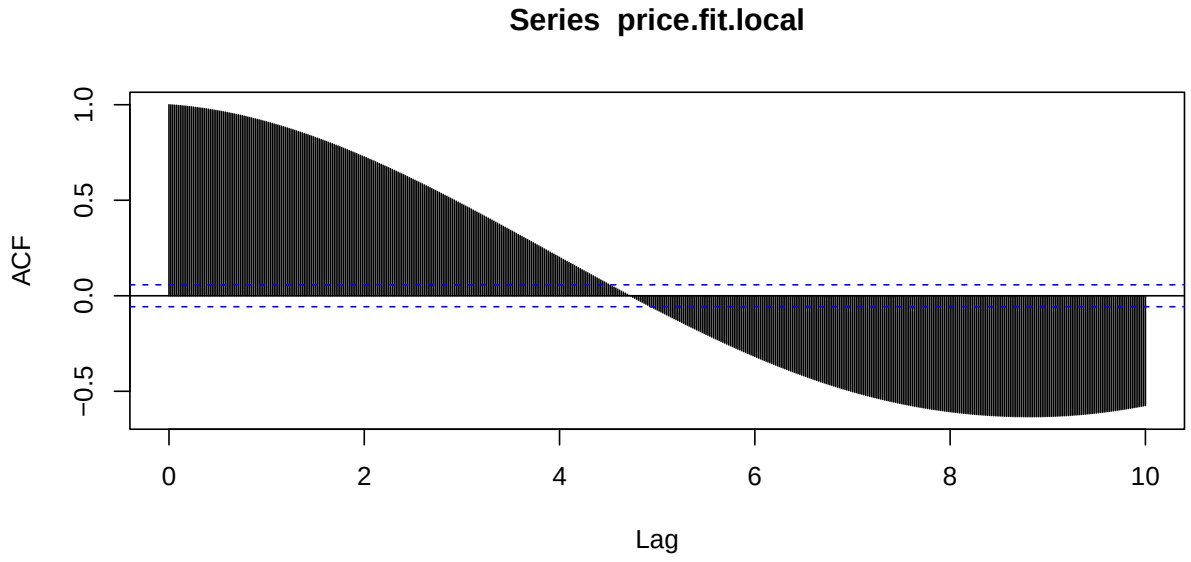
```
#ACF
acf(price.fit.ma, lag.max = 52*10)
```



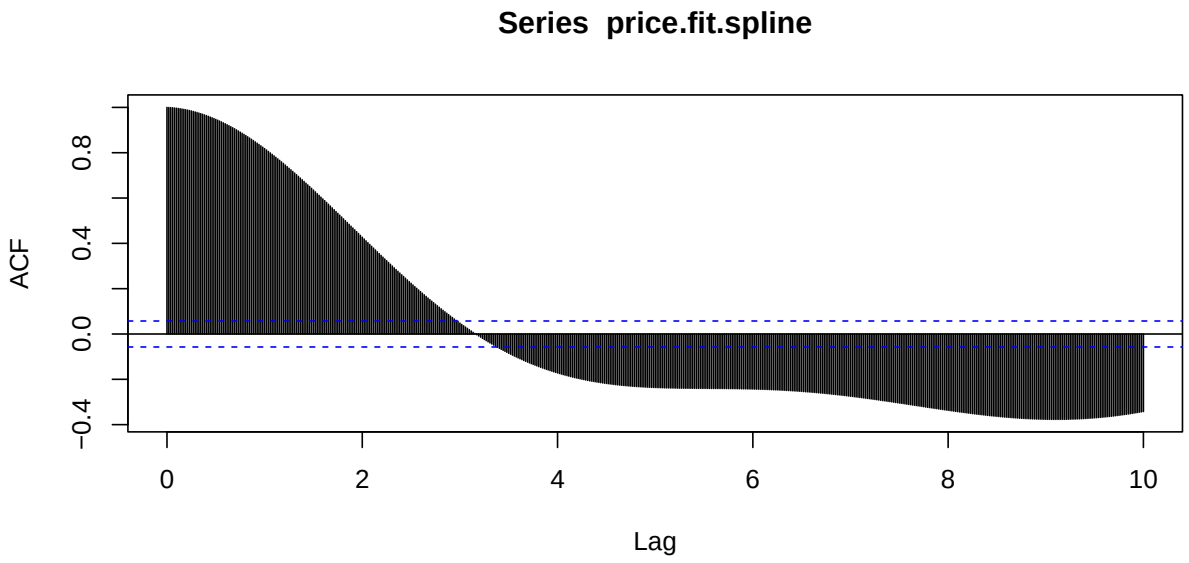

```
acf(price.fit.param, lag.max = 52*10)
```




```
acf(price.fit.local, lag.max = 52*10)
```




```
acf(price.fit.spline, lag.max = 52*10)
```



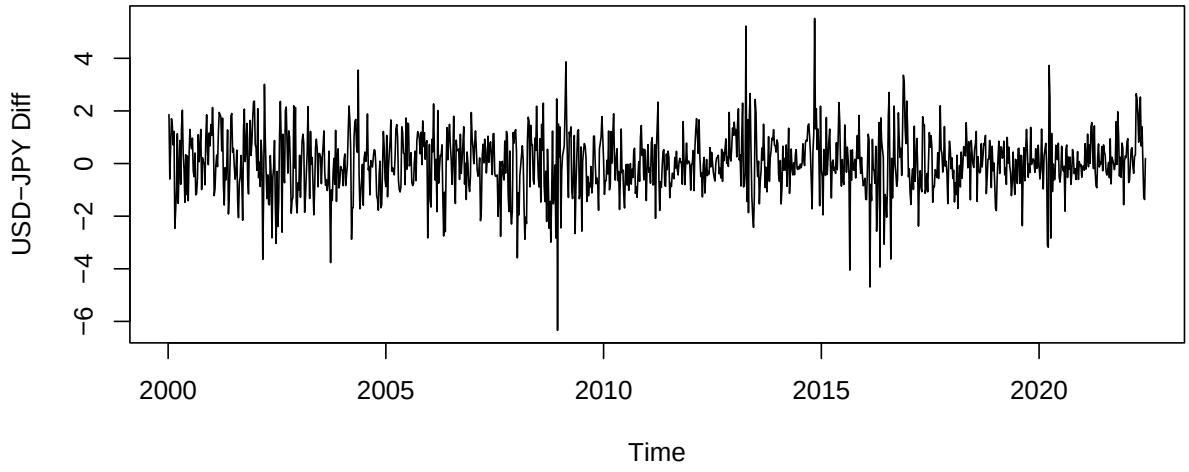
Response: Appropriateness of the trend model for stationarity Based on the plots, there's no show of seasonality or cyclical trend. However, it does suggest stationarity holds because the residual ACF plots decreases with time.

