PSTAT 174 Final Project

Minghui Zhou 4140190

Cassie Guo 3902640

Tianwei Zhao4315750

Background of topic

In ecomonics data, time series analysis and forecasting can be used widely, including setting monetray and fiscal policies, government budgeting, financial management, risk hedging, and investment strategies. In ecomonic forecasting, we need to first select appropriate models and assess the uncertainty associated with our data forecast. In this project, we will use fundamental time series analysis methods to analysis the trend, seasonality, and period in the ecomony data, which can greatly help policy makers to set future policies.

Data description

The data is collected from https://fred.stlouisfed.org/tags/series, which is a Monthly 10-Year Treasury Constant Maturity Minus 2-Year Treasury Constant Maturity data. The data is calculated as the spread between 10-Year Treasury Constant Maturity (BC_10YEARM) and 2-Year Treasury Constant Maturity (BC_2YEARM). This is an interest rate data, which is closely related in financial marketing and investments. Analysis this data can help us know the relationship between interest rates and economy growths.

Exploratory analysis

Read the data

First let us have a look of the first five rows of the data read in the file *T10Y2YM.csv*:

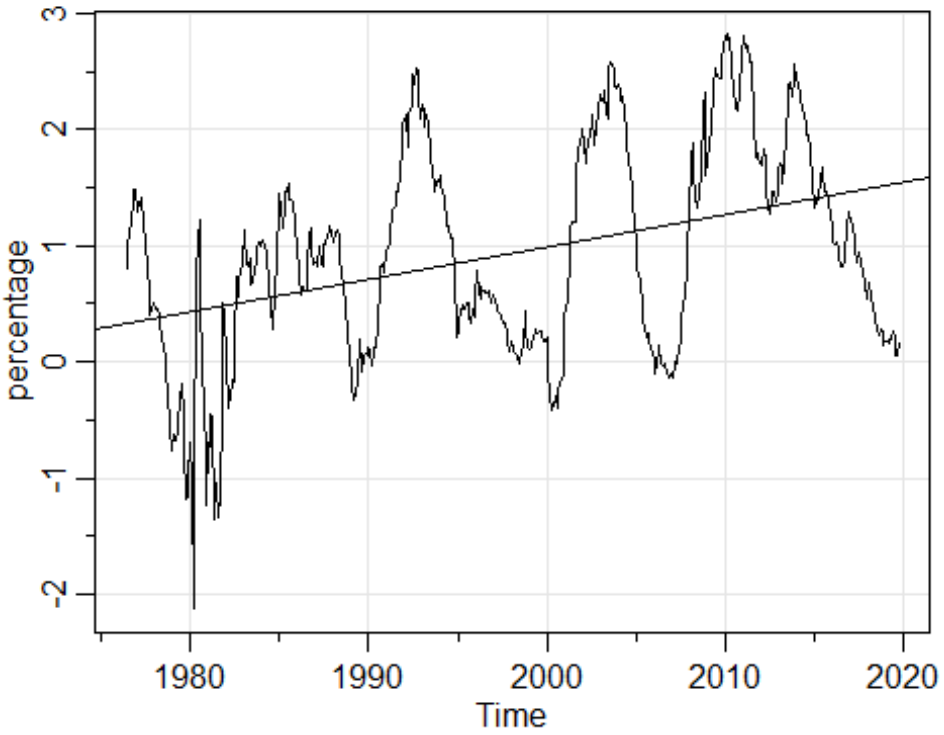
DATE	T10Y2YM
1976-06-01	0.80
1976-07-01	0.98
1976-08-01	1.14
1976-09-01	1.17
1976-10-01	1.43
1976-11-01	1.48

```
## [1] 2019   10
```

We could see the data is collected from June 1976, and the last data is October 2019.

Time series plot

Then we plot the time series, with a single linear regression line to see it's trend.



From the plot we could see the interest rate is increasing for the total trend, though there is some singnificant cycle, from 6 years to 10 years. Furthermore, the growing line indicates the time series is not stationary.

