hw1

Jim Liu

8/27/2020

ISyE 6402 Homework 1

Question 1: Temperature Analysis (30 Points)

In this problem, we will analyze aggregated temperature data.

Data file LA Temp Monthly.csvPreview the document contains the monthly average temperature of Los Angeles from January 1950 through December 2018. Run the following code to prepare the data for analysis:

```
#Note 'TSA' is now depreciated, use the following to load the parent library
library(locfit)
```

```
## locfit 1.5-9.4    2020-03-24
```

```
library(mgcv)
```

```
## Warning: package 'mgcv' was built under R version 3.6.2
```

```
## Loading required package: nlme
```

```
## This is mgcv 1.8-32. For overview type 'help("mgcv-package")'.
```

```
data <- read.csv("/Users/jim/Dropbox (GaTech)/Courses/ISyE6402/Homework/HW1/LA Temp Monthly.csv")
data <- data[,2]
#Convert to TS data in proper frame
temp <- ts(data,start=c(1950,1),freq=12)
```

Question 1a: Exploratory Data Analysis

- Plot the Time Series and ACF plots. Comment on the main features, and identify what (if any) assumptions of stationarity are violated. Hint: Before plotting, can you infer anything from the nature of the data?

```
data < 46.8
```

```
##   [1] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [13] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [25] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [37] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [49] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [61] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [73] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [85] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [97] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [109] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [121] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [133] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [145] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [157] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [169] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [181] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [193] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [205] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [217] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [229] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [241] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [253] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [265] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [277] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [289] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [301] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [313] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [325] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [337] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [349] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [361] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [373] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [385] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [397] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [409] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [421] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [433] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [445] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [457] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [469] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [481] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [493] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [505] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [517] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [529] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [541] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [553] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [565] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [577] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [589] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [601] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [613] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [625] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [637] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [649] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [661] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [673] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [685] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [697] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [709] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [721] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [733] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [745] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [757] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [769] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [781] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [793] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [805] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [817] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

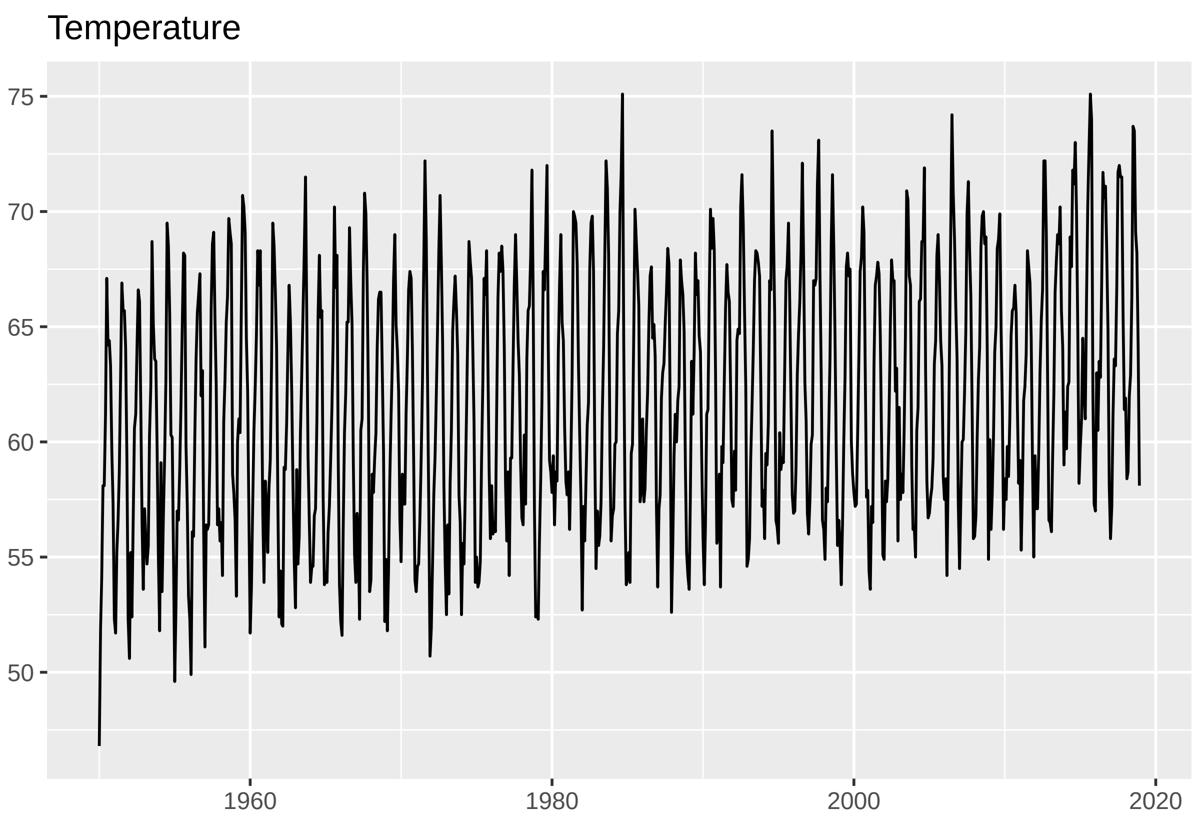
Before we plot time series, we could see there might be an increasing pattern because there is no one data point whihc is less than 46.8 except itself.

```
library(ggfortify)
```

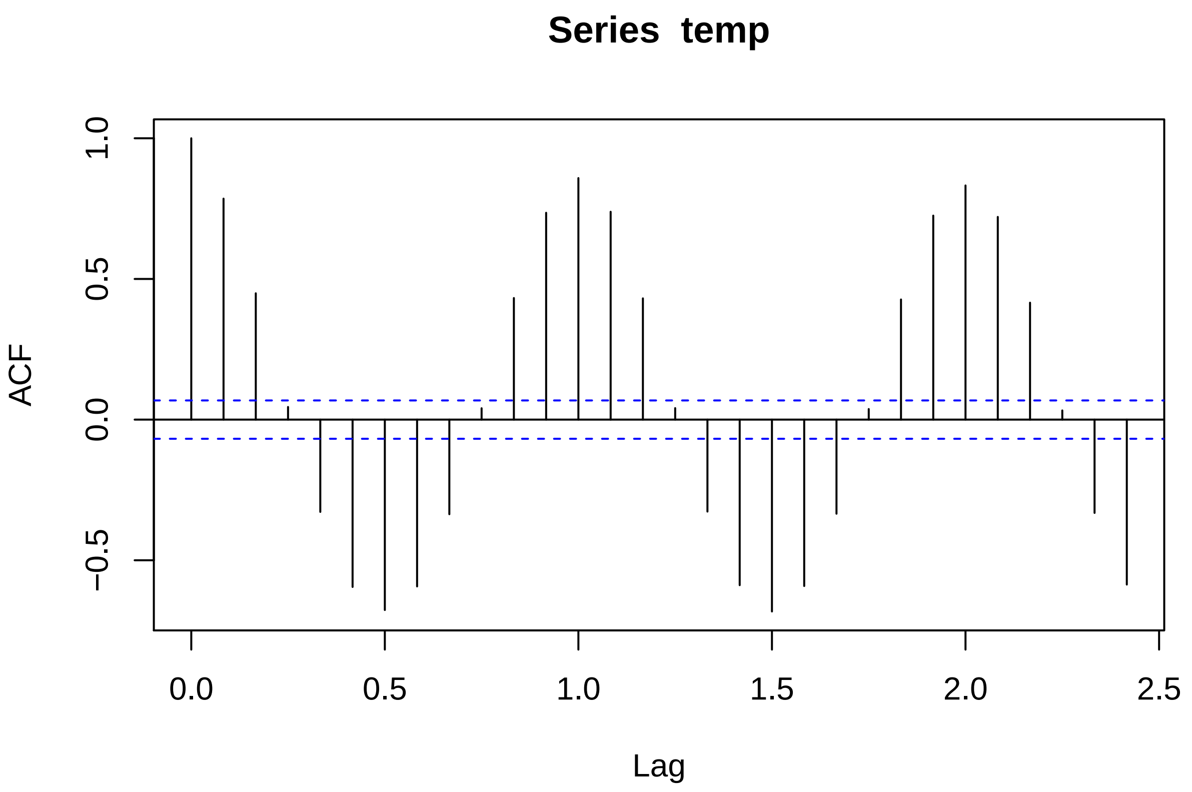
```
## Warning: package 'ggfortify' was built under R version 3.6.2
```

```
## Loading required package: ggplot2
```

```
autoplot(temp) + labs(title="Temperature")
```



```
acf(temp)
```



*

Comment on the main features, and identify what (if any) assumptions of stationarity are violated.

For the time Series plot, we could clearly see an increasing pattern of temperature from year to year. But, this increasing trend is not big, but a little slowly going up. As for the ACF plot, there is seasonality. Therefore, it violates autocovariance independent of time clearly. Plus, it also violates constant mean assumption although it is a little increasing.

- On its own, which type of model do you think will fit the data best: trend or seasonality fitting?

Seasonality model fit the data best since from acf plot, we could see a periodic pattern. And there is an increasing trend, but seasonality problem should be addressed first.

Question 1b: Trend Estimation

Fit the following trend estimation models:

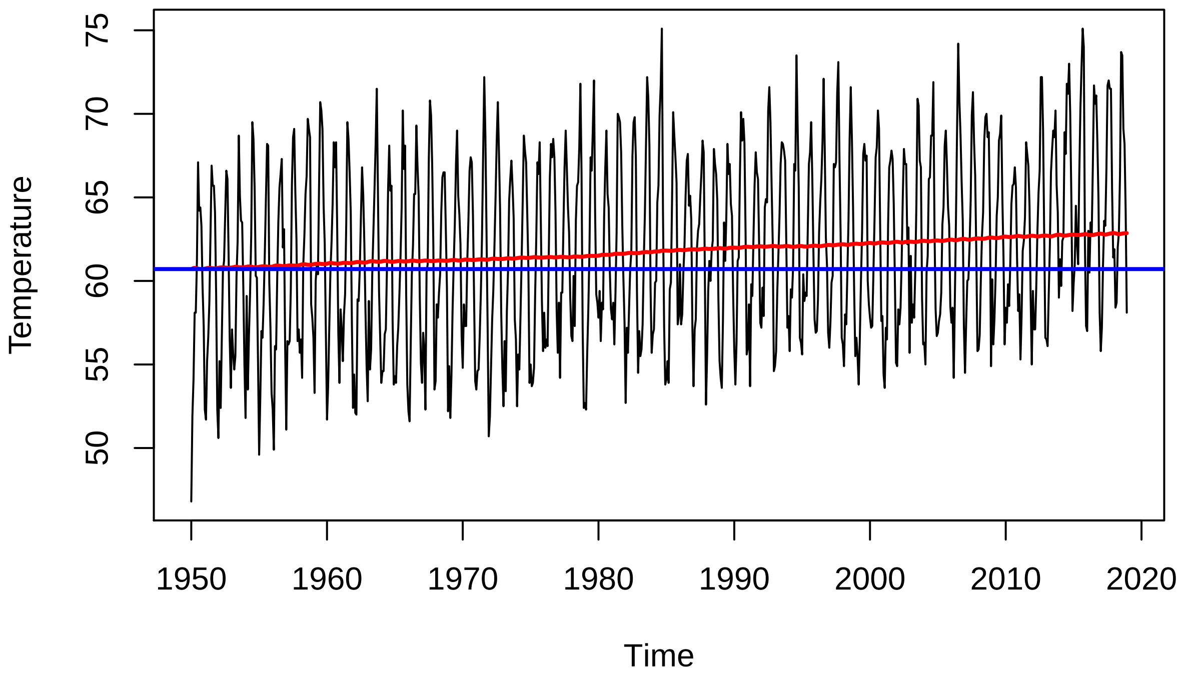
- Moving average
- Parametric quadratic polynomial
- Local Polynomial
- Splines

Overlay the fitted values on the original time series. Construct and plot the residuals with respect to time and ACF of residuals. Comment on the four models fit and on the appropriateness of the stationarity assumption of the residuals.

- Overlay the fitted values on the original time series.

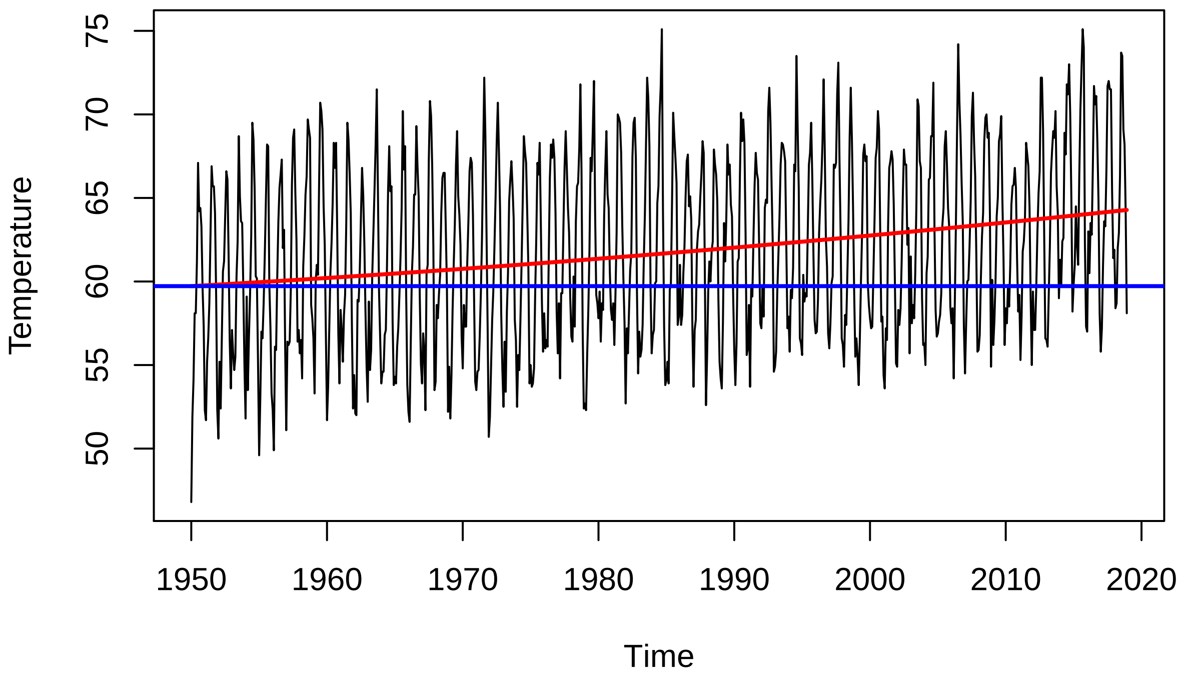
(1) Moving Average