Question 1c: Differenced Data Modeling

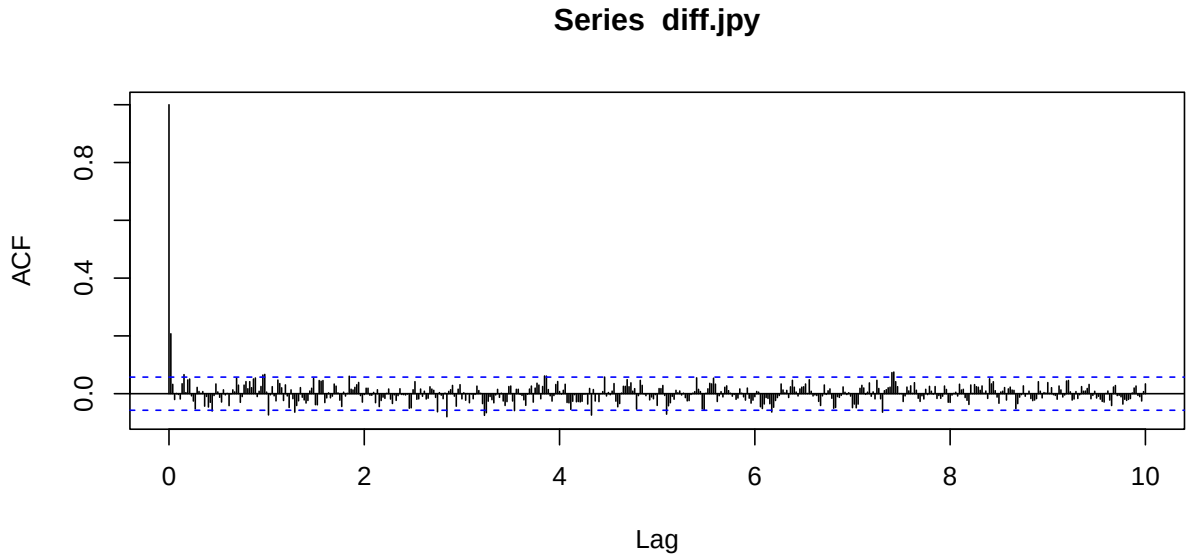
Now plot the difference time series and its ACF plot. Apply the four trend models in Question 1b to the differenced time series. What can you conclude about the difference data in terms of stationarity? Which model would you recommend to apply (trend removal via fitting trend vs differencing) such that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

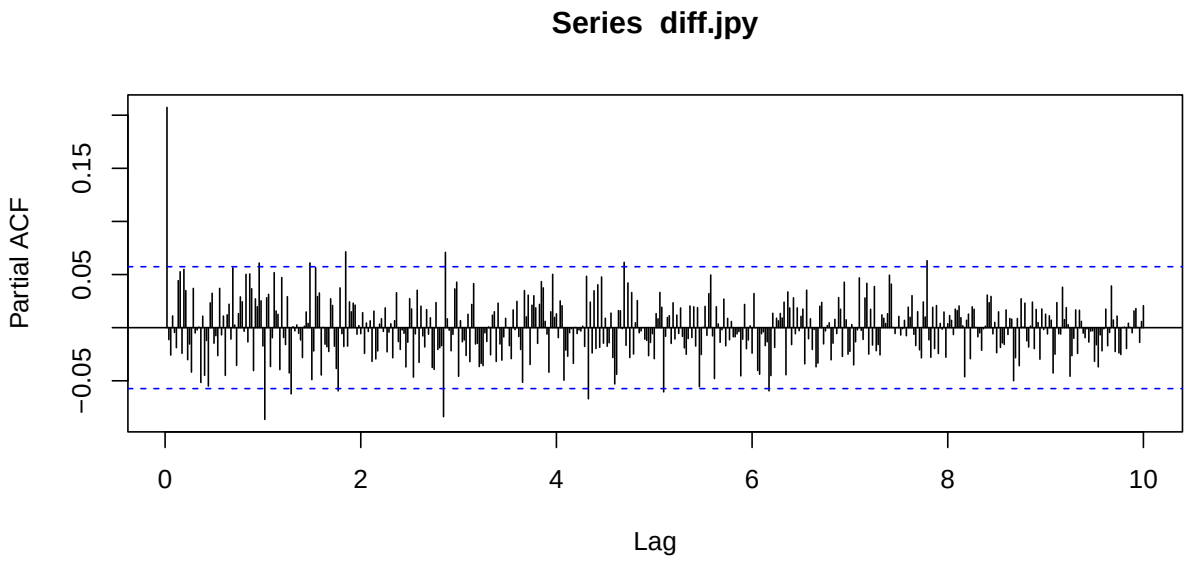
```
diff.jpy <- diff(jpy.ts)
ts.plot(diff.jpy, ylab="USD-JPY Diff")
```