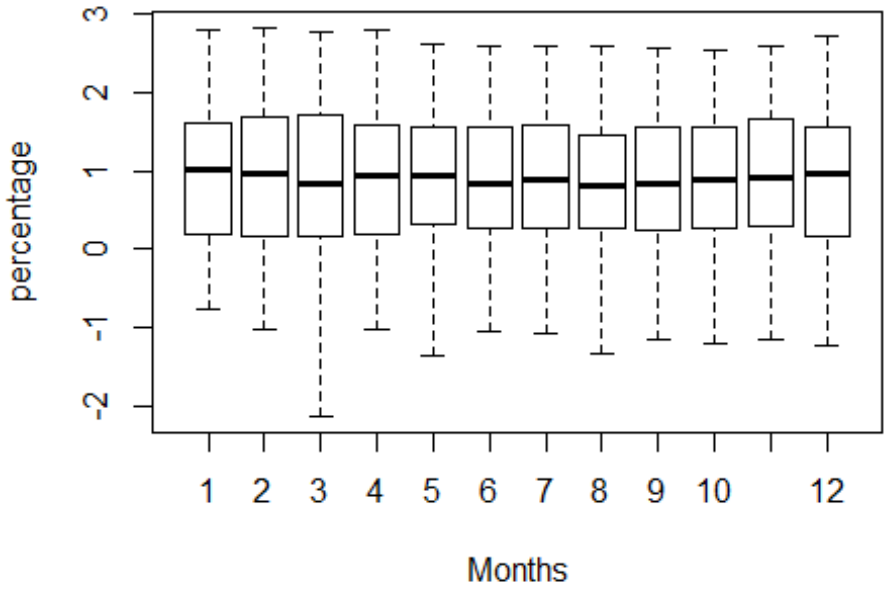
Time series summary

Next we check the summary of the time series data, such as minimum, maximum, quantile, mean and median.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -2.1300  0.2500  0.8900  0.9388  1.6000  2.8300
```

Boxplot on month effects

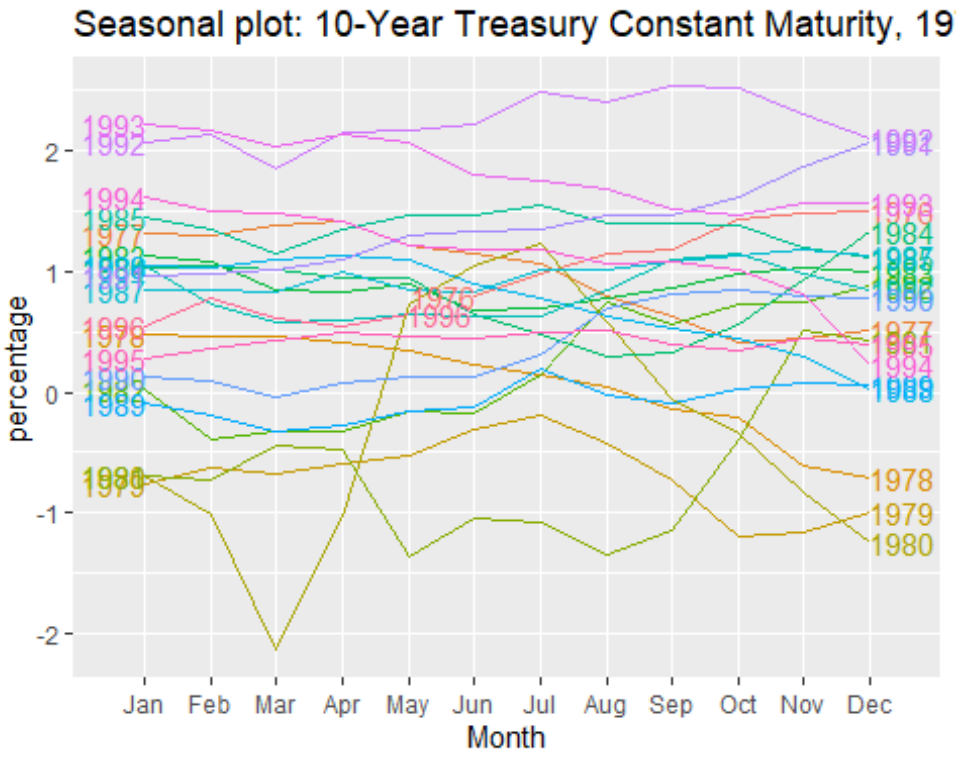
Next we will using boxplot on the month average to check the seasonal effect.

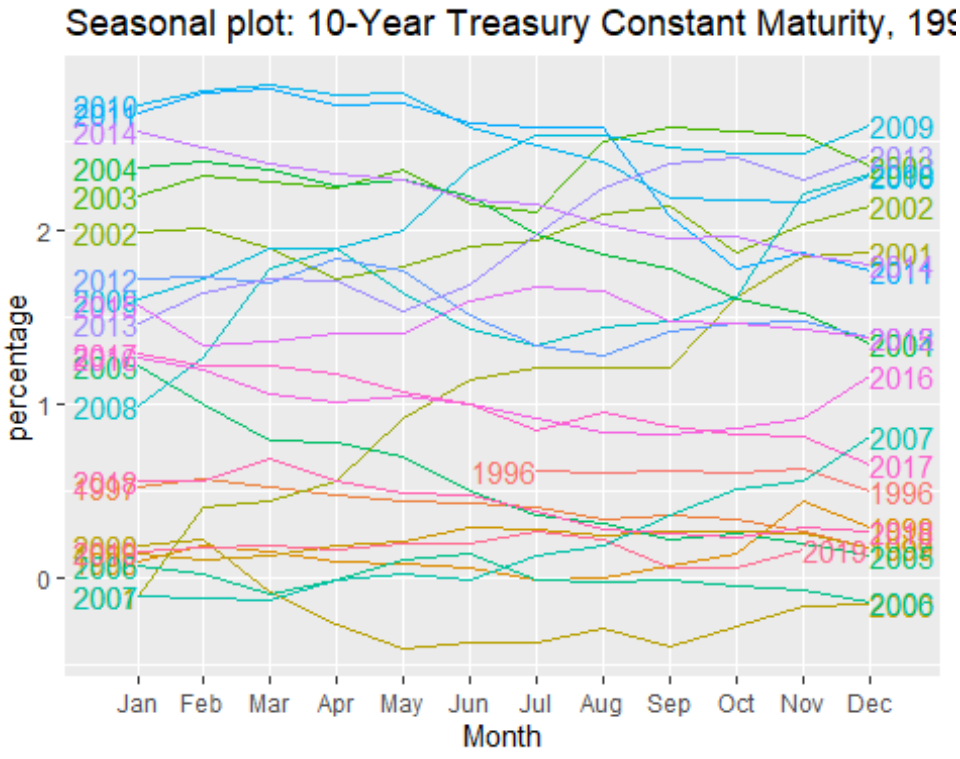


From the boxplot result we could see that there is no seasonal effect on every months. The mean and the quantiles are quite similiar. The variance on each months seems to be quite similiar, too.

Seasonal plot

We could also use the seasonal plot to check seasonality. These plots shows the same data as were shown earlier, with data from each season are overlapped. A seasonal plot help us check seasonal pattern more clearly, and is especially useful in identifying years in which the pattern changes.