```
time.pts = c(1:length(temp)) # 1656 even
time.pts = c(time.pts-min(time.pts))/max(time.pts)
maf = ksmooth(time.pts, temp, kernel = "box", bandwidth=1)
maf_fit = ts(maf$y, start=c(1950,1), frequency = 12)

plot(temp, ylab="Temperature")
lines(maf_fit, lwd=2, col='red') # use kernel
abline(maf_fit[1],0,lwd=2, col='blue') # compare trend
```



(2) Parametric quadratic polynomial

```
poly_model = lm(data ~ poly(time.pts,degree=2))
poly_model_fit = ts(fitted(poly_model),start=c(1950,1), frequency = 12)
plot(temp, ylab="Temperature")
lines(poly_model_fit, lwd=2, col='red')
abline(poly_model_fit[1], 0, lwd=2, col='blue')
```

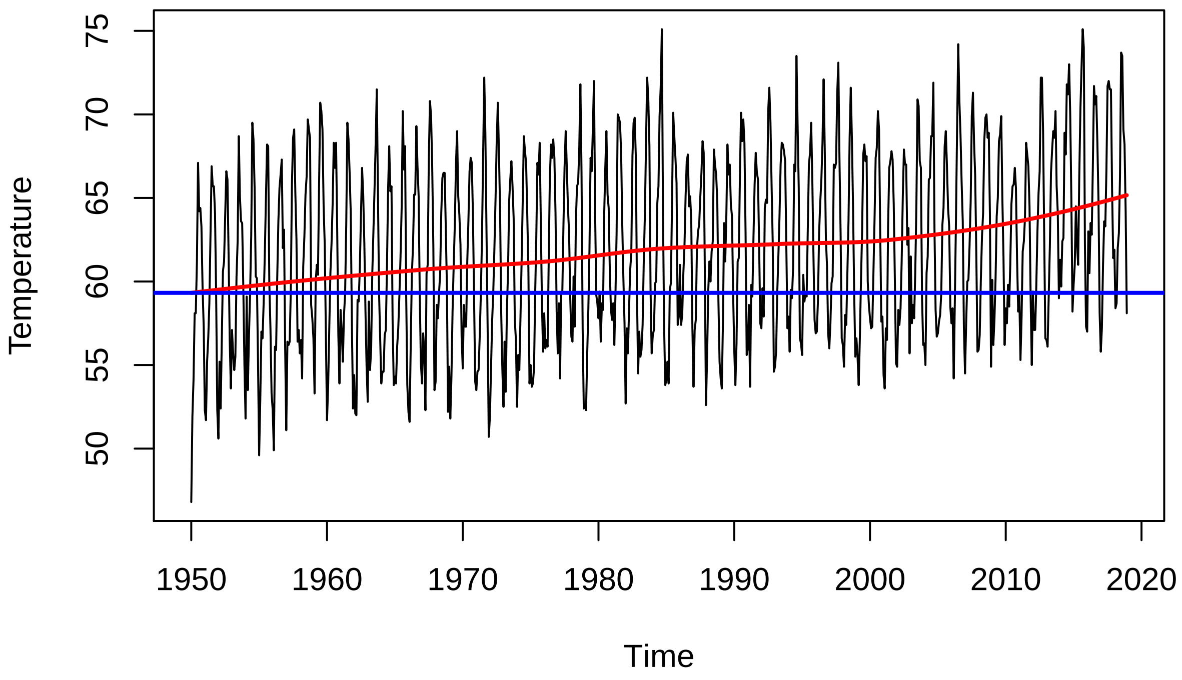


```
summary(poly_model)
```

```
##
## Call:
## lm(formula = data ~ poly(time.pts, degree = 2))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.9226  -4.4207  -0.2526   4.6385  13.4337
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    61.7694     0.1834 336.800  < 2e-16 ***
## poly(time.pts, degree = 2)1   37.9216     5.2774   7.186  1.5e-12 ***
## poly(time.pts, degree = 2)2    3.0092     5.2774   0.570    0.569
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.277 on 825 degrees of freedom
## Multiple R-squared:  0.05925,    Adjusted R-squared:  0.05697
## F-statistic: 25.98 on 2 and 825 DF,  p-value: 1.143e-11
```

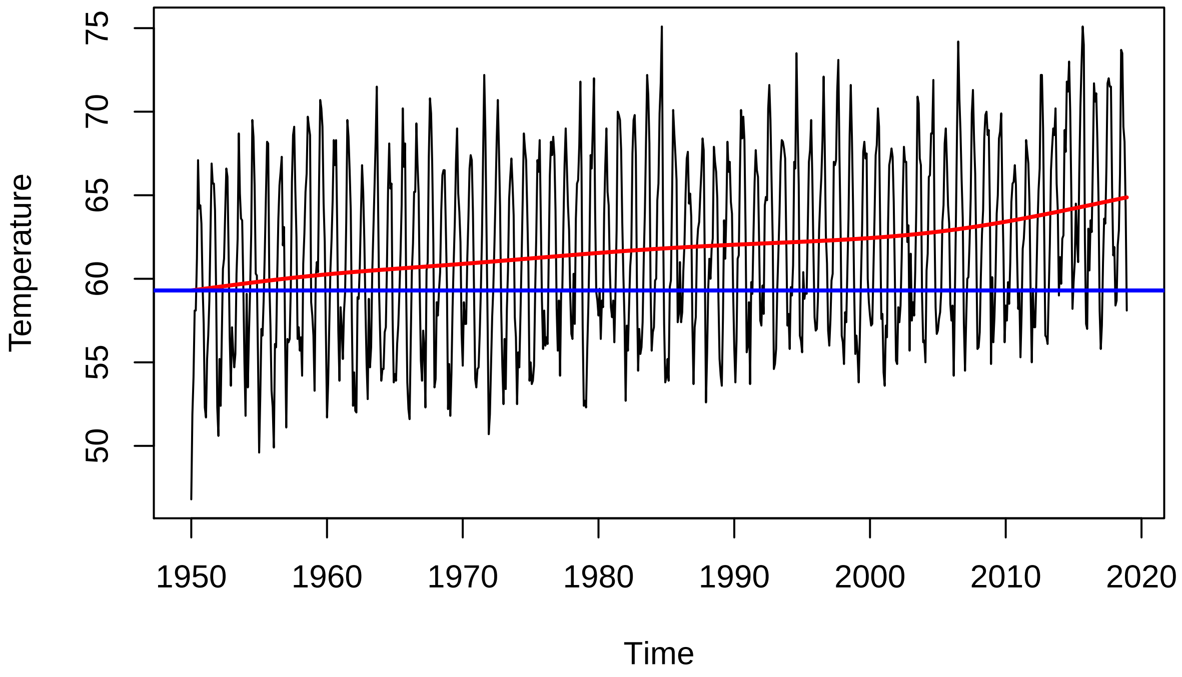
(3) Local Polynomial

```
# loess
loc_fit = loess(temp~time.pts)
temp_loc_fit = ts(fitted(loc_fit), start=c(1950,1), frequency = 12)
plot(temp, ylab="Temperature")
lines(temp_loc_fit, lwd=2, col='red')
abline(temp_loc_fit[1], 0, lwd=2, col='blue')
```



(4) Splines

```
sp_fit = gam(temp~s(time.pts))
temp_sp_fit = ts(fitted(sp_fit), start=c(1950,1), frequency = 12)