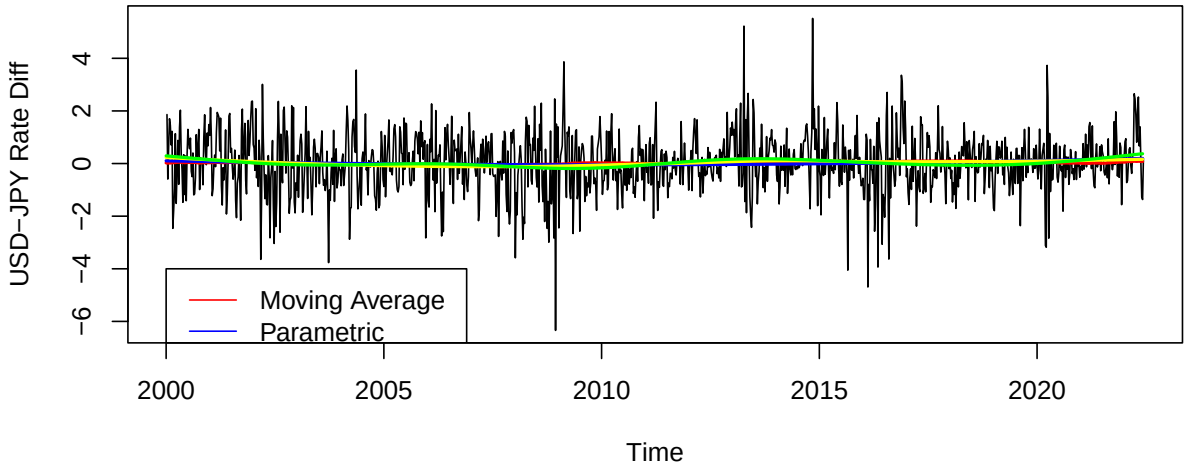

```
acf(diff.jpy, lag.max=52*10)
```



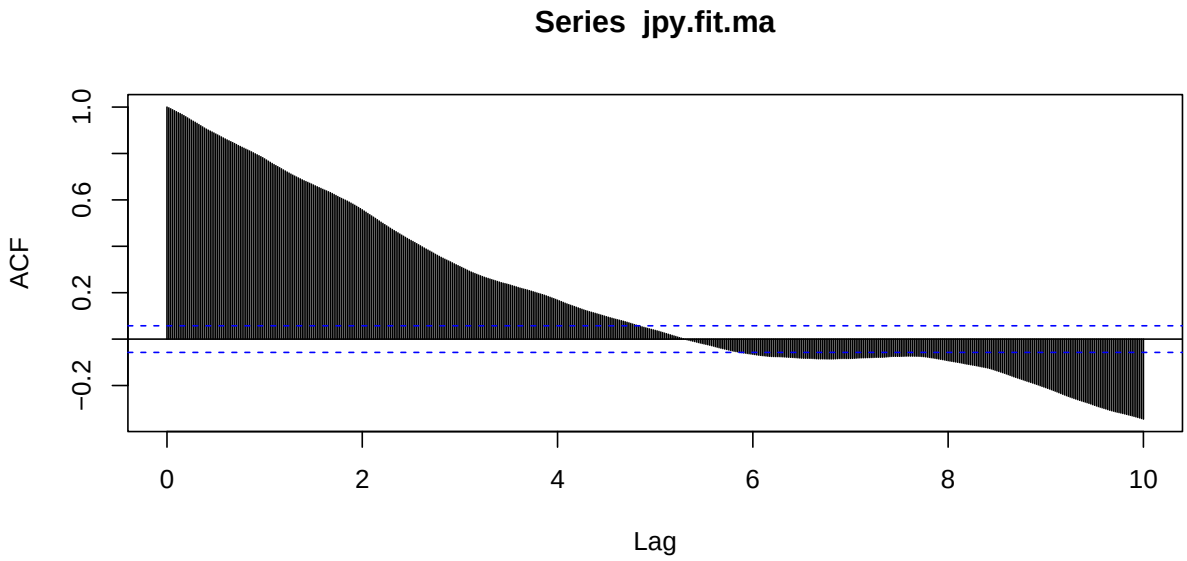

```
pacf(diff.jpy, lag.max=52*10)
```



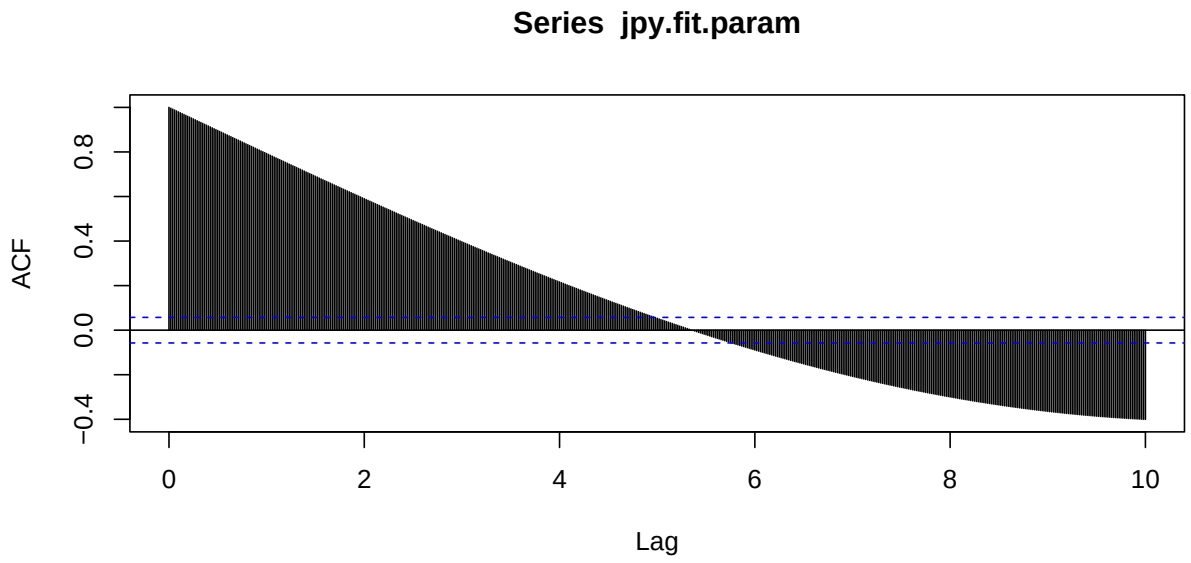

```
time.pts = c(1:length(diff.jpy))
time.pts = c(time.pts-min(time.pts))/max(time.pts)
##Moving Average
time.pts = tail(time.pts,length(time.pts))
ma.fit=ksmooth(time.pts, diff.jpy, kernel="box")
jpy.fit.ma = ts(ma.fit$y, start=2000, frequency = 52)
##Parametric Quadratic Polynomial
x1=time.pts
x2=time.pts^2
param.fit = lm(diff.jpy~x1+x2)
jpy.fit.param = ts(fitted(param.fit), start=2000, frequency=52)
##Local Polynomial
local.fit = loess(diff.jpy ~ time.pts)
jpy.fit.local = ts(fitted(local.fit), start=2000, frequency=52)
#Spline
spline.fit = gam(diff.jpy~s(time.pts))
jpy.fit.spline = ts(fitted(spline.fit), start=2000, frequency=52)
##Overlay fitted values
ts.plot(diff.jpy,ylab="USD-JPY Rate Diff")
lines(jpy.fit.ma, lwd=2, col="red")
lines(jpy.fit.param, lwd=2, col="blue")
lines(jpy.fit.local, lwd=2, col="yellow")
lines(jpy.fit.spline, lwd=2, col="green")
legend(x=2000, y=-4, legend=c("Moving Average", "Parametric", "Local Polynomial", "Spline"), lty=1,
       col=c("red","blue","yellow","green"))
```