From those two seasonal plots we could not find any singnificant seasonal effect.

Modeling goals

Our goal of this project is to use the ARIMA model to build the best model and forcast the future data. We will use 80% of the data as training set, and the last 20% as forcasting set. Furthermore, We will also use spectral analysis methods to analysis cyclic patterns of the time series data. Some hypothesis will be tested, too, such as whether the time series have a seasonal effect. We will also test whether the time series has stationarity or trend stationarity using the unit root test. Last, we will try Kalman filter and smoother model for our data, and compare this model with the ARIMA model.

Seasonality and decomposition

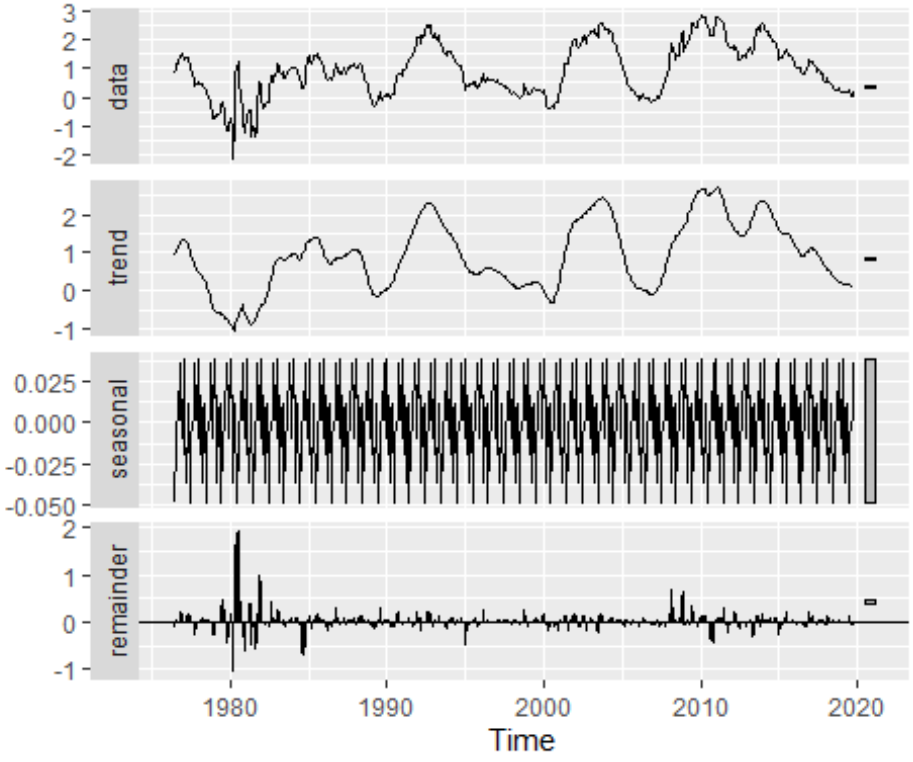