plot(temp, ylab="Temperature")
lines(temp_sp_fit, lwd=2, col='red')
abline(temp_sp_fit[1], 0, lwd=2, col="blue")
```

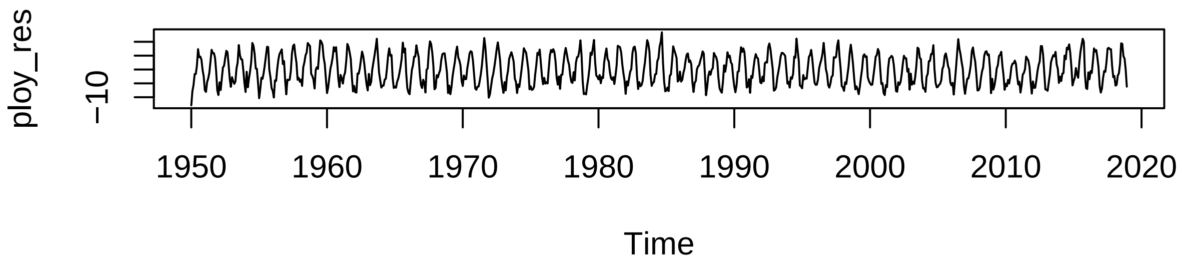


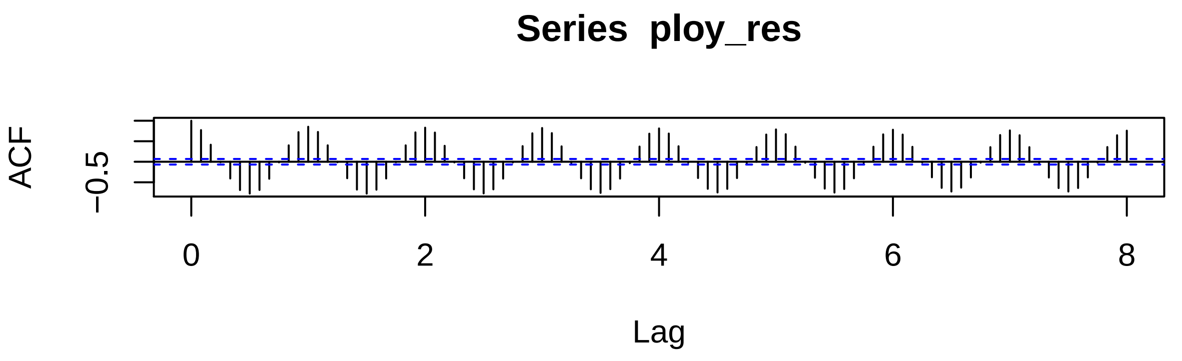
```
summary(temp_sp_fit)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   59.30   60.73   61.80   61.77   62.54   64.87
```

- Construct and plot the residuals with respect to time and ACF of residuals.

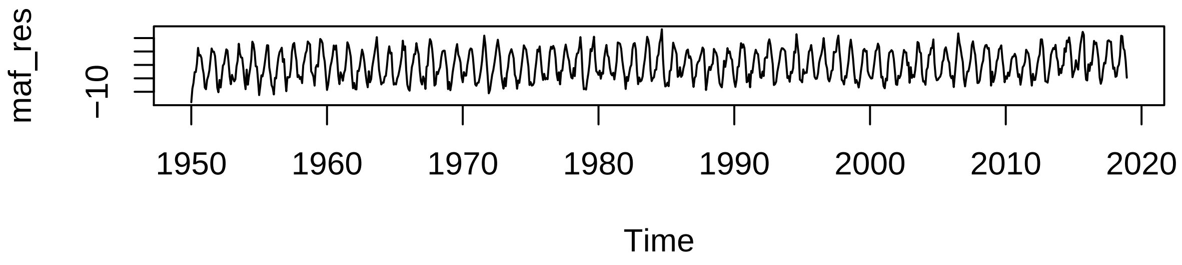
```
par(mfrow=c(2,1))
ploy_res = temp - poly_model_fit
plot(ploy_res)
acf(ploy_res, lag.max=12*8)
```

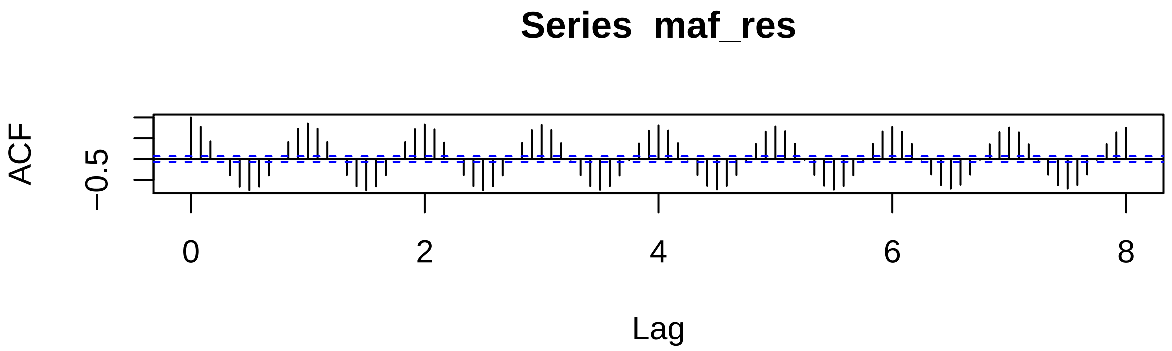




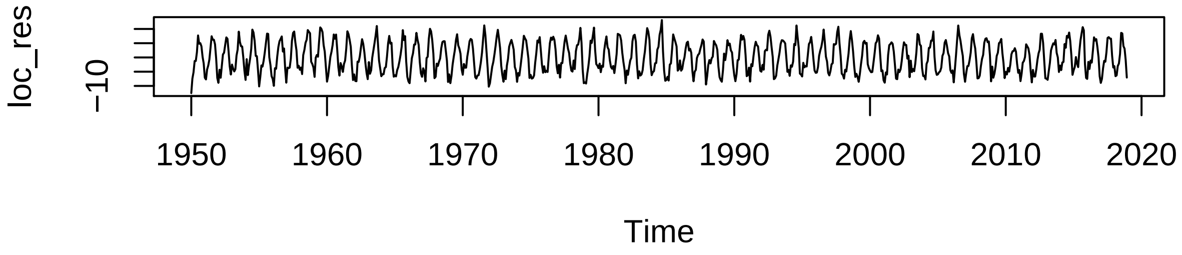
```
# maf_res = maf_fit - temp
# acf(maf_res)
```

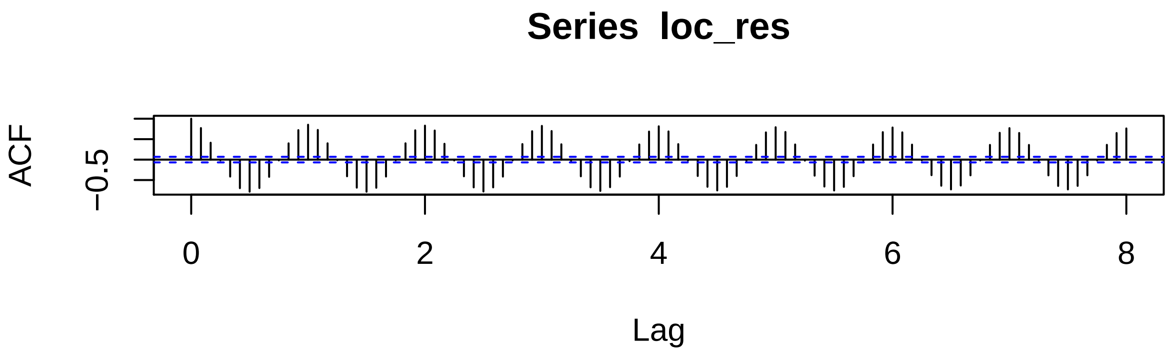
```
par(mfrow=c(2,1))
maf_res = temp - maf_fit
plot(maf_res)
acf(maf_res, lag.max=12*8)
```





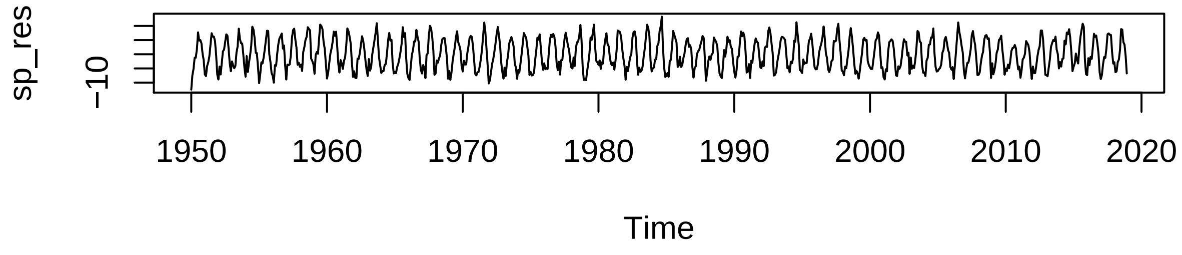
```r
par(mfrow=c(2,1))
loc_res = temp - temp_loc_fit
plot(loc_res)
acf(loc_res, lag.max=12*8)
```

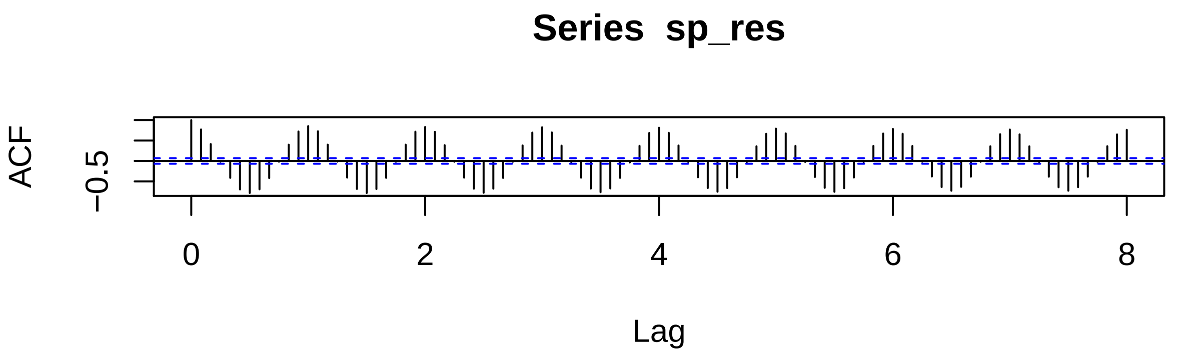




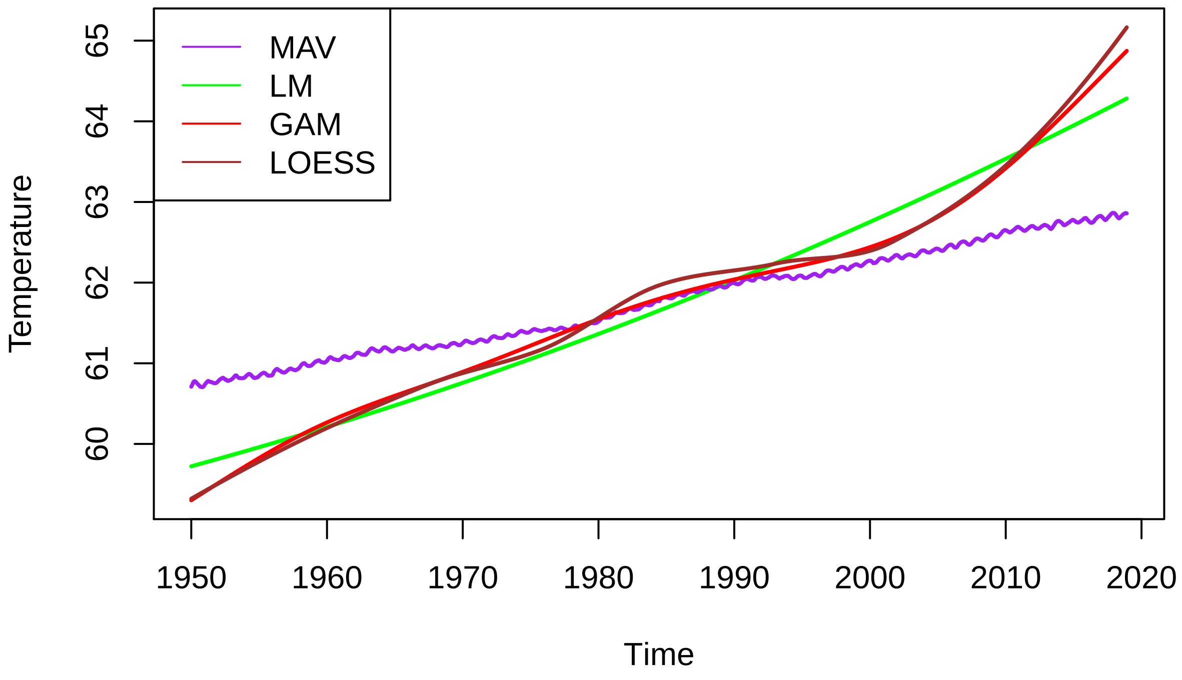
```r
par(mfrow=c(2,1))

sp_res = temp - temp_sp_fit
```

```
plot(sp_res)
acf(sp_res, lag.max=12*8)
```





```
all_val = c(maf_fit, poly_model_fit, temp_loc_fit, temp_sp_fit)
y_bound = c(min(all_val), max(all_val))
plot(poly_model_fit, lwd=2, col="green", ylim=y_bound,ylab="Temperature")
lines(maf_fit, lwd=2, col="purple")
lines(temp_sp_fit, lwd=2, col="red")
lines(temp_loc_fit, lwd=2, col="brown")
legend(x= "topleft",legend=c("MAV","LM","GAM","LOESS"),lty=1, col=c("purple","green","red","brown"))
```



- Comment on the four models fit and on the appropriateness of the stationarity assumption of the residuals.

Based on above plot, we could see GAM and LOESS go up quickly after 2010. I think it is not good estimation because global warming or temperature increasing would not increase quickly so I prefer using moving average method at this point. As for stationary assumption, in this time series data, the most important issue is seasonality. By just estimating trend, the seasonality pattern still exists from ACF plots.

Question 1c: Seasonality Estimation

Seasonality Estimation:

Fit the following seasonality estimation models.

- Categorical Linear Regression (ANOVA)
- COS-SIN

Overlay the fitted values on the original time series. Construct and plot the residuals with respect to time and ACF plots. Comment on how the two models fit and on the appropriateness of the stationarity assumption of the residuals. Also compare the fits to those in part B and comment if your initial prediction was correct.

```r
library(dynlm)
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 3.6.2
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```r
# Seasonal Means Model
model1 = dynlm(temp~season(temp))
summary(model1)
```

```
##
## Time series regression with "ts" data:
## Start = 1950(1), End = 2018(12)
##
## Call:
## dynlm(formula = temp ~ season(temp))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7145 -1.5855 -0.0319  1.6000  8.6768
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      55.5145     0.2864 193.867  < 2e-16 ***
## season(temp)Feb   0.7420     0.4050   1.832 0.067266 .
## season(temp)Mar   1.5333     0.4050   3.786 0.000164 ***
## season(temp)Apr   3.6188     0.4050   8.936  < 2e-16 ***
## season(temp)May   6.0797     0.4050  15.013  < 2e-16 ***
## season(temp)Jun   9.1014     0.4050  22.475  < 2e-16 ***
## season(temp)Jul  12.4420     0.4050  30.724  < 2e-16 ***
## season(temp)Aug  13.4681     0.4050  33.258  < 2e-16 ***
## season(temp)Sep  12.7855     0.4050  31.572  < 2e-16 ***
## season(temp)Oct   9.8087     0.4050  24.221  < 2e-16 ***
## season(temp)Nov   4.9609     0.4050  12.250  < 2e-16 ***
## season(temp)Dec   0.5188     0.4050   1.281 0.200487
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.379 on 816 degrees of freedom
## Multiple R-squared:  0.811,  Adjusted R-squared:  0.8084
## F-statistic: 318.3 on 11 and 816 DF,  p-value: < 2.2e-16
```