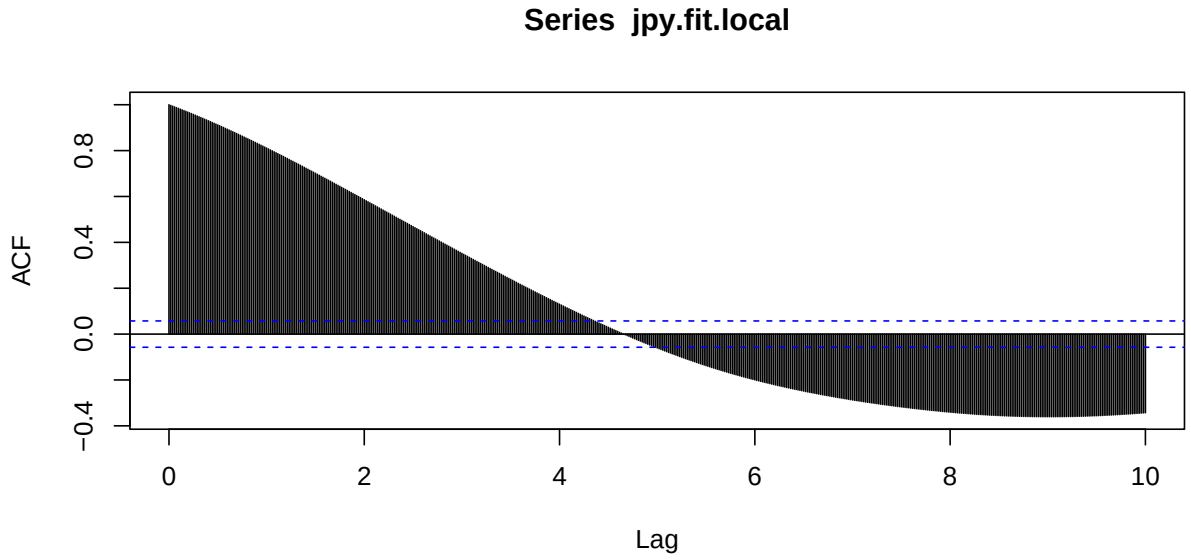

```
#ACF
acf(jpy.fit.ma, lag.max = 52*10)
```



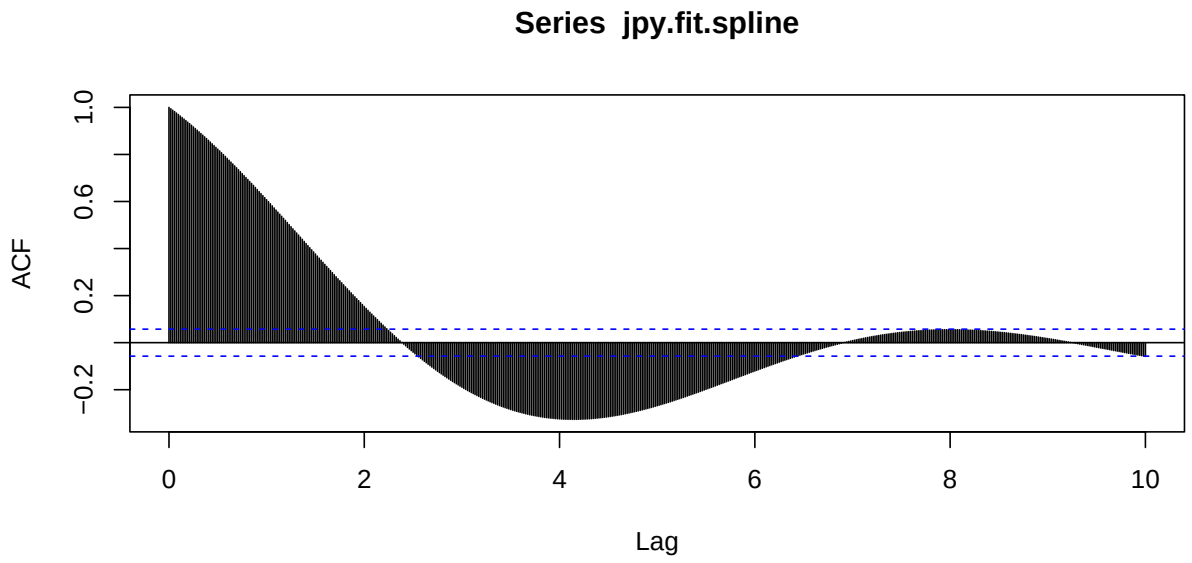

```
acf(jpy.fit.param, lag.max = 52*10)
```




```
acf(jpy.fit.local, lag.max = 52*10)
```




```
acf(jpy.fit.spline, lag.max = 52*10)
```



Response: Comments about the stationarity of the difference data: Based on the plots, we can tell that the stationarity assumption worked better on the differenced data. The ACf plots shows autocorrelation approaches to 0. Therefore, we're likely to have stationarity. Incomparison, the spline model performed the best.

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.

```
setwd("C:/Users/angel/Downloads")
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:

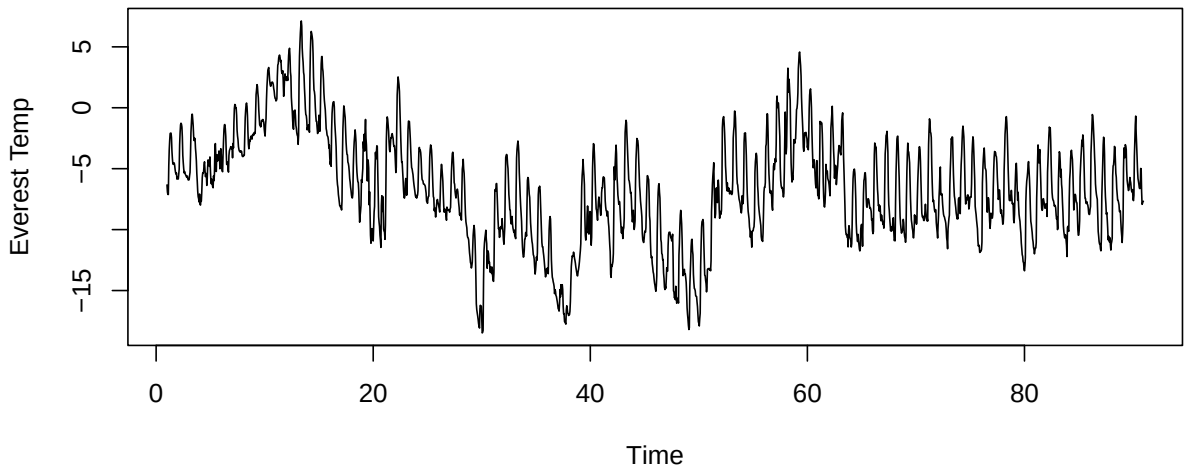
```
df$timestamp<-ymd_hms(df$timestamp)
temp <- ts(df$temp, freq = 24)