Firstly, we will use decomposition model to check the seasonality in our data. There are two models of decomposition, S_t, T_t, R_t are seasonal component, trend-cycle component, and remainder component respectively. * Additive decomposition

$$y_t = S_t + T_t + R_t$$

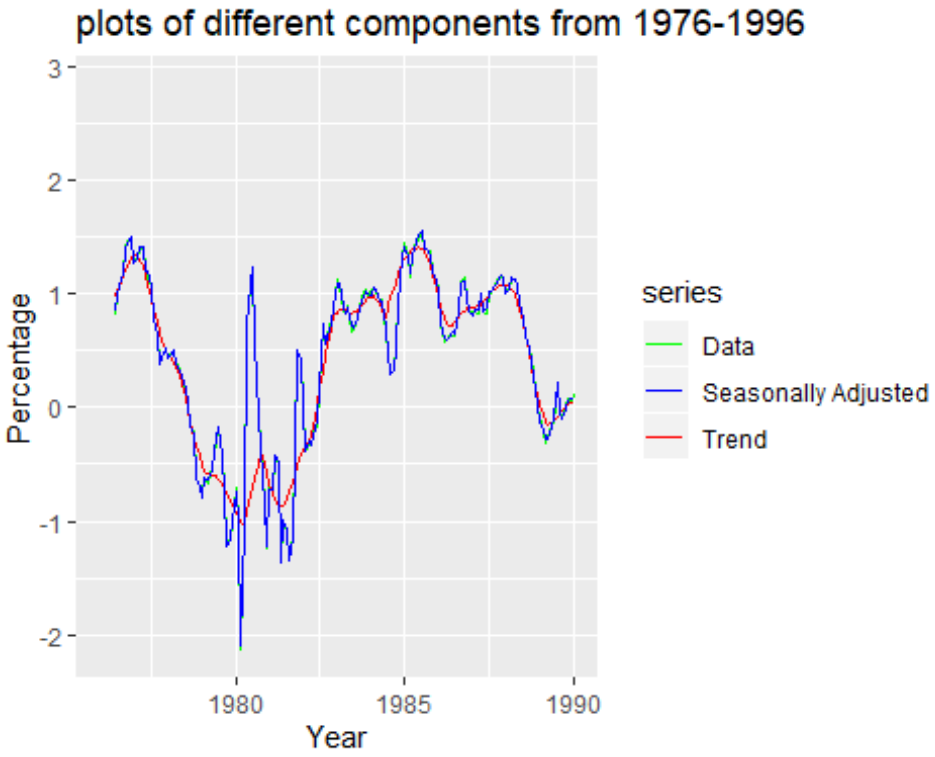
- Multiplicative decomposition

$$y_t = S_t \times T_t \times R_t$$

Here we will try the STL decomposition, which is "Seasonal and Trend decomposition using Loess", while Loess is a method for estimating nonlinear relationships, which will be covered in the next part. In our data set the STL decomposition sugest our data should use the additive decomposition, and the decomposed part are plot below:



We can also see all of the parts in one picture: (1976-1996)



From this plot we could see that the seasonality effect is small compared to the remainder part. Thus we could decide that there is few seasonality in our data.