```
## without intercept
model2 = dynlm(temp~season(temp)+0)
summary(model2)
```

```
##
## Time series regression with "ts" data:
## Start = 1950(1), End = 2018(12)
##
## Call:
## dynlm(formula = temp ~ season(temp) + 0)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -8.7145 -1.5855 -0.0319  1.6000  8.6768
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## season(temp)Jan  55.5145     0.2864   193.9   <2e-16 ***
## season(temp)Feb  56.2565     0.2864   196.5   <2e-16 ***
## season(temp)Mar  57.0478     0.2864   199.2   <2e-16 ***
## season(temp)Apr  59.1333     0.2864   206.5   <2e-16 ***
## season(temp)May  61.5942     0.2864   215.1   <2e-16 ***
## season(temp)Jun  64.6159     0.2864   225.7   <2e-16 ***
## season(temp)Jul  67.9565     0.2864   237.3   <2e-16 ***
## season(temp)Aug  68.9826     0.2864   240.9   <2e-16 ***
## season(temp)Sep  68.3000     0.2864   238.5   <2e-16 ***
## season(temp)Oct  65.3232     0.2864   228.1   <2e-16 ***
## season(temp)Nov  60.4754     0.2864   211.2   <2e-16 ***
## season(temp)Dec  56.0333     0.2864   195.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.379 on 816 degrees of freedom
## Multiple R-squared:  0.9985, Adjusted R-squared:  0.9985
## F-statistic: 4.682e+04 on 12 and 816 DF,  p-value: < 2.2e-16
```

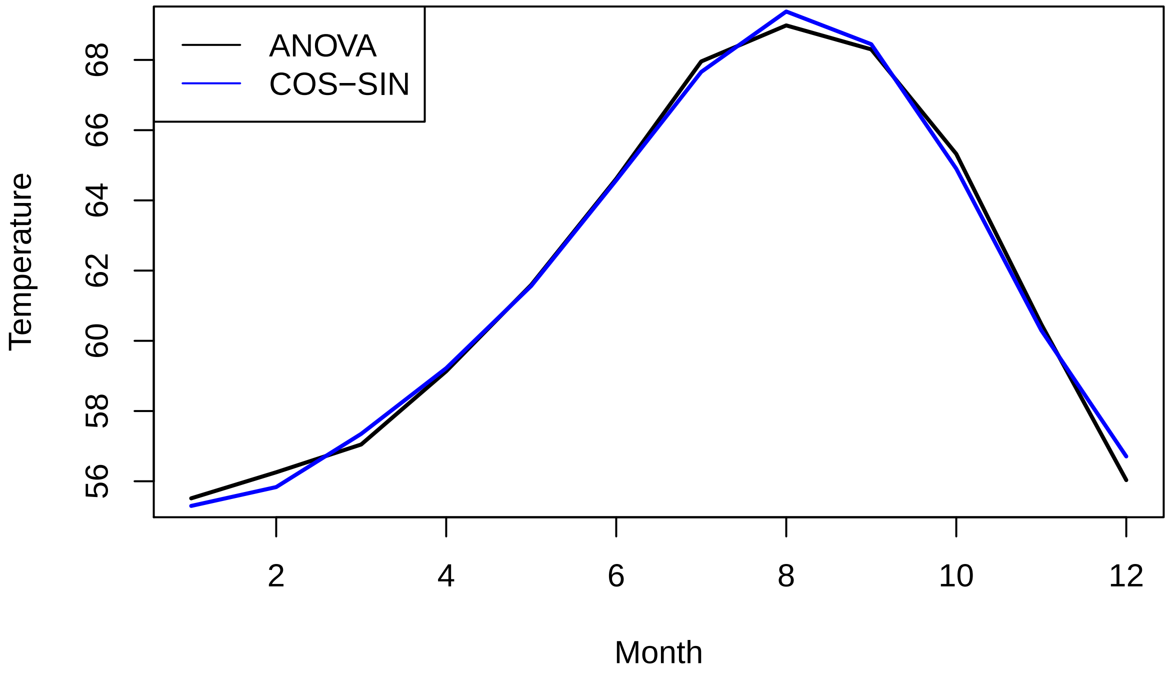
```r
## harmonic
model3 = dynlm(temp~harmon(temp, 1))
summary(model3)
```

```
##
## Time series regression with "ts" data:
## Start = 1950(1), End = 2018(12)
##
## Call:
## dynlm(formula = temp ~ harmon(temp, 1))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.789 -1.742 -0.109  1.573  9.391
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         61.76944    0.08786  703.06   <2e-16 ***
## harmon(temp, 1)cos -6.18023    0.12425  -49.74   <2e-16 ***
## harmon(temp, 1)sin -2.83955    0.12425  -22.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.528 on 825 degrees of freedom
## Multiple R-squared:  0.7841, Adjusted R-squared:  0.7836
## F-statistic:  1498 on 2 and 825 DF,  p-value: < 2.2e-16
```

```r
model4 = dynlm(temp~harmon(temp, 2))
summary(model4)
```

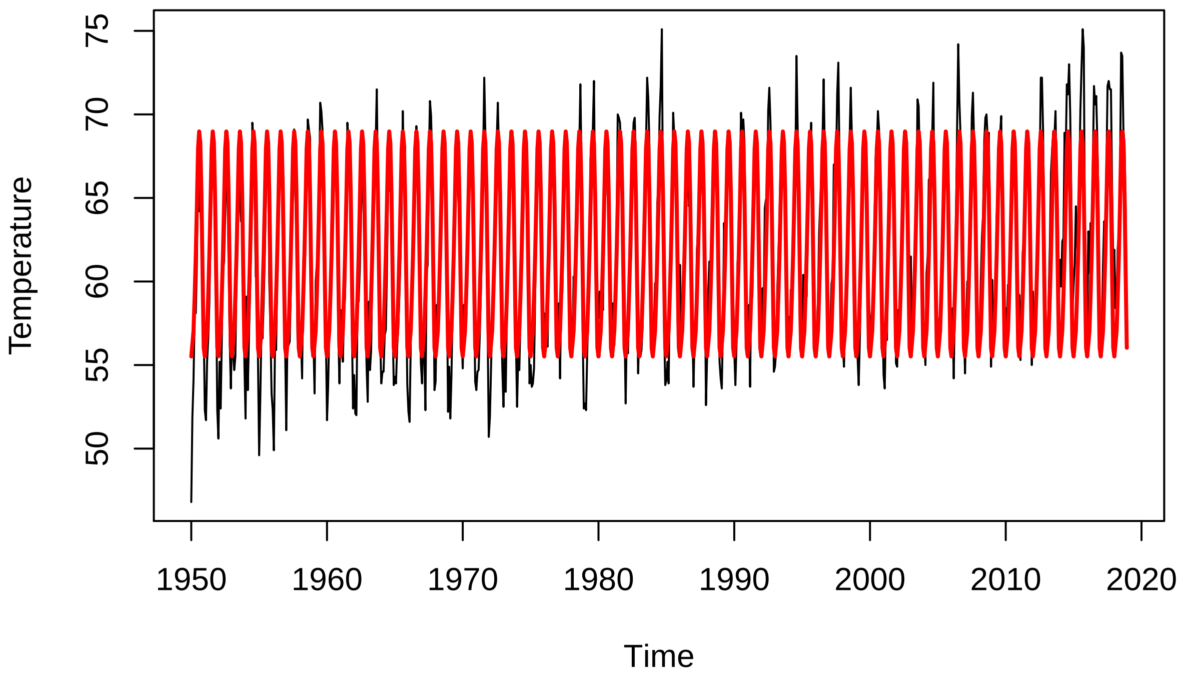
```
##
## Time series regression with "ts" data:
## Start = 1950(1), End = 2018(12)
##
## Call:
## dynlm(formula = temp ~ harmon(temp, 2))
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -8.498 -1.653 -0.148  1.572  9.099
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)          61.76944    0.08308 743.484   <2e-16 ***
## harmon(temp, 2)cos1 -6.18023    0.11749 -52.600   <2e-16 ***
## harmon(temp, 2)cos2 -0.29167    0.11749  -2.482   0.0132 *
## harmon(temp, 2)sin1 -2.83955    0.11749 -24.168   <2e-16 ***
## harmon(temp, 2)sin2  1.13566    0.11749   9.666   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.391 on 823 degrees of freedom
## Multiple R-squared:  0.8074, Adjusted R-squared:  0.8065
## F-statistic: 862.6 on 4 and 823 DF,  p-value: < 2.2e-16
```

```
series1 = coef(model2)
series2 = fitted(model4)[1:12]
plot(1:12, series1, lwd=2, type="l",xlab="Month", ylab="Temperature")
lines(1:12, series2, lwd=2, col="blue")
legend(x= "topleft",legend=c("ANOVA","COS-SIN"), lty=1, col=c("BLACK","blue"))
```

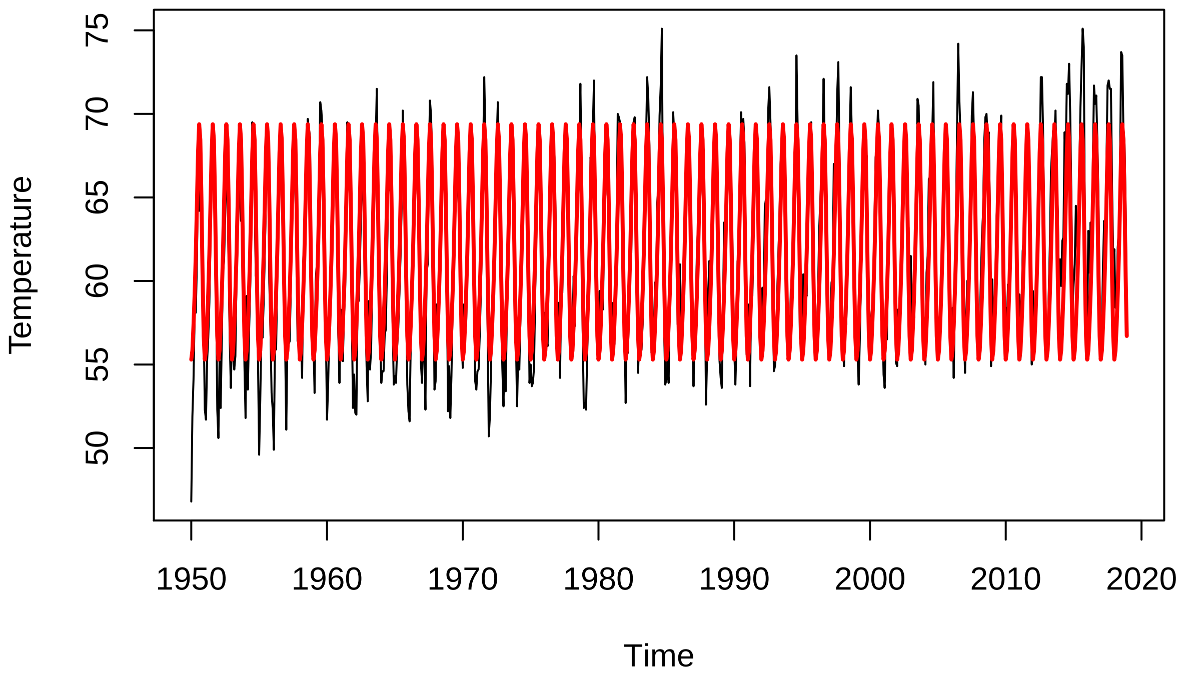


- Overlay the fitted values on the original time series.

```
cat_lm = ts(fitted(model2), start=c(1950,1), frequency = 12)
plot(temp, ylab="Temperature")
lines(cat_lm, lwd=2, col='red')
```



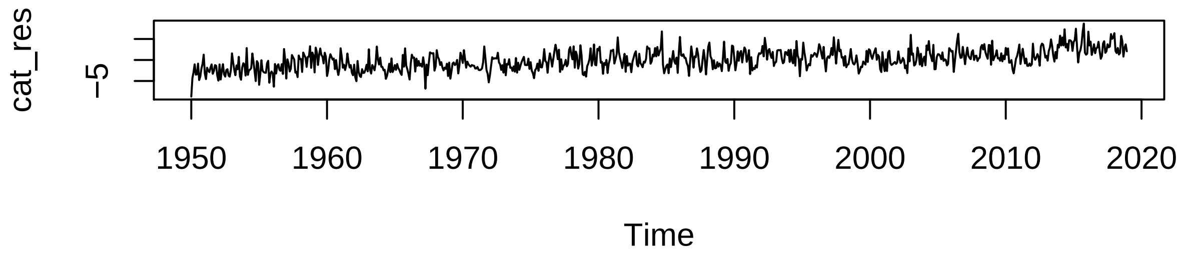
```r
harmon_lm = ts(fitted(model4), start=c(1950,1), frequency = 12)
plot(temp, ylab="Temperature")
lines(harmon_lm, lwd=2, col='red')
```

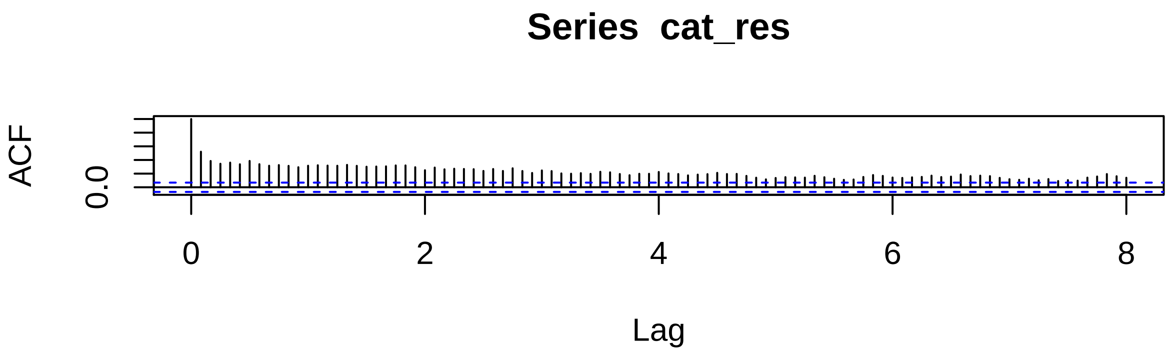


- Construct and plot the residuals with respect to time and ACF plots.

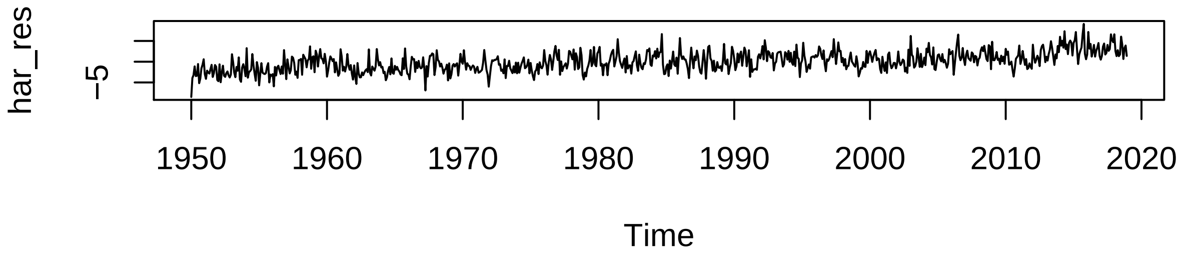
```r
par(mfrow=c(2,1))

cat_res = temp - cat_lm
```

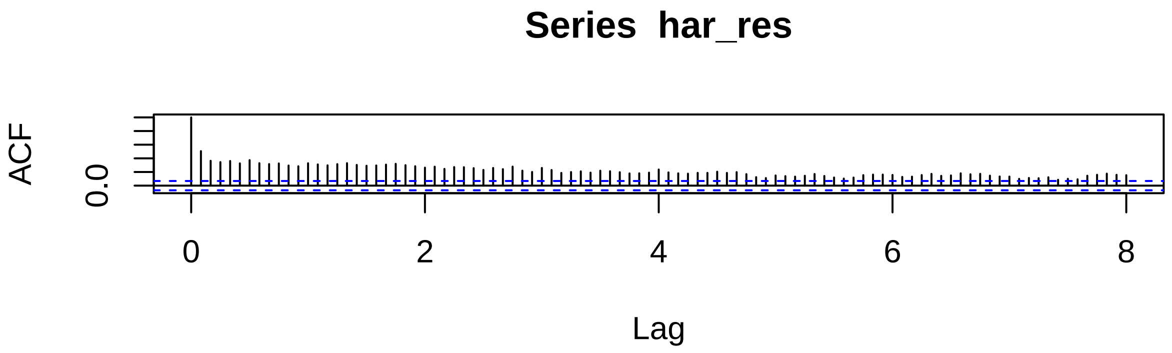
```
plot(cat_res)
acf(cat_res, lag.max=12*8)
```