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

Question 2a: Exploratory Data Analysis

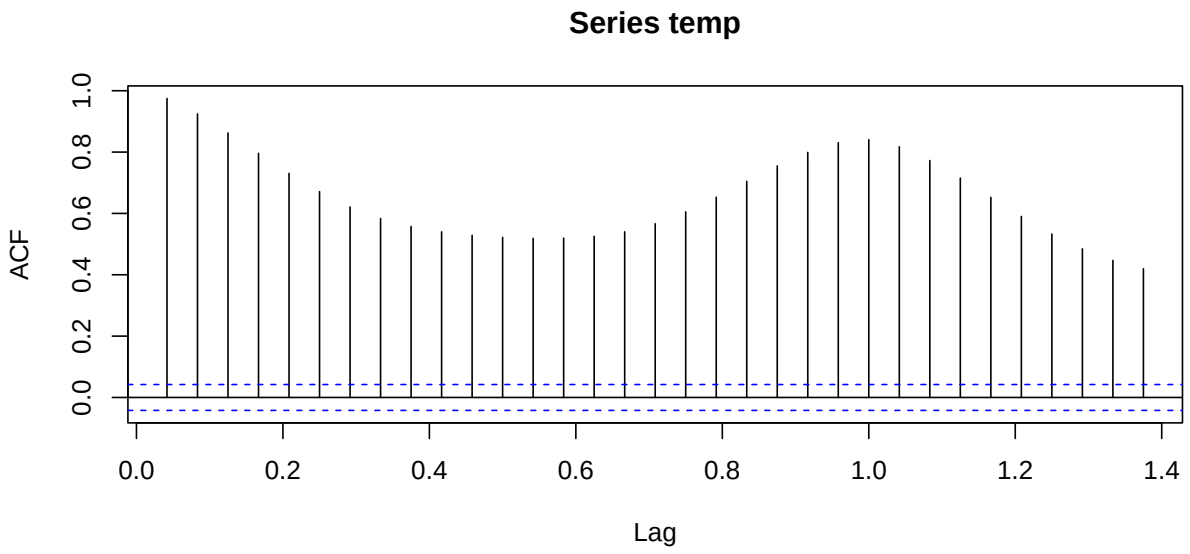
Plot both the Time Series and ACF plots. Comment on the main features, and identify what (if any) assumptions of stationarity are violated. Additionally, comment if you believe the differenced data is more appropriate for use in fitting the data. Support your response with a graphical analysis.

Hint: Make sure to use the appropriate differenced data.

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

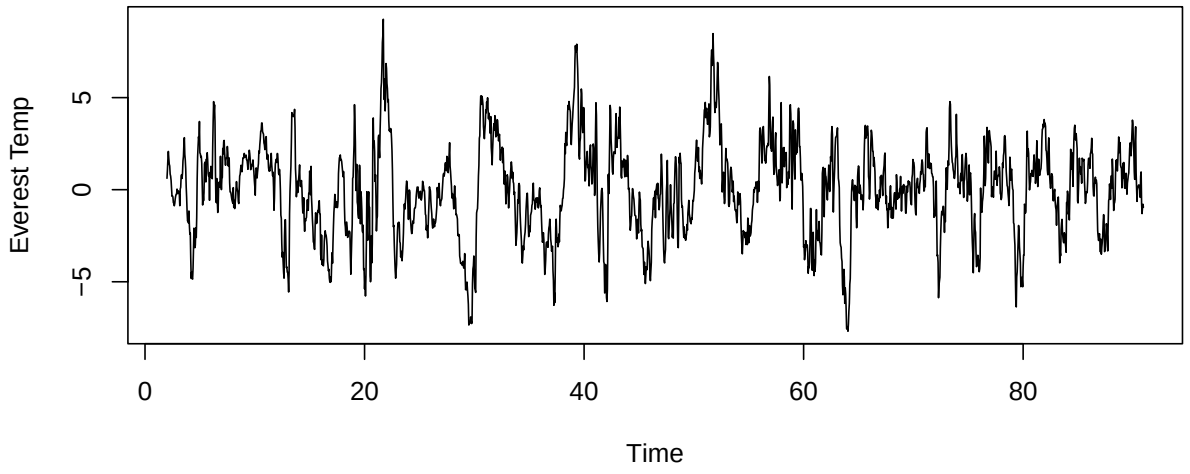



```
acf(temp)
```

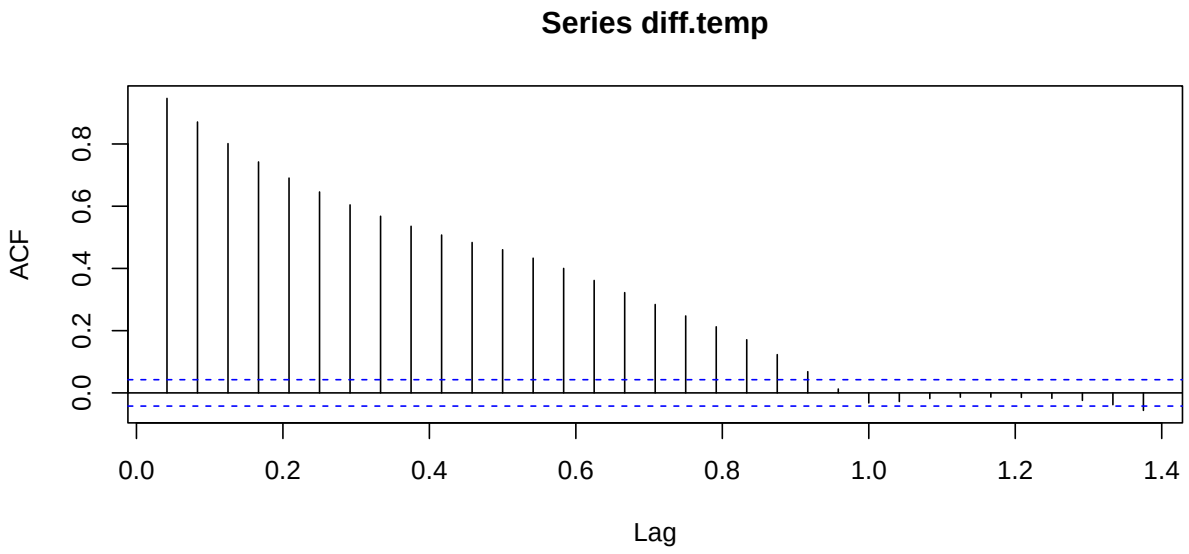


Response: Comments about the time series and ACF plots of the original time series Based on the plot,it shows cyclical seasonality and a slightly downward trend. The ACF plot shows no autocorrelation because it does not approach zero, therefore no show of stationarity.

```
diff.temp = diff(temp,24)
ts.plot(diff.temp, ylab="Everest Temp")
```




```
acf(diff.temp)
```