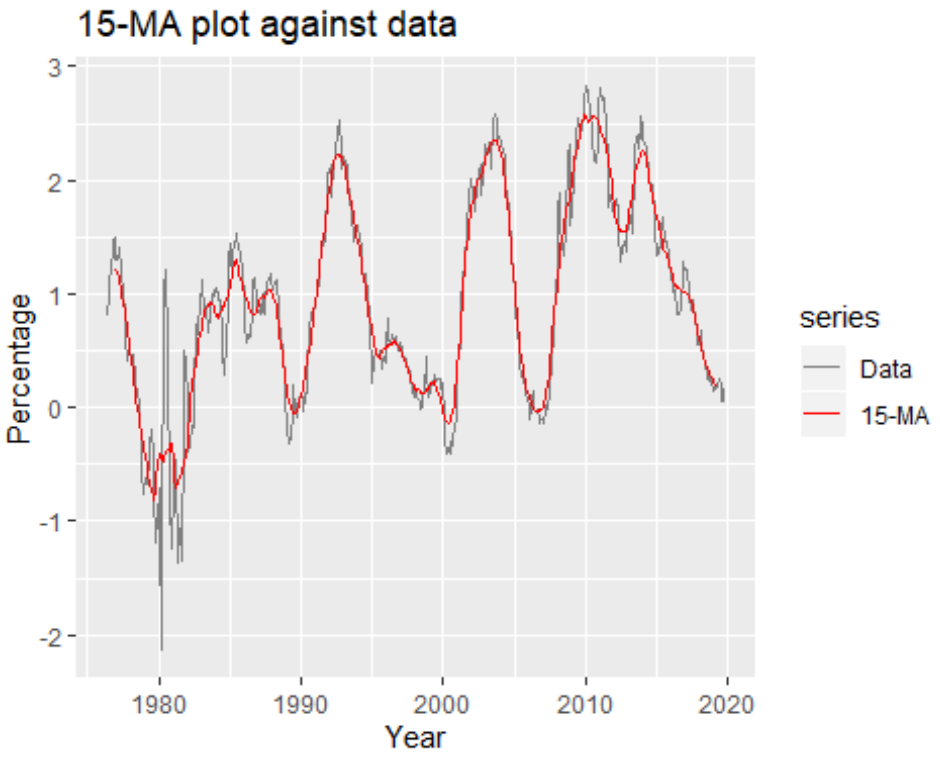
Smoothing

Smoothing methods are used in TSA analysis to get the trend and cycle analysis. In this project we will use two smoothing methods:

- Moving average smoothing A moving average of order m can be written as

$$\hat{T}_t = \frac{1}{m} \sum_{j=-k}^{k} y_{t+j}$$

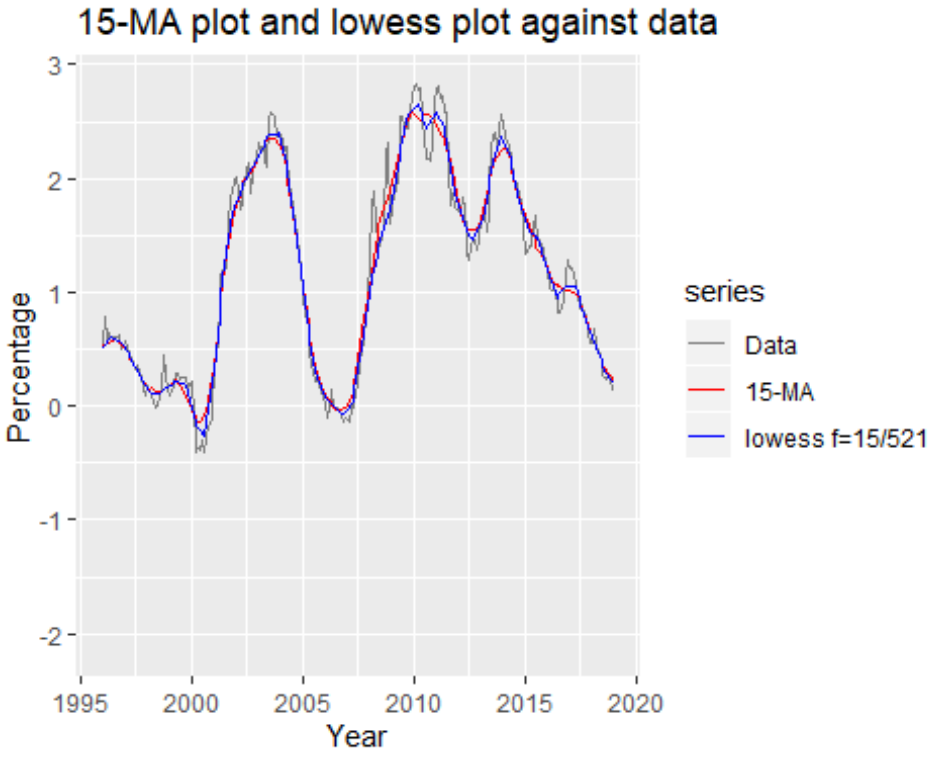
Where $m = 2k + 1$. The estimate of the trend-cycle at time t is acquired with averaging values from the time series within k periods of t. The average estimator can eliminate some of the random noise in the data and give us a smooth trend-cycle component. $m - MA$ means a moving average of order m.



- Lowess Smoothing LOWESS (Locally Weighted Scatterplot Smoothing), sometimes called LOESS (locally weighted smoothing), is another smoothing tool that creates a smooth line through a timeplot to help us reveal trends.