```
par(mfrow=c(2,1))

har_res = temp - harmon_lm
plot(har_res)
acf(har_res, lag.max=12*8)
```





- Comment on how the two models fit and on the appropriateness of the stationarity assumption of the residuals. Also compare the fits to those in part B and comment if your initial prediction was correct.

As for two models, there is no so much difference. True, based on two methods, we would like to address seasonality first. But after seeing the ACF plots for seaonality estimation, we think we should estimate trend and seasonality at the same time. ACF plots shows that after seasonality removal, there is a trend pattern that decays with time.

Question 2: Currency Conversion Analysis (40 Points)

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

File USD-EUR Exchange.csvPreview the document contains the average exchange rate of USD/EUR from January 2000 through June 2019, aggregated weekly (ending Wednesdays). Load and prep the data for analysis with the following code:

```r
library(locfit)
library(mgcv)
#Load data
data2 <- read.csv("/Users/jim/Dropbox (GaTech)/Courses/ISyE6402/Homework/HW1/USD-EUR Exchange.csv")
data2 <- data2[,2]
#Convert to TS data in proper frame
rate <- ts(data2,start=c(2000,1),freq=52)
#Generate differenced data
rate.dif <- diff(rate)
```

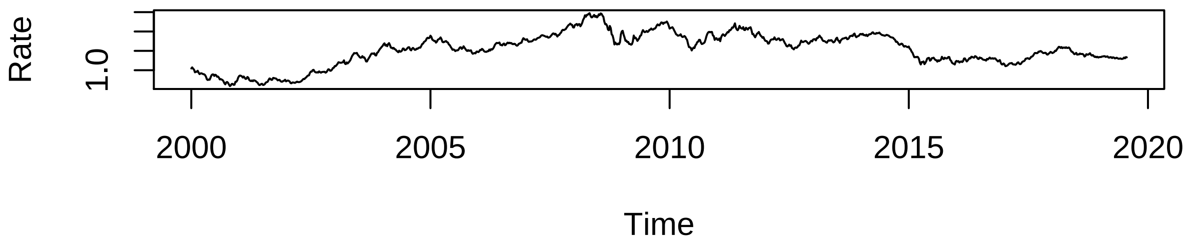
Question 2a: Exploratory Data Analysis

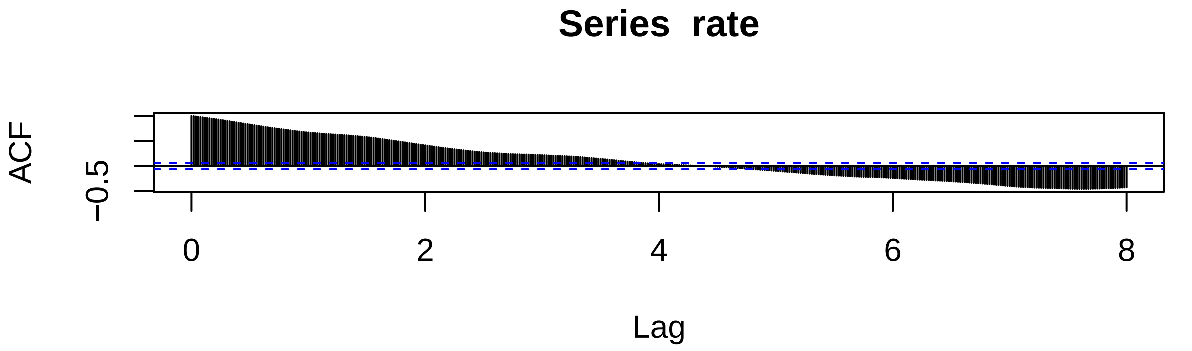
Plot the Time Series and ACF plots. Comment on the main features, and identify what (if any) assumptions of stationarity are violated.

Using the differenced rate data ('rate.dif'), 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 analysis. Support your position with your graphical analysis.

- Plot the Time Series and ACF plots. Comment on the main features, and identify what (if any) assumptions of stationarity are violated.

```r
par(mfrow=c(2,1))
ts.plot(rate, ylab="Rate")
acf(rate, lag.max = 52 * 8)
```

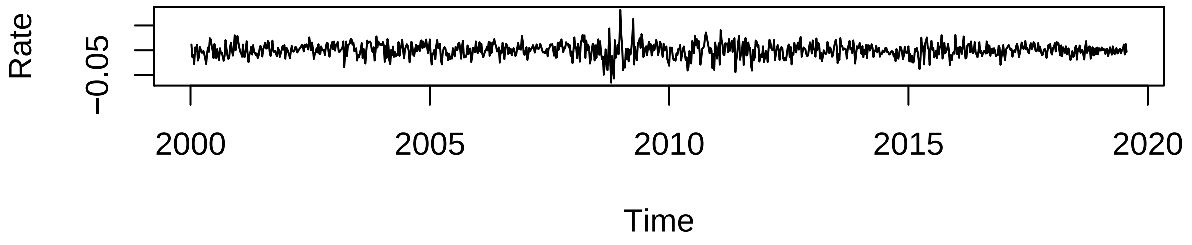


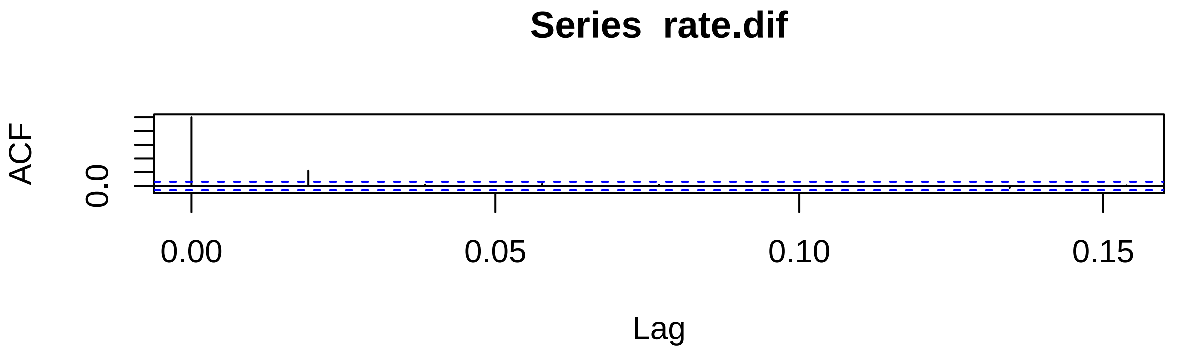


Seeing these two plots, we could not tell there is a trend or seasonality from time series plot. But, ACF plot tells me there is a trend increasing and then decreasing.

- Using the differenced rate data (‘rate.dif’), plot both the Time Series and ACF plots.
- plot both the Time Series and ACF plots

```r
par(mfrow=c(2,1))
ts.plot(rate.dif, ylab="Rate")
acf(rate.dif, lag.max = 8)
```





- 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 analysis. Support your position with your graphical analysis.

From the time series plot, except around 2008 (Financial Crisis) with large fluctuation, the time series is seemly like a stationary process. And, from the ACF plot, we could see it is like stationary process. Plus, based on the plots, there is no violations for assumptionas of stationarity. So differenced data is more appropriate.

Question 2b: Trend-Seasonality Estimation

Using the original time series data, fit the following models to estimate both trend and seasonality:

- Parametric Polynomial Regression
- Non-parametric model

Overlay the fitted values on the original time series. Construct and plot the residuals with respect to time and ACF of residuals. Comment on how the two models fit and on the appropriateness of the stationarity assumption of the residuals. For sake of simplicity, only use Categorical Regression (ANOVA) seasonality modelling.

- Parametric Polynomial Regression

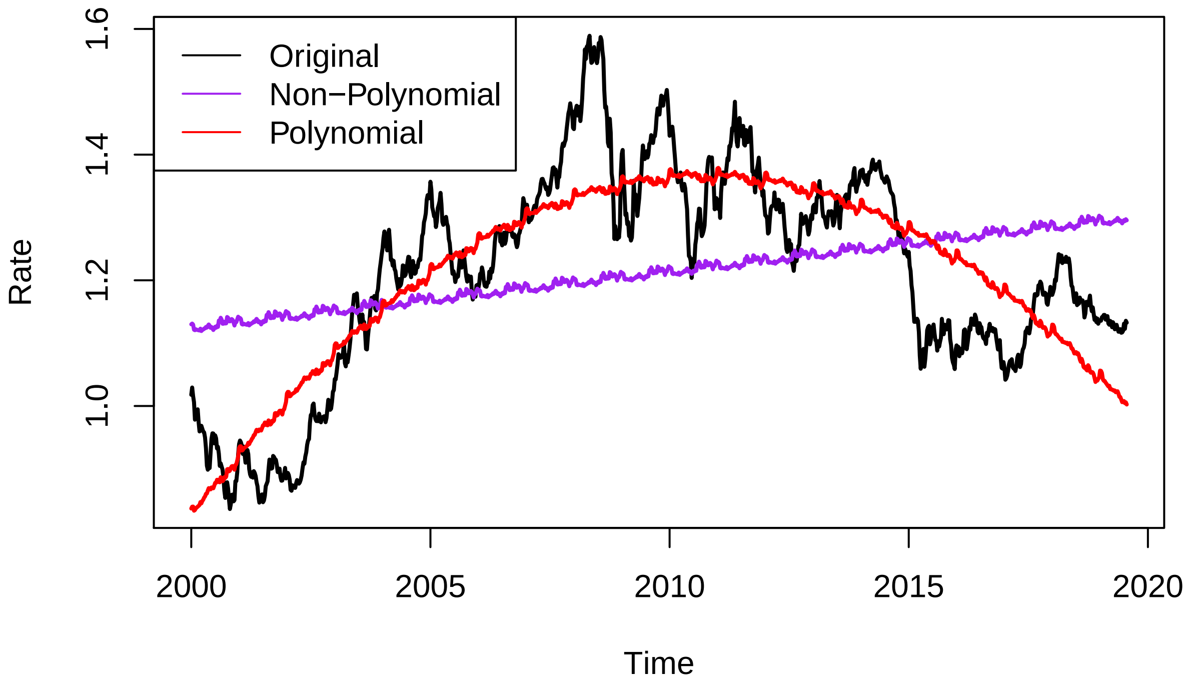
```r
library(zoo)
library(dynlm)
lm_fit_par = dynlm(rate~trend(rate)+season(rate))
## Polynomial
poly_time.pts = c(1:length(rate))
poly_time.pts = c(poly_time.pts-min(poly_time.pts))/max(poly_time.pts)
x1_square = poly_time.pts
x2_square = poly_time.pts^2
lm_fit_square = dynlm(rate~x1_square+x2_square+season(rate))
summary(lm_fit_par)
```