Response: Comments about the time series and ACF plots of the difference time series The ACF shows autocorrelation as it approaches to zero, which means we can use a differenced data for better result. BAsed on the plots, we can argue that there is a strong trend but a weak seasonality. # 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?

```
model1 <- lm(temp~harmonic(temp,1))
summary(model1)
```

```
##
## Call:
## lm(formula = temp ~ harmonic(temp, 1))
##
## 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, 1)cos(2*pi*t) -1.40540    0.12369  -11.36   <2e-16 ***
## harmonic(temp, 1)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
```

```r
#ANOVA
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|)
## (Intercept)                -8.637733   0.419589 -20.586  < 2e-16 ***
## season(temp - 1)Season-2   -0.089922   0.593388  -0.152 0.879564
## season(temp - 1)Season-3    0.007367   0.593388   0.012 0.990096
## season(temp - 1)Season-4    1.270233   0.593388   2.141 0.032416 *
## season(temp - 1)Season-5    3.556589   0.593388   5.994 2.40e-09 ***
## season(temp - 1)Season-6    4.840344   0.593388   8.157 5.79e-16 ***
## season(temp - 1)Season-7    5.439567   0.593388   9.167  < 2e-16 ***
## season(temp - 1)Season-8    5.754467   0.593388   9.698  < 2e-16 ***
## season(temp - 1)Season-9    5.580767   0.593388   9.405  < 2e-16 ***
## season(temp - 1)Season-10   5.180078   0.593388   8.730  < 2e-16 ***
## season(temp - 1)Season-11   4.415444   0.593388   7.441 1.44e-13 ***
## season(temp - 1)Season-12   3.262233   0.593388   5.498 4.31e-08 ***
## season(temp - 1)Season-13   2.256178   0.593388   3.802 0.000147 ***
## season(temp - 1)Season-14   1.569878   0.593388   2.646 0.008214 **
## season(temp - 1)Season-15   1.310044   0.593388   2.208 0.027369 *
## season(temp - 1)Season-16   1.070478   0.593388   1.804 0.071371 .
## season(temp - 1)Season-17   0.818689   0.593388   1.380 0.167828
## season(temp - 1)Season-18   0.701811   0.593388   1.183 0.237053
## season(temp - 1)Season-19   0.522100   0.593388   0.880 0.379033
## season(temp - 1)Season-20   0.408722   0.593388   0.689 0.491028
## season(temp - 1)Season-21   0.266567   0.593388   0.449 0.653313
## season(temp - 1)Season-22   0.158144   0.593388   0.267 0.789872
## season(temp - 1)Season-23   0.031811   0.593388   0.054 0.957251
## season(temp - 1)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
```

Response: Compare Seasonality Models Based the summaries, the anova model has higher performance due to higher R^2 value of 0.75 in comparison to 0.17. # 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.

```
#parametric
x1=datetime
x2=datetime^2
lm.fit <- lm(temp~x1+x2+season(temp,24))
summary(lm.fit)
```

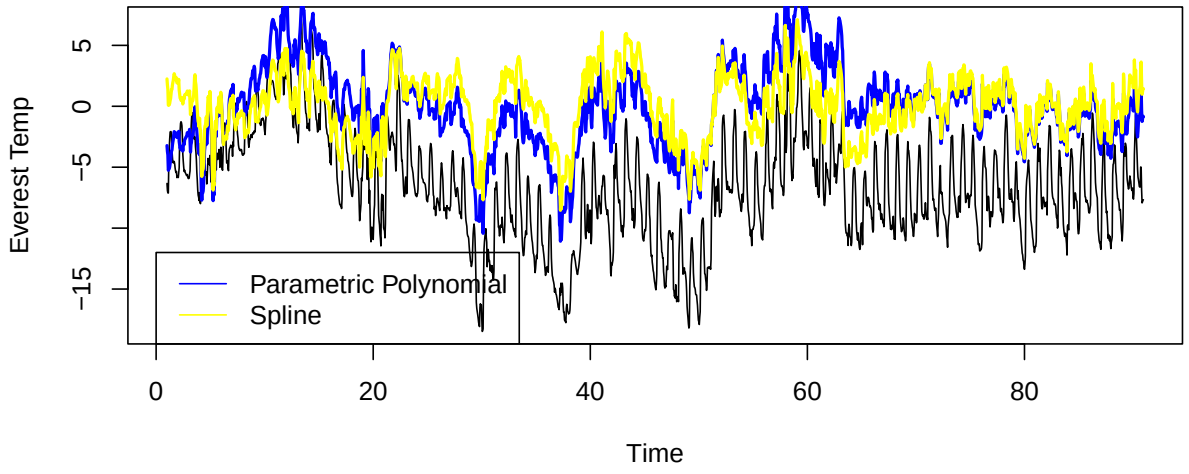
```
##
## Call:
## lm(formula = temp ~ x1 + x2 + season(temp, 24))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -11.0762  -2.0997  -0.0568   1.8439  10.9177
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)            9.381e+05  4.294e+04  21.848  < 2e-16 ***
## x1                    -1.162e-03  5.323e-05 -21.839  < 2e-16 ***
## x2                     3.601e-13  1.650e-14  21.829  < 2e-16 ***
## season(temp, 24)242   -8.804e-02  5.151e-01  -0.171  0.86432
## season(temp, 24)243    1.113e-02  5.151e-01   0.022  0.98276
## season(temp, 24)244    1.276e+00  5.151e-01   2.477  0.01333 *
## season(temp, 24)245    3.564e+00  5.151e-01   6.919 5.99e-12 ***
## season(temp, 24)246    4.850e+00  5.151e-01   9.415  < 2e-16 ***
## season(temp, 24)247    5.451e+00  5.151e-01  10.582  < 2e-16 ***
## season(temp, 24)248    5.767e+00  5.151e-01  11.197  < 2e-16 ***
## season(temp, 24)249    5.596e+00  5.151e-01  10.863  < 2e-16 ***
## season(temp, 24)2410   5.197e+00  5.151e-01  10.088  < 2e-16 ***
## season(temp, 24)2411   4.434e+00  5.151e-01   8.608  < 2e-16 ***
## season(temp, 24)2412   3.282e+00  5.151e-01   6.372 2.27e-10 ***
## season(temp, 24)2413   2.278e+00  5.151e-01   4.423 1.02e-05 ***
## season(temp, 24)2414   1.594e+00  5.151e-01   3.094  0.00200 **
## season(temp, 24)2415   1.336e+00  5.151e-01   2.593  0.00958 **
## season(temp, 24)2416   1.098e+00  5.151e-01   2.131  0.03319 *
## season(temp, 24)2417   8.478e-01  5.151e-01   1.646  0.09996 .
## season(temp, 24)2418   7.326e-01  5.151e-01   1.422  0.15510
## season(temp, 24)2419   5.546e-01  5.151e-01   1.077  0.28172
## season(temp, 24)2420   4.430e-01  5.151e-01   0.860  0.38991
## season(temp, 24)2421   3.025e-01  5.151e-01   0.587  0.55705
## season(temp, 24)2422   1.958e-01  5.151e-01   0.380  0.70388
## season(temp, 24)2423   7.117e-02  5.151e-01   0.138  0.89012
## season(temp, 24)2424   1.244e-01  5.151e-01   0.242  0.80916
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.455 on 2134 degrees of freedom
## Multiple R-squared:  0.409,  Adjusted R-squared:  0.402
```

```
## F-statistic: 59.07 on 25 and 2134 DF,  p-value: < 2.2e-16
```