The name local regression comes from the fact that the fitting at point x is weighted toward the data nearest to x. The distance from x is controled using a parameter αin this smoothing. When α is less than 1 it represents the proportion of the data that is considered to be neighbouring x, and the weighting that is used is proportional to $1 - (distance/maximum\ distance)^3)^3$, which is known as tricubic.

Compare the 15-MA plot and the lowess with parameter set as 15/length is:

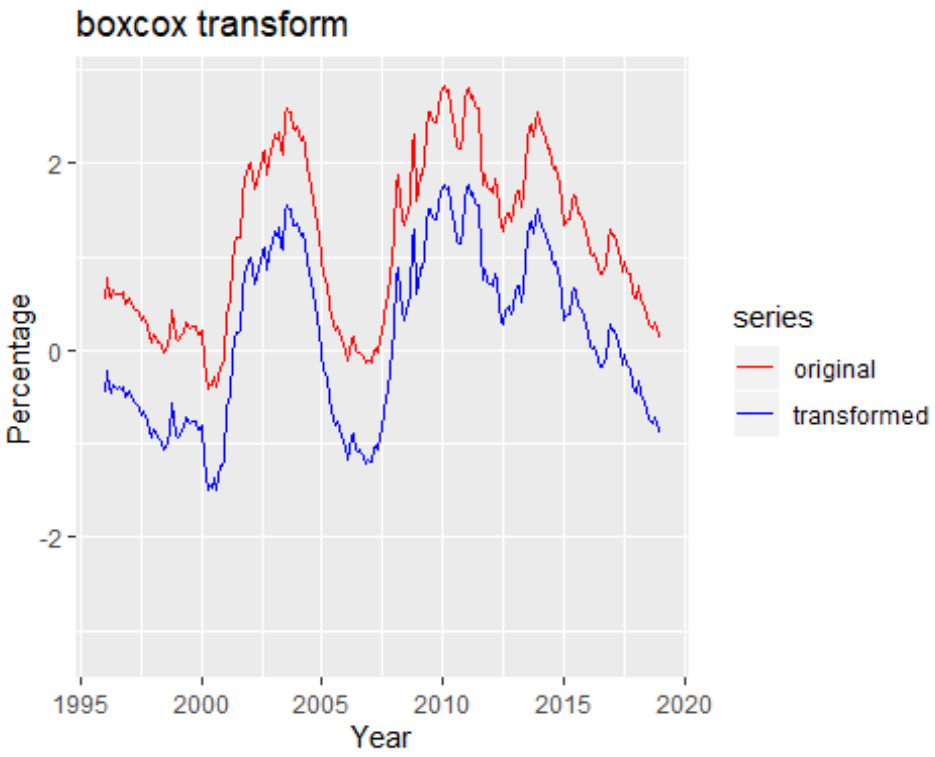


From which we could see those two smoothing methods do quite the same job.

ARIMA models

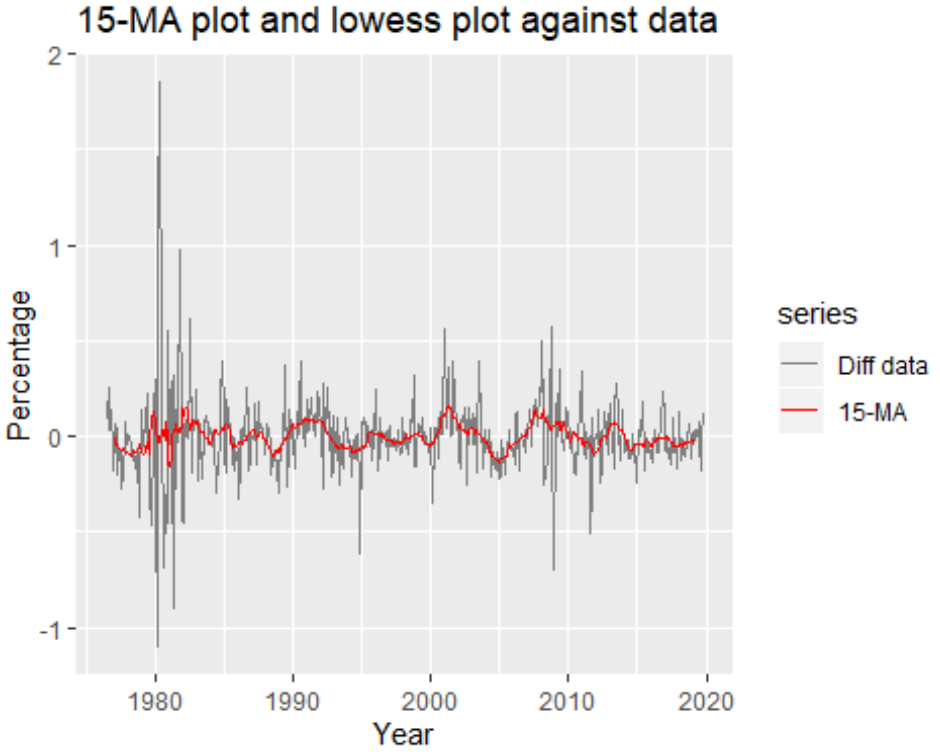
To apply ARIMA models, we first need to find a stationary time series with constant variance. To make the data constant variance, we check the box cox transformation:

```
## [1] 0.9546802
```

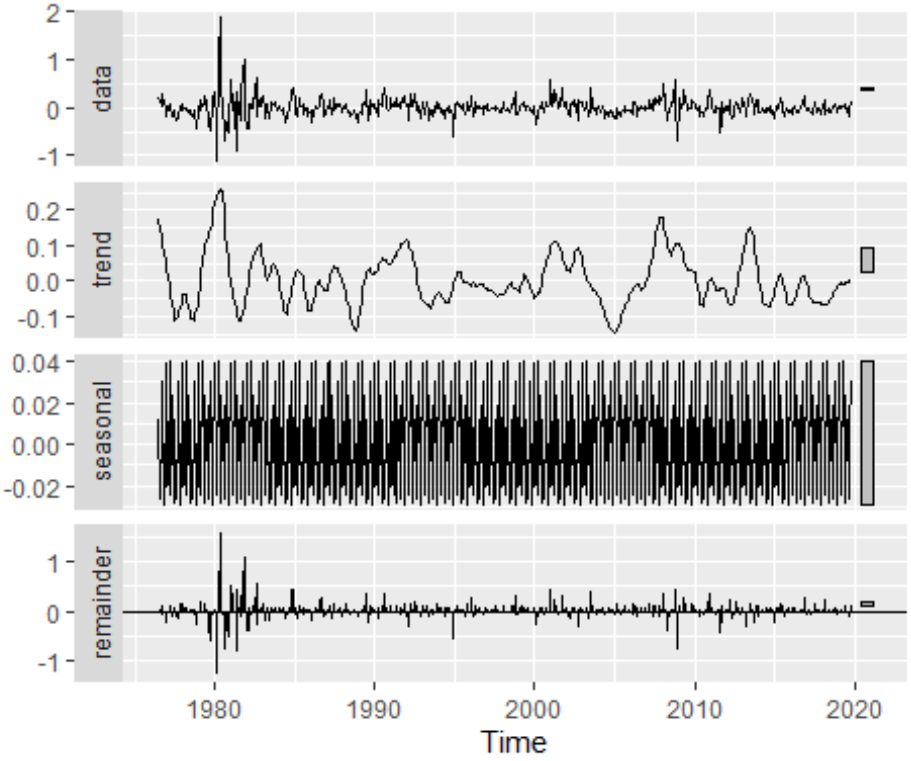


A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary. In our data, there is obvious a trend, thus we need to difference our time series to make it stationary.

Let's difference our transformed time series once and check the 15-MA curve:



Let's check the seasonality:



From the first plot, we could conclude that there is no obvious trend, and on the second plot, we could see the seasonal curve is very small compared to the remainder curve. Thus we could use the ARIMA model on y1.

Unit root tests

One way to determine more objectively whether differencing is required is to use a unit root test. These are statistical hypothesis tests of stationarity that are designed for determining whether differencing is required. In this test, the null hypothesis is that the data are stationary, and we look for evidence that the null hypothesis is false. Consequently, small p-values suggest that differencing is required. The test can be computed using the *ur.kpss()* function from the *urca* package.