```
##
## Time series regression with "ts" data:
## Start = 2000(1), End = 2019(30)
##
## Call:
## dynlm(formula = rate ~ trend(rate) + season(rate))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29849 -0.15166  0.03484  0.11600  0.39295
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    1.130e+00  3.776e-02  29.921   <2e-16 ***
## trend(rate)    8.930e-03  9.118e-04   9.794   <2e-16 ***
## season(rate)2  1.030e-03  5.197e-02   0.020    0.984
## season(rate)3 -7.083e-04  5.197e-02  -0.014    0.989
## season(rate)4  -9.175e-03  5.197e-02  -0.177    0.860
## season(rate)5  -1.004e-02  5.197e-02  -0.193    0.847
## season(rate)6  -8.758e-03  5.197e-02  -0.169    0.866
## season(rate)7  -9.754e-03  5.197e-02  -0.188    0.851
## season(rate)8  -1.027e-02  5.197e-02  -0.198    0.843
## season(rate)9  -1.090e-02  5.197e-02  -0.210    0.834
## season(rate)10 -1.136e-02  5.197e-02  -0.219    0.827
## season(rate)11 -8.639e-03  5.197e-02  -0.166    0.868
## season(rate)12 -1.258e-02  5.197e-02  -0.242    0.809
## season(rate)13 -1.081e-02  5.197e-02  -0.208    0.835
## season(rate)14 -1.060e-02  5.197e-02  -0.204    0.838
## season(rate)15 -9.180e-03  5.197e-02  -0.177    0.860
## season(rate)16 -9.054e-03  5.197e-02  -0.174    0.862
## season(rate)17 -6.926e-03  5.197e-02  -0.133    0.894
## season(rate)18 -8.191e-03  5.197e-02  -0.158    0.875
## season(rate)19 -5.555e-03  5.197e-02  -0.107    0.915
## season(rate)20 -3.507e-03  5.197e-02  -0.067    0.946
## season(rate)21 -5.783e-03  5.197e-02  -0.111    0.911
## season(rate)22 -9.262e-03  5.197e-02  -0.178    0.859
## season(rate)23 -8.072e-03  5.197e-02  -0.155    0.877
## season(rate)24 -1.037e-02  5.197e-02  -0.200    0.842
## season(rate)25 -1.319e-02  5.197e-02  -0.254    0.800
## season(rate)26 -1.098e-02  5.197e-02  -0.211    0.833
## season(rate)27 -8.312e-03  5.197e-02  -0.160    0.873
## season(rate)28 -1.010e-02  5.197e-02  -0.194    0.846
## season(rate)29 -7.214e-03  5.197e-02  -0.139    0.890
## season(rate)30 -9.148e-03  5.197e-02  -0.176    0.860
## season(rate)31  8.963e-04  5.265e-02   0.017    0.986
## season(rate)32 -1.290e-03  5.265e-02  -0.025    0.980
## season(rate)33  4.955e-03  5.265e-02   0.094    0.925
## season(rate)34  1.813e-03  5.265e-02   0.034    0.973
## season(rate)35 -4.945e-03  5.265e-02  -0.094    0.925
## season(rate)36 -5.124e-03  5.265e-02  -0.097    0.922
## season(rate)37 -2.077e-03  5.265e-02  -0.039    0.969
## season(rate)38 -5.029e-03  5.265e-02  -0.096    0.924
## season(rate)39 -4.919e-03  5.265e-02  -0.093    0.926
## season(rate)40  4.973e-03  5.265e-02   0.094    0.925
## season(rate)41  3.710e-03  5.265e-02   0.070    0.944
## season(rate)42 -1.926e-03  5.265e-02  -0.037    0.971
## season(rate)43 -2.631e-03  5.265e-02  -0.050    0.960
## season(rate)44  8.469e-05  5.265e-02   0.002    0.999
## season(rate)45  8.416e-04  5.265e-02   0.016    0.987
## season(rate)46 -6.382e-04  5.265e-02  -0.012    0.990
## season(rate)47 -5.232e-03  5.265e-02  -0.099    0.921
## season(rate)48 -1.002e-02  5.265e-02  -0.190    0.849
## season(rate)49 -4.416e-03  5.265e-02  -0.084    0.933
## season(rate)50 -5.025e-03  5.265e-02  -0.095    0.924
## season(rate)51 -6.753e-04  5.265e-02  -0.013    0.990
## season(rate)52  3.349e-03  5.265e-02   0.064    0.949
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1643 on 965 degrees of freedom
## Multiple R-squared:  0.09115,    Adjusted R-squared:  0.04218
## F-statistic: 1.861 on 52 and 965 DF,  p-value: 0.0002804
```

```
summary(lm_fit_square)
```

```
##
## Time series regression with "ts" data:
## Start = 2000(1), End = 2019(30)
##
## Call:
## dynlm(formula = rate ~ x1_square + x2_square + season(rate))
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.211931 -0.066917 -0.002785  0.062843  0.246424
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)     0.8368860  0.0214713  38.977   <2e-16 ***
## x1_square       1.9761760  0.0389215  50.773   <2e-16 ***
## x2_square      -1.8031330  0.0377232 -47.799   <2e-16 ***
## season(rate)2   0.0009811  0.0283243   0.035    0.972
## season(rate)3  -0.0008022  0.0283243  -0.028    0.977
## season(rate)4  -0.0093104  0.0283243  -0.329    0.742
## season(rate)5  -0.0102119  0.0283243  -0.361    0.719
## season(rate)6  -0.0089664  0.0283243  -0.317    0.752
## season(rate)7  -0.0099945  0.0283243  -0.353    0.724
## season(rate)8  -0.0105335  0.0283243  -0.372    0.710
## season(rate)9  -0.0111946  0.0283244  -0.395    0.693
## season(rate)10 -0.0116752  0.0283244  -0.412    0.680
## season(rate)11 -0.0089694  0.0283244  -0.317    0.752
## season(rate)12 -0.0129212  0.0283244  -0.456    0.648
## season(rate)13 -0.0111654  0.0283245  -0.394    0.694
## season(rate)14 -0.0109591  0.0283245  -0.387    0.699
## season(rate)15 -0.0095453  0.0283246  -0.337    0.736
## season(rate)16 -0.0094190  0.0283246  -0.333    0.740
## season(rate)17 -0.0072882  0.0283247  -0.257    0.797
## season(rate)18 -0.0085460  0.0283247  -0.302    0.763
## season(rate)19 -0.0058993  0.0283248  -0.208    0.835
## season(rate)20 -0.0038381  0.0283248  -0.136    0.892
## season(rate)21 -0.0060964  0.0283249  -0.215    0.830
## season(rate)22 -0.0095547  0.0283250  -0.337    0.736
## season(rate)23 -0.0083396  0.0283250  -0.294    0.768
## season(rate)24 -0.0106110  0.0283251  -0.375    0.708
## season(rate)25 -0.0134011  0.0283252  -0.473    0.636
## season(rate)26 -0.0111498  0.0283253  -0.394    0.694
## season(rate)27 -0.0084478  0.0283253  -0.298    0.766
## season(rate)28 -0.0101962  0.0283254  -0.360    0.719
## season(rate)29 -0.0072630  0.0283255  -0.256    0.798
## season(rate)30 -0.0091482  0.0283256  -0.323    0.747
## season(rate)31 -0.0145682  0.0286964  -0.508    0.612
## season(rate)32 -0.0167895  0.0286964  -0.585    0.559
## season(rate)33 -0.0105757  0.0286964  -0.369    0.713
## season(rate)34 -0.0137458  0.0286964  -0.479    0.632
## season(rate)35 -0.0205282  0.0286965  -0.715    0.475
## season(rate)36 -0.0207276  0.0286965  -0.722    0.470
## season(rate)37 -0.0176983  0.0286965  -0.617    0.538
## season(rate)38 -0.0206639  0.0286966  -0.720    0.472
## season(rate)39 -0.0205648  0.0286966  -0.717    0.474
## season(rate)40 -0.0106795  0.0286967  -0.372    0.710
## season(rate)41 -0.0119455  0.0286967  -0.416    0.677
## season(rate)42 -0.0175820  0.0286967  -0.613    0.540
## season(rate)43 -0.0182831  0.0286968  -0.637    0.524
## season(rate)44 -0.0155608  0.0286968  -0.542    0.588
## season(rate)45 -0.0147934  0.0286969  -0.516    0.606
## season(rate)46 -0.0162593  0.0286970  -0.567    0.571
## season(rate)47 -0.0208352  0.0286970  -0.726    0.468
## season(rate)48 -0.0256074  0.0286971  -0.892    0.372
## season(rate)49 -0.0199740  0.0286971  -0.696    0.487
## season(rate)50 -0.0205552  0.0286972  -0.716    0.474
## season(rate)51 -0.0161746  0.0286973  -0.564    0.573
## season(rate)52 -0.0121160  0.0286973  -0.422    0.673
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08957 on 964 degrees of freedom
## Multiple R-squared:  0.7303, Adjusted R-squared:  0.7155
## F-statistic: 49.26 on 53 and 964 DF,  p-value: < 2.2e-16
```

```r
##### compare two models
plot(rate, lwd=2, col="black",ylab="Rate")
lines(fitted(lm_fit_par), lwd=2, col="purple")
lines(fitted(lm_fit_square), lwd=2, col="red")
legend(x= "topleft",legend=c("Original" ,"Non-Polynomial","Polynomial"),lty=1, col=c("black","purple","
```



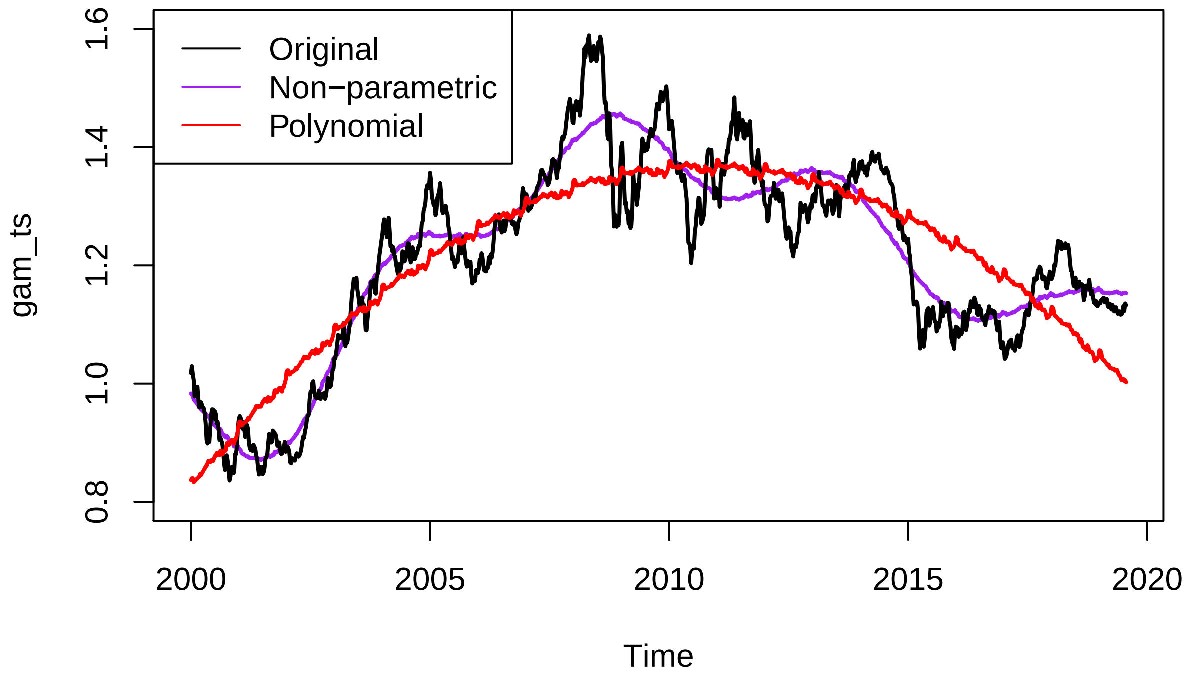
- Non-parametric model

```r
# For sake of simplicity, only use Categorical Regression (ANOVA) seasonality modelling.
week = dynlm(rate~season(rate)+0)
summary(week)
```