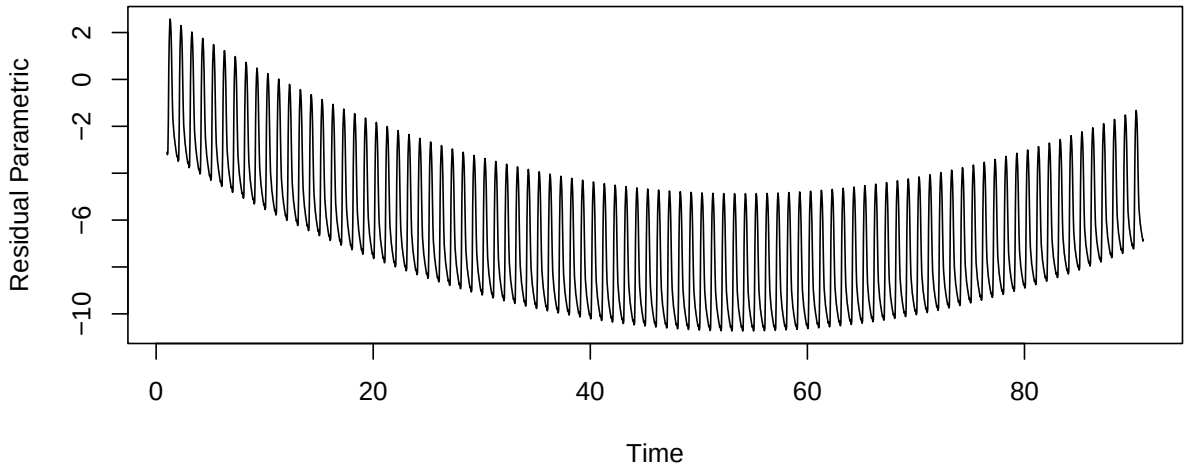
```
diff.fit.lm = ts(temp-fitted(lm.fit),start=1,frequency=24)
#non-parametric
gam.fit = gam(temp~s(datetime)+season(temp,24))
summary(gam.fit)
```

```
##
## Family: gaussian
## Link function: identity
##
## Formula:
## temp ~ s(datetime) + season(temp, 24)
##
## Parametric coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -8.640172   0.270126 -31.986  < 2e-16 ***
## season(temp, 24)242  -0.089852   0.381998  -0.235 0.814064
## season(temp, 24)243   0.007526   0.381998   0.020 0.984282
## season(temp, 24)244   1.270502   0.381999   3.326 0.000896 ***
## season(temp, 24)245   3.556986   0.381999   9.312  < 2e-16 ***
## season(temp, 24)246   4.840889   0.382000  12.672  < 2e-16 ***
## season(temp, 24)247   5.440278   0.382001  14.242  < 2e-16 ***
## season(temp, 24)248   5.755364   0.382001  15.066  < 2e-16 ***
## season(temp, 24)249   5.581870   0.382002  14.612  < 2e-16 ***
## season(temp, 24)2410  5.181406   0.382003  13.564  < 2e-16 ***
## season(temp, 24)2411  4.417017   0.382005  11.563  < 2e-16 ***
## season(temp, 24)2412  3.264069   0.382006   8.545  < 2e-16 ***
## season(temp, 24)2413  2.258297   0.382008   5.912 3.94e-09 ***
## season(temp, 24)2414  1.572299   0.382009   4.116 4.00e-05 ***
## season(temp, 24)2415  1.312787   0.382011   3.437 0.000601 ***
## season(temp, 24)2416  1.073562   0.382013   2.810 0.004995 **
## season(temp, 24)2417  0.822133   0.382015   2.152 0.031502 *
## season(temp, 24)2418  0.705635   0.382017   1.847 0.064867 .
## season(temp, 24)2419  0.526323   0.382019   1.378 0.168429
## season(temp, 24)2420  0.413364   0.382022   1.082 0.279356
## season(temp, 24)2421  0.271646   0.382024   0.711 0.477119
## season(temp, 24)2422  0.163680   0.382027   0.428 0.668365
## season(temp, 24)2423  0.037823   0.382030   0.099 0.921142
## season(temp, 24)2424  0.089886   0.382032   0.235 0.814012
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                edf Ref.df     F p-value
## s(datetime) 8.946  8.999 335.5  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.671   Deviance explained = 67.6%
## GCV = 6.6682  Scale est. = 6.5665    n = 2160
```

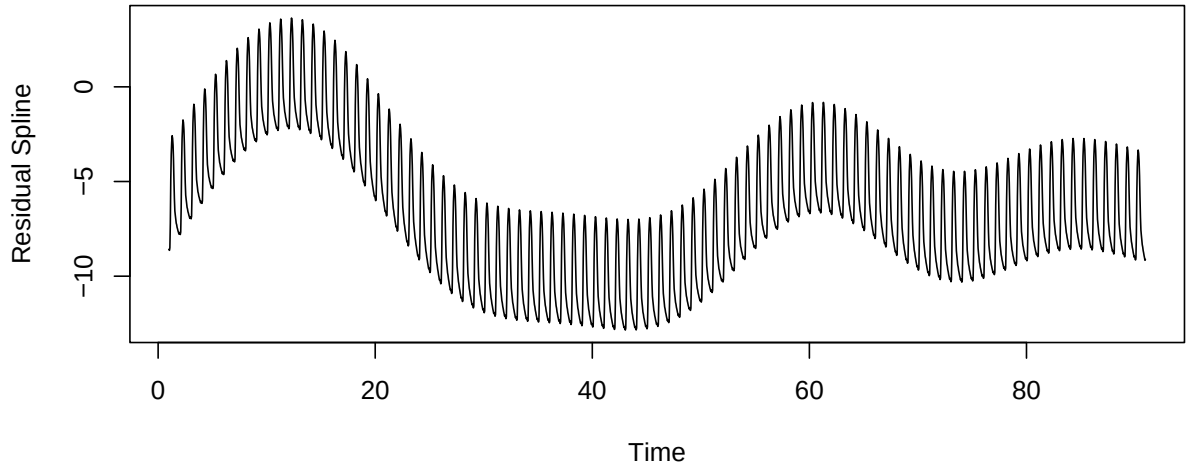
```r
diff.fit.gam=ts(temp-fitted(gam.fit), start=1, frequency = 24)

#fitted values on original
ts.plot(temp,ylab="Everest Temp")
lines(diff.fit.lm, lwd=2, col="blue")
lines(diff.fit.gam, lwd=2, col="yellow")
legend(x=0, y=-12, legend=c("Parametric Polynomial", "Spline"), lty=1, col=c("blue", "yellow"))
```



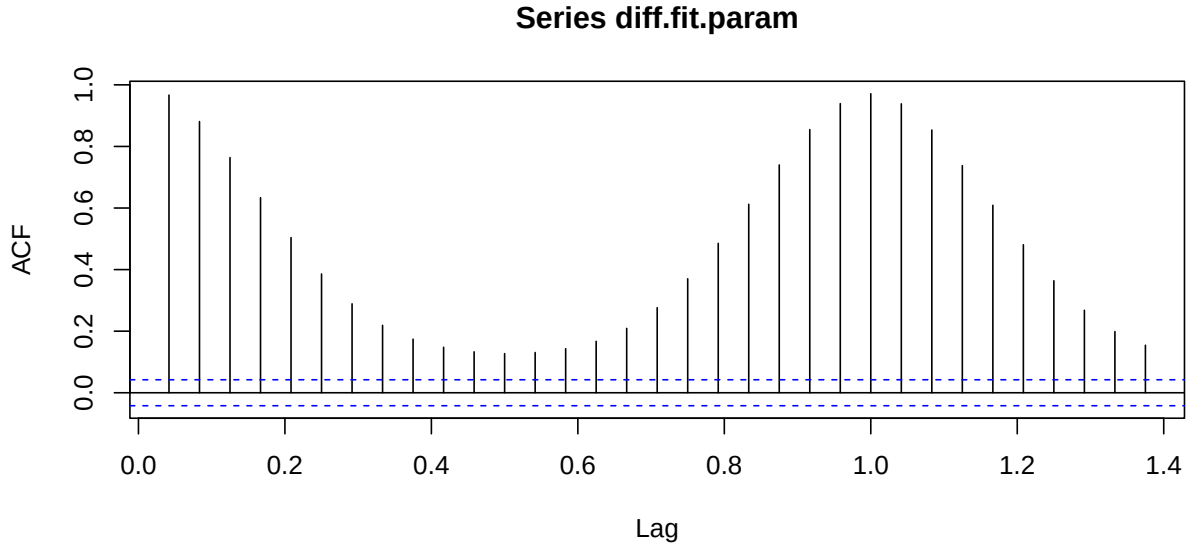

```r
#residuals with respect to time
diff.fit.param = ts(temp-diff.fit.lm, start = 1, frequency = 24)
ts.plot(diff.fit.param, ylab="Residual Parametric")
```



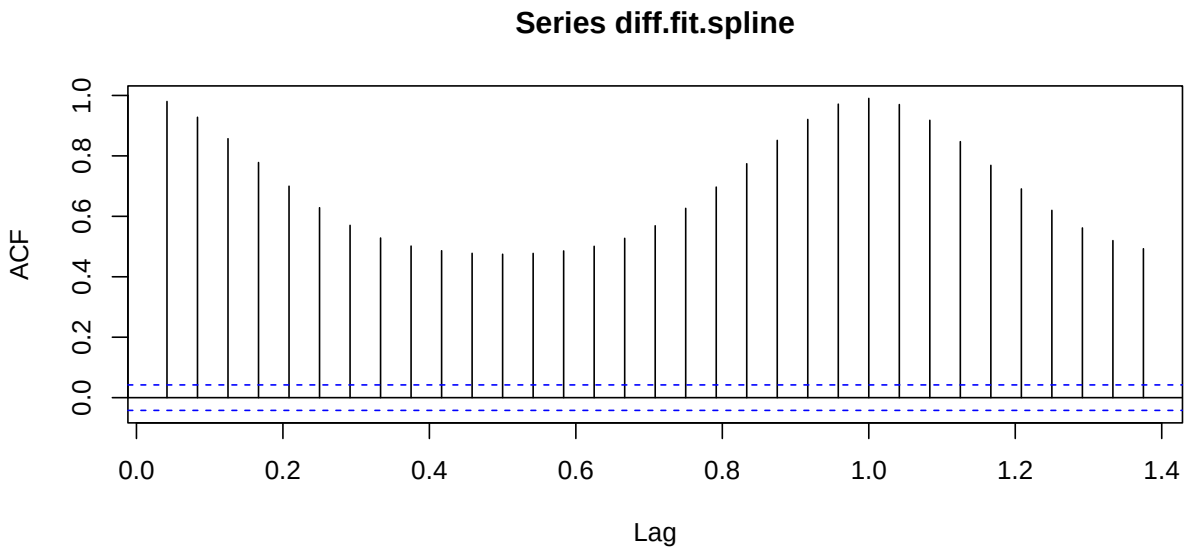
```
diff.fit.spline = ts(temp-diff.fit.gam, start = 1, frequency = 24)
ts.plot(diff.fit.spline, ylab="Residual Spline")
```




```
#ACF
acf(diff.fit.param)
```




```
acf(diff.fit.spline)
```



Response: Model Comparison Both seasonal models given seem to capture the seasonal variation , however neither of them capture the trend. It shows that both lines seem to be biased higher than the mean of actual temp. The ACF plots show that there is significant autocorrelation but some stationarity assumption does not hold. Performance wise, the spline model has better explanatory power due to its higher R^2 value.