```
##
## Time series regression with "ts" data:
## Start = 2000(1), End = 2019(30)
##
## Call:
## dynlm(formula = rate ~ season(rate) + 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.37885 -0.09772  0.01871  0.12186  0.37956
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## season(rate)1   1.21482    0.03851   31.54   <2e-16 ***
## season(rate)2   1.21602    0.03851   31.57   <2e-16 ***
## season(rate)3   1.21445    0.03851   31.54   <2e-16 ***
## season(rate)4   1.20616    0.03851   31.32   <2e-16 ***
## season(rate)5   1.20546    0.03851   31.30   <2e-16 ***
## season(rate)6   1.20692    0.03851   31.34   <2e-16 ***
## season(rate)7   1.20609    0.03851   31.32   <2e-16 ***
## season(rate)8   1.20575    0.03851   31.31   <2e-16 ***
## season(rate)9   1.20529    0.03851   31.30   <2e-16 ***
## season(rate)10  1.20500    0.03851   31.29   <2e-16 ***
## season(rate)11  1.20789    0.03851   31.36   <2e-16 ***
## season(rate)12  1.20413    0.03851   31.27   <2e-16 ***
## season(rate)13  1.20607    0.03851   31.32   <2e-16 ***
## season(rate)14  1.20645    0.03851   31.33   <2e-16 ***
## season(rate)15  1.20804    0.03851   31.37   <2e-16 ***
## season(rate)16  1.20834    0.03851   31.38   <2e-16 ***
## season(rate)17  1.21064    0.03851   31.44   <2e-16 ***
## season(rate)18  1.20954    0.03851   31.41   <2e-16 ***
## season(rate)19  1.21235    0.03851   31.48   <2e-16 ***
## season(rate)20  1.21457    0.03851   31.54   <2e-16 ***
## season(rate)21  1.21247    0.03851   31.48   <2e-16 ***
## season(rate)22  1.20916    0.03851   31.40   <2e-16 ***
## season(rate)23  1.21052    0.03851   31.43   <2e-16 ***
## season(rate)24  1.20839    0.03851   31.38   <2e-16 ***
## season(rate)25  1.20574    0.03851   31.31   <2e-16 ***
## season(rate)26  1.20813    0.03851   31.37   <2e-16 ***
## season(rate)27  1.21097    0.03851   31.44   <2e-16 ***
## season(rate)28  1.20935    0.03851   31.40   <2e-16 ***
## season(rate)29  1.21241    0.03851   31.48   <2e-16 ***
## season(rate)30  1.21065    0.03851   31.44   <2e-16 ***
## season(rate)31  1.21640    0.03951   30.79   <2e-16 ***
## season(rate)32  1.21438    0.03951   30.73   <2e-16 ***
## season(rate)33  1.22080    0.03951   30.90   <2e-16 ***
## season(rate)34  1.21783    0.03951   30.82   <2e-16 ***
## season(rate)35  1.21124    0.03951   30.66   <2e-16 ***
## season(rate)36  1.21124    0.03951   30.66   <2e-16 ***
## season(rate)37  1.21446    0.03951   30.74   <2e-16 ***
## season(rate)38  1.21168    0.03951   30.67   <2e-16 ***
## season(rate)39  1.21196    0.03951   30.67   <2e-16 ***
## season(rate)40  1.22202    0.03951   30.93   <2e-16 ***
## season(rate)41  1.22093    0.03951   30.90   <2e-16 ***
## season(rate)42  1.21547    0.03951   30.76   <2e-16 ***
## season(rate)43  1.21493    0.03951   30.75   <2e-16 ***
## season(rate)44  1.21782    0.03951   30.82   <2e-16 ***
## season(rate)45  1.21875    0.03951   30.84   <2e-16 ***
## season(rate)46  1.21744    0.03951   30.81   <2e-16 ***
## season(rate)47  1.21302    0.03951   30.70   <2e-16 ***
## season(rate)48  1.20840    0.03951   30.58   <2e-16 ***
## season(rate)49  1.21418    0.03951   30.73   <2e-16 ***
## season(rate)50  1.21374    0.03951   30.72   <2e-16 ***
## season(rate)51  1.21826    0.03951   30.83   <2e-16 ***
## season(rate)52  1.22246    0.03951   30.94   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1722 on 966 degrees of freedom
## Multiple R-squared:  0.9812, Adjusted R-squared:  0.9802
## F-statistic: 969.2 on 52 and 966 DF,  p-value: < 2.2e-16
```

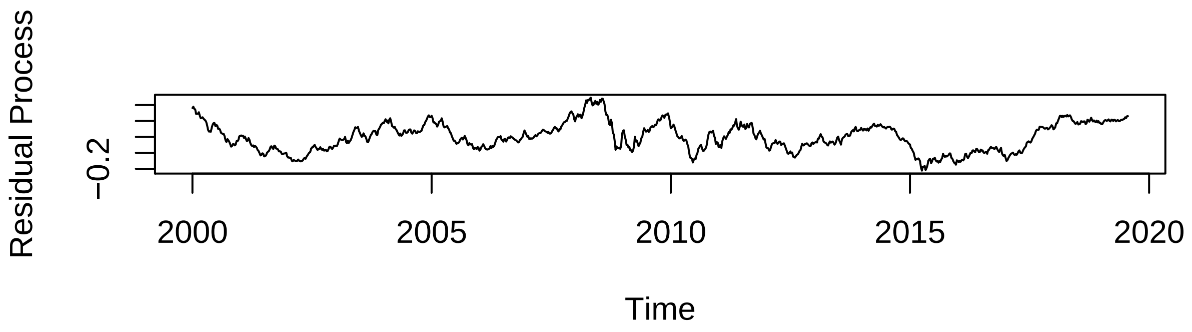
```
week = fitted(week)
```

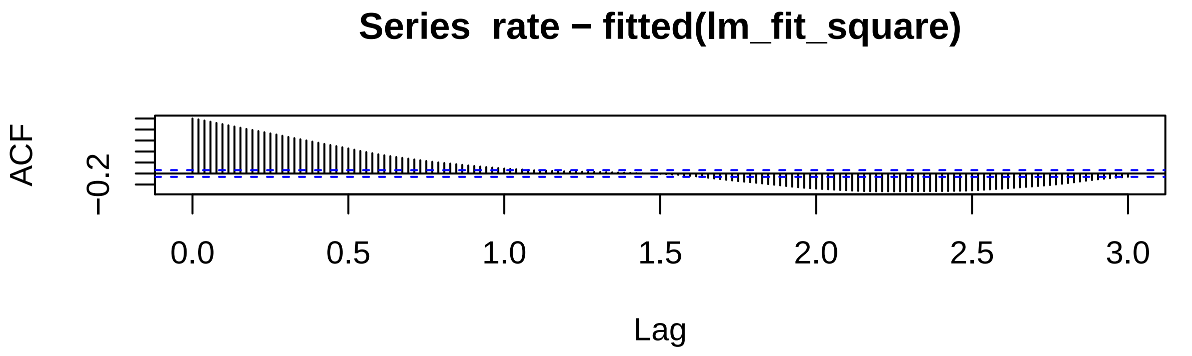
```
## Fit a non-parametric model for trend and linear model for seasonality
gam.fit = gam(rate~s(poly_time.pts)+week)
gam_ts = ts(fitted(gam.fit),start=c(2000,1),freq=52)

plot(gam_ts, lwd=2, ylim= c(0.8, 1.6), col="purple")
lines(rate, lwd=2, col="black",ylab="Rate")
lines(fitted(lm_fit_square), lwd=2, col="red")
legend(x= "topleft",legend=c("Original" ,"Non-parametric","Polynomial"),lty=1, col=c("black","purple","
```



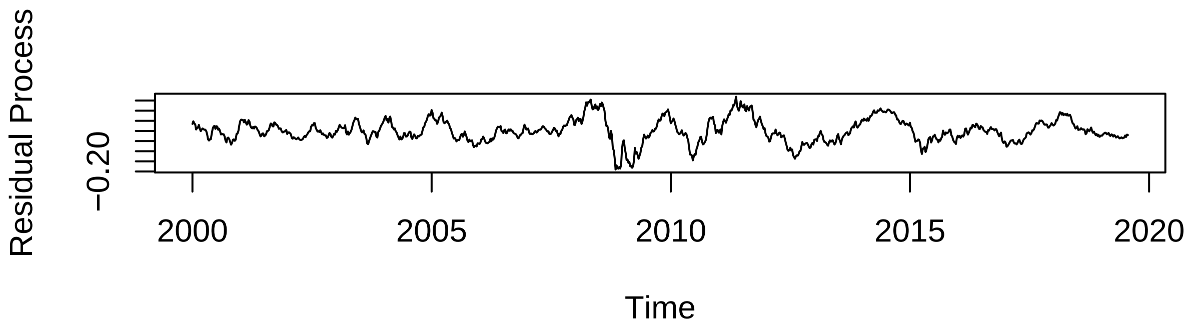
- Construct and plot the residuals with respect to time and ACF of residuals.

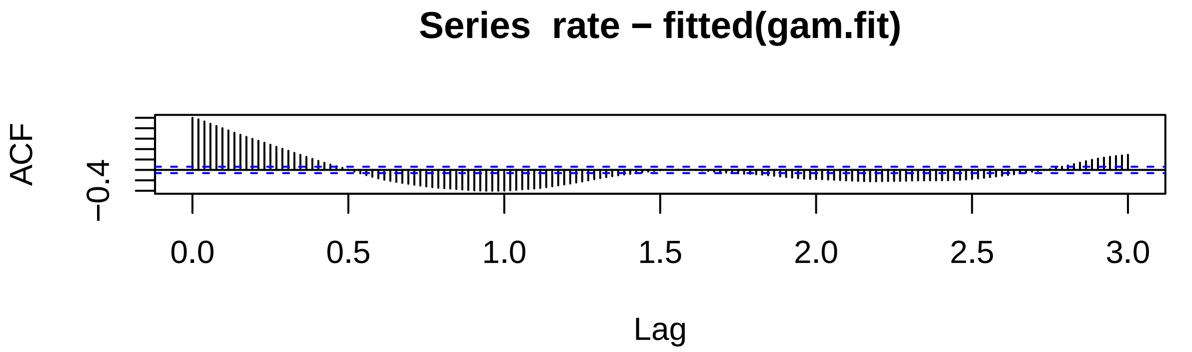
```
par(mfrow=c(2,1))
dif.fit.lm = ts((rate-fitted(lm_fit_square)),start=c(2000,1),freq=52)
plot(dif.fit.lm,ylab="Residual Process")

acf(rate-fitted(lm_fit_square), lag=52 * 3)
```





```
par(mfrow=c(2,1))
res.fit.gam = ts((rate-fitted(gam.fit)),start=c(2000,1),freq=52)
ts.plot(res.fit.gam,ylab="Residual Process")

acf(rate-fitted(gam.fit), lag=52 * 3)
```





- Comment on how the two models fit and on the appropriateness of the stationarity assumption of the residuals.

For model fit, I think the gerneralized addivitive model could capture the shape of time series better but based on two ACF plots, we could see they both violate constant mean assumption becasue we could see decreasing and increasing pattern in ACF plots.

Question 2c: Trend-Seasonality Estimation with Differenced Data

Now using the differenced time series data, construct the same type of models as you did above.

Overlay the fitted values on the original time series.Construct and plot the residuals with respect to time and ACF of residuals. Comment on the two models fit and on the appropriateness of the stationarity assumption of the residuals. Additionally, comment if models built with original or differenced data appear to differ in quality of fit; which (if any) is better?

Hint:

When TS data is differenced, the resulting dataset begins observations at the second time point of the original series. To ensure fitted values line up properly, convert them to time series with the following function:

ts(fit, start=c(2000,2),freq=52)

Where "fit" represents the appropriate fitted values. This functions communicates that the time series is broken down into 52 equal sized chunks (weeks) each year (freq=52), and that this particular series begins with the second chunk of the year 2000 (start=c(2000,2)).

```
#Convert to TS data in proper frame
rate <- ts(data2,start=c(2000,1),freq=52)
#Generate differenced data
rate.dif <- diff(rate)
```

```
# plot(rate.dif)
week_dif = dynlm(rate.dif~season(rate.dif)+0)
summary(week_dif)
```

```
##
## Time series regression with "ts" data:
## Start = 2000(2), End = 2019(30)
##
## Call:
## dynlm(formula = rate.dif ~ season(rate.dif) + 0)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.059575 -0.008532  0.000922  0.008546  0.077364
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## season(rate.dif)1   2.724e-03  3.164e-03   0.861  0.38948
## season(rate.dif)2   1.202e-03  3.084e-03   0.390  0.69696
## season(rate.dif)3  -1.566e-03  3.084e-03  -0.508  0.61169
## season(rate.dif)4  -8.295e-03  3.084e-03  -2.689  0.00728 **
## season(rate.dif)5  -6.915e-04  3.084e-03  -0.224  0.82265
## season(rate.dif)6   1.452e-03  3.084e-03   0.471  0.63792
## season(rate.dif)7  -8.250e-04  3.084e-03  -0.267  0.78916
## season(rate.dif)8  -3.395e-04  3.084e-03  -0.110  0.91238
## season(rate.dif)9  -4.650e-04  3.084e-03  -0.151  0.88020
## season(rate.dif)10 -2.880e-04  3.084e-03  -0.093  0.92563
## season(rate.dif)11  2.895e-03  3.084e-03   0.939  0.34818
## season(rate.dif)12 -3.766e-03  3.084e-03  -1.221  0.22236
## season(rate.dif)13  1.938e-03  3.084e-03   0.628  0.52993
## season(rate.dif)14  3.850e-04  3.084e-03   0.125  0.90069
## season(rate.dif)15  1.589e-03  3.084e-03   0.515  0.60654
## season(rate.dif)16  2.980e-04  3.084e-03   0.097  0.92305
## season(rate.dif)17  2.299e-03  3.084e-03   0.745  0.45623
## season(rate.dif)18 -1.093e-03  3.084e-03  -0.354  0.72314
## season(rate.dif)19  2.808e-03  3.084e-03   0.910  0.36284
## season(rate.dif)20  2.219e-03  3.084e-03   0.719  0.47204
## season(rate.dif)21 -2.104e-03  3.084e-03  -0.682  0.49531
## season(rate.dif)22 -3.308e-03  3.084e-03  -1.072  0.28383
## season(rate.dif)23  1.362e-03  3.084e-03   0.442  0.65877
## season(rate.dif)24 -2.128e-03  3.084e-03  -0.690  0.49050
## season(rate.dif)25 -2.650e-03  3.084e-03  -0.859  0.39050
## season(rate.dif)26  2.388e-03  3.084e-03   0.774  0.43894
## season(rate.dif)27  2.835e-03  3.084e-03   0.919  0.35816
## season(rate.dif)28 -1.618e-03  3.084e-03  -0.525  0.59988
## season(rate.dif)29  3.060e-03  3.084e-03   0.992  0.32143
## season(rate.dif)30 -1.762e-03  3.084e-03  -0.571  0.56789
## season(rate.dif)31  1.643e-03  3.164e-03   0.519  0.60382
## season(rate.dif)32 -2.015e-03  3.164e-03  -0.637  0.52449
## season(rate.dif)33  6.417e-03  3.164e-03   2.028  0.04286 *
## season(rate.dif)34 -2.971e-03  3.164e-03  -0.939  0.34811
## season(rate.dif)35 -6.586e-03  3.164e-03  -2.081  0.03767 *
## season(rate.dif)36 -6.842e-06  3.164e-03  -0.002  0.99828
## season(rate.dif)37  3.218e-03  3.164e-03   1.017  0.30939
## season(rate.dif)38 -2.780e-03  3.164e-03  -0.879  0.37989
## season(rate.dif)39  2.813e-04  3.164e-03   0.089  0.92918
## season(rate.dif)40  1.006e-02  3.164e-03   3.180  0.00152 **
## season(rate.dif)41 -1.091e-03  3.164e-03  -0.345  0.73040
## season(rate.dif)42 -5.465e-03  3.164e-03  -1.727  0.08451 .
## season(rate.dif)43 -5.329e-04  3.164e-03  -0.168  0.86630
## season(rate.dif)44  2.887e-03  3.164e-03   0.912  0.36181
## season(rate.dif)45  9.287e-04  3.164e-03   0.293  0.76922
## season(rate.dif)46 -1.308e-03  3.164e-03  -0.413  0.67941
## season(rate.dif)47 -4.422e-03  3.164e-03  -1.397  0.16266
## season(rate.dif)48 -4.621e-03  3.164e-03  -1.460  0.14451
## season(rate.dif)49  5.781e-03  3.164e-03   1.827  0.06804 .
## season(rate.dif)50 -4.374e-04  3.164e-03  -0.138  0.89010
## season(rate.dif)51  4.521e-03  3.164e-03   1.429  0.15342
## season(rate.dif)52  4.196e-03  3.164e-03   1.326  0.18521
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01379 on 965 degrees of freedom
## Multiple R-squared:  0.05605,    Adjusted R-squared:  0.005181
## F-statistic: 1.102 on 52 and 965 DF,  p-value: 0.2917
```

```
week_dif = fitted(week_dif)
```

- Parametric Polynomial Regression

```
library(zoo)
library(dynlm)
## Polynomial
diff_time.pts = c(1:length(rate.dif))
diff_time.pts = c(diff_time.pts-min(diff_time.pts))/max(diff_time.pts)
x1_square_dif = diff_time.pts
x2_square_dif = diff_time.pts^2
lm_fit_square_diff = dynlm(rate.dif~x1_square_dif+x2_square_dif+week_dif)
summary(lm_fit_square_diff)
```

```
##
## Time series regression with "ts" data:
## Start = 2000(2), End = 2019(30)
##
## Call:
## dynlm(formula = rate.dif ~ x1_square_dif + x2_square_dif + week_dif)
##
```

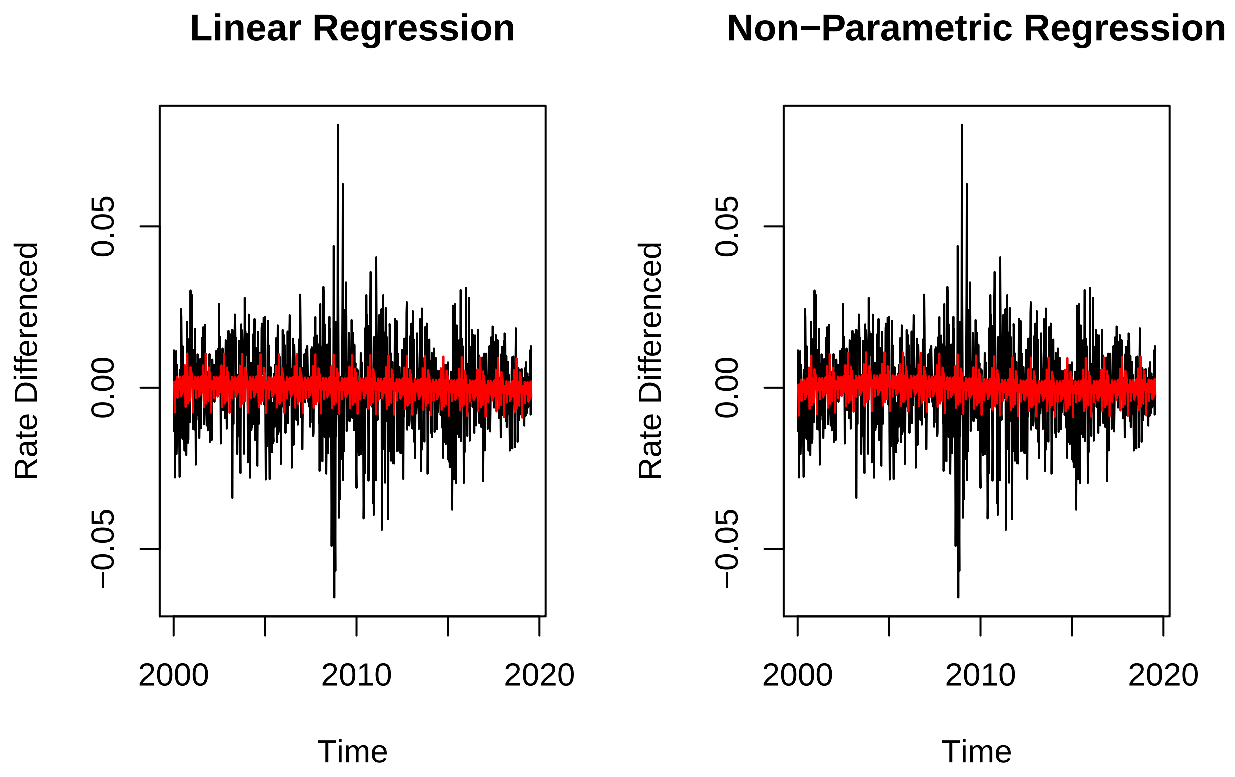
```
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.059783 -0.008401  0.001094  0.008676  0.077168
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.0005720  0.0012632   0.453    0.651
## x1_square_dif -0.0001157  0.0058404  -0.020    0.984
## x2_square_dif -0.0015451  0.0056602  -0.273    0.785
## week_dif       1.0004115  0.1289346   7.759 2.08e-14 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01345 on 1013 degrees of freedom
## Multiple R-squared:  0.05725,    Adjusted R-squared:  0.05446
## F-statistic: 20.51 on 3 and 1013 DF,  p-value: 6.637e-13
```

- Non-parametric model

```
## Fit a non-parametric model for trend and linear model for seasonality
gam.fit.dif = gam(rate.dif~s(x1_square_dif)+week_dif)
gam_ts.dif = ts(fitted(gam.fit.dif),start=c(2000,2),freq=52)
```

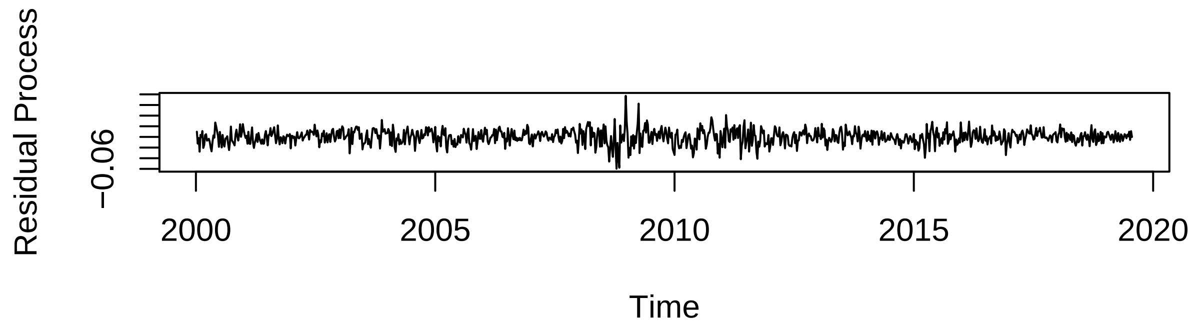
- Overlay the fitted values on the original time series

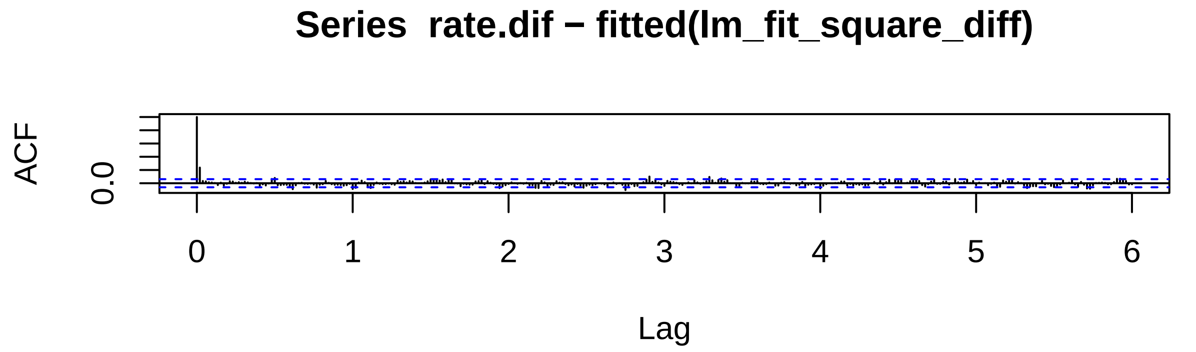
```
par(mfrow=c(1,2))
ts.plot(ts(rate.dif,start=c(2000,2),freq=52),main='Linear Regression',ylab='Rate Differenced',col='blac
lines(fitted(lm_fit_square_diff),col='red')
ts.plot(ts(rate.dif,start=c(2000,2),freq=52),main='Non-Parametric Regression',ylab='Rate Differenced',c
lines(gam_ts.dif,col='red')
```



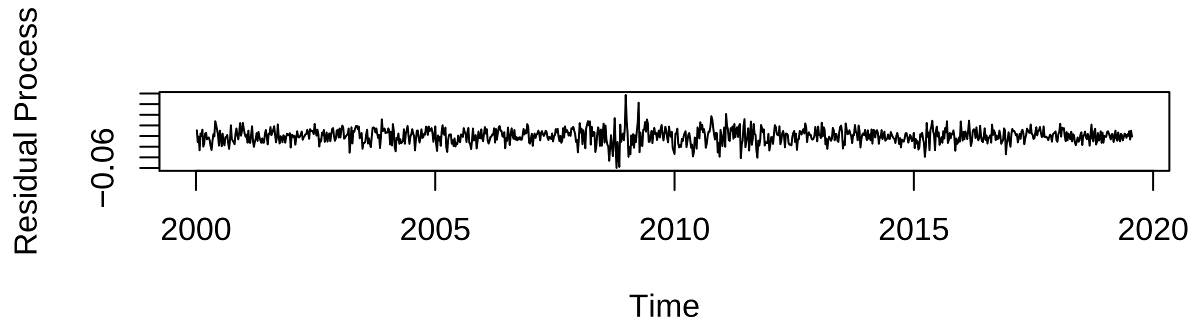
- Construct and plot the residuals with respect to time and ACF of residuals.

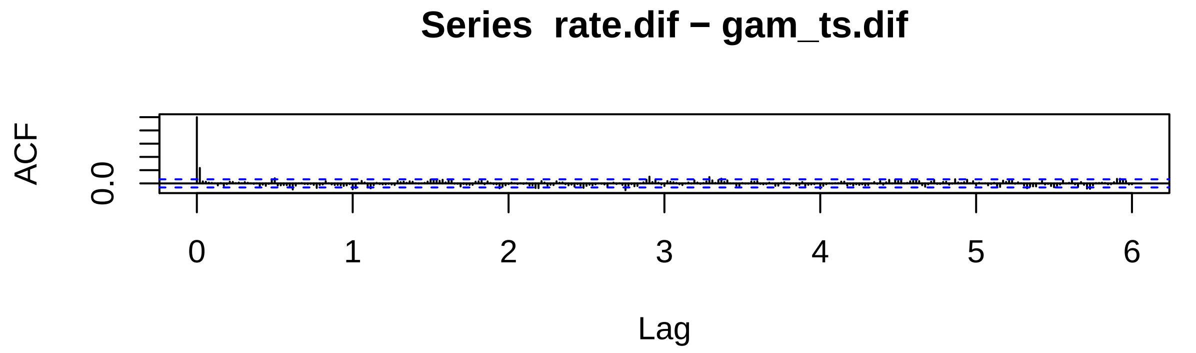
```r
par(mfrow=c(2,1))
res.fit.lm.dif = ts((rate.dif-fitted(lm_fit_square_diff)),start=c(2000,2),freq=52)
plot(res.fit.lm.dif,ylab="Residual Process")

acf(rate.dif-fitted(lm_fit_square_diff), lag=52 * 6)
```





```
par(mfrow=c(2,1))
res.fit.gam.dif = ts((rate.dif-gam_ts.dif),start=c(2000,2),freq=52)
ts.plot(res.fit.gam.dif,ylab="Residual Process")

acf(rate.dif-gam_ts.dif, lag=52 * 6)
```





- Comment on the two models fit and on the appropriateness of the stationarity assumption of the residuals. Additionally, comment if models built with original or differenced data appear to differ in quality of fit; which (if any) is better?

Both models work similarly and don't violate assumptions of stationary process. Clearly, differenced data works better than raw data. We could clearly see there is not trend and seasonality from ACF plots.