```
## [1] "Unit root test for original data"

##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 6 lags.
##
## Value of test-statistic is: 1.4081
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739

## [1] "Unit root test for differenced data"

##
## ########################
## # KPSS Unit Root Test #
## ########################
##
## Test is of type: mu with 6 lags.
##
## Value of test-statistic is: 0.0479
##
## Critical value for a significance level of:
##                 10pct  5pct 2.5pct  1pct
## critical values 0.347 0.463  0.574 0.739
```

From the result we could see the original data need differencing ($p<0.01$), and the differenced data do not need differencing. ($P>0.1$)

Fitting ARIMA models

The non-seasonal ARIMA model, AutoRegressive Integrated Moving Average modek, can be written as:

$$y'_t = c + \phi_1 y'_{t-1} + \cdots + \phi_p y'_{t-p} + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} + \varepsilon_t$$

Where y_t' is the differenced series. The right-hand side include both lagged values of y_t and lagged errors. We call this an ARIMA(p, d, q) model, p is the order of the autoregressive part, d is the degree of first differencing involved, and q is the order of the moving average part. This model must meet the same stationarity and invertibility conditions as moving average model and autoregressive model.

If we define the backshift notation B as:

$$By_t = y_{t-1}$$

Then $y_t' = y_t - y_{t-1} = (1 - B)y_t$, and a dth order difference can be written as $(1 - B)^d y_t$. In other words, the ARIMA model can be written as:

$$(1 - \phi_1 B - \cdots - \phi_p B^p)(y_t' - \mu) = (1 + \theta_1 B + \cdots + \theta_q B^q)\varepsilon_t,$$

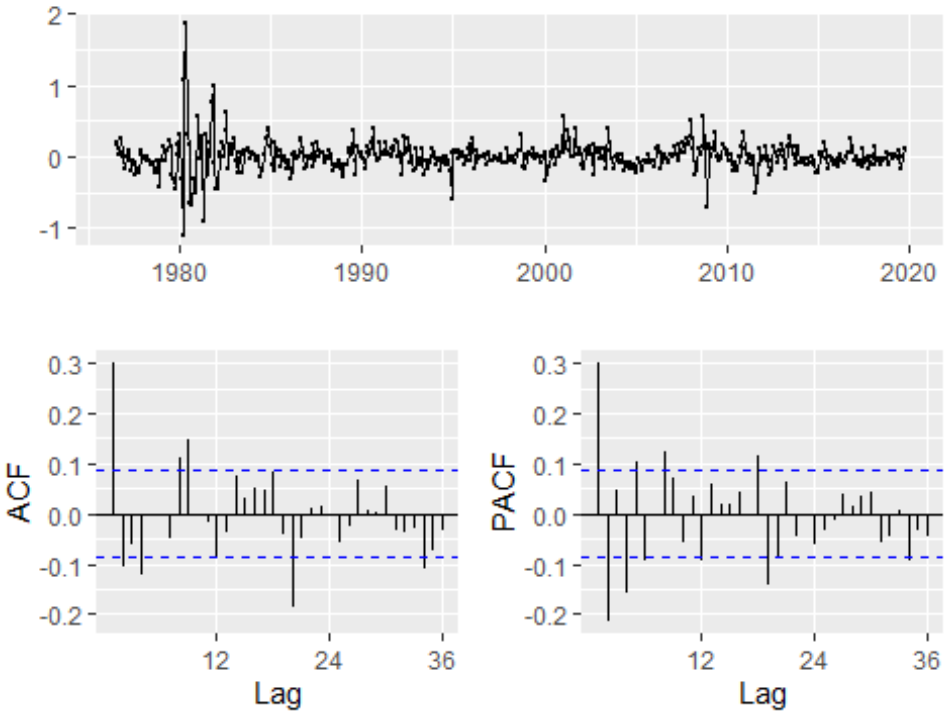
The value of p determines whether the data have cycles. When $p >= 2$ with some additional conditions on the parameters, the forcast can have cyclic properties. For an AR(2) model, cyclic behaviour occurs if $\phi_1^2 + 4\phi_2 < 0$.

The R get the estimation of ARIMA models with MLE estimate. Akaike's Information Criterion can be used in model selection. It can be written as:

$$\text{AIC} = -2\log(L) + 2(p + q + k + 1),$$

And AIC, BIC criterion is only effective in selecting parameter p and q, not d. This is because the differencing changes the data on which the likelihood is computed, making the AIC values between models with different orders of differencing not comparable.

To select an appropriate model, we could check the ACF/PACF plot:

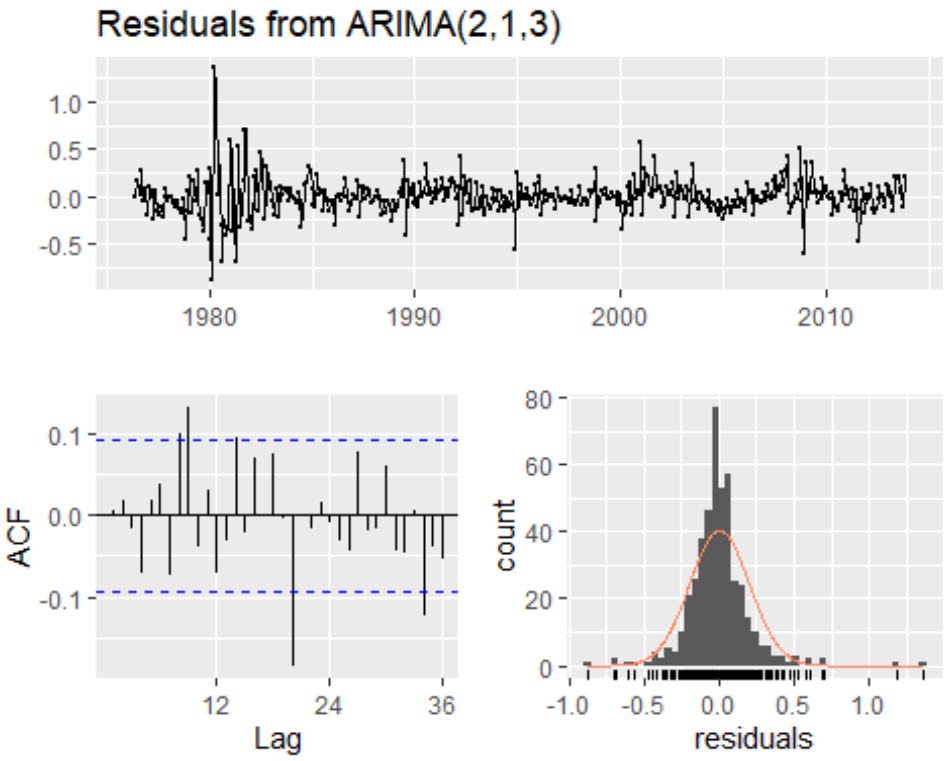


We could find that the ACF plot have 1-2 spike, the PACF plot have 2-3 spikes, and as our series need differ once, we could try ARIMA(2,1,1) or ARIMA(3,1,1) or ARIMA(2,1,2). In the time plot except there is some sudden changes in 1980, and there is no evidence of changing variance.

```
##   aic_2.1.1 aic_3.1.1 aic_2.1.2 aic_2.1.3 aic_2.1.4 aic_3.1.2 aic_3.
1.3
## 1  166.7624  166.0001  165.9821   165.467   174.125  166.8034  169.4
955
```

It shows that we should use an ARIMA(2,1,3) model.

Next we check the residuals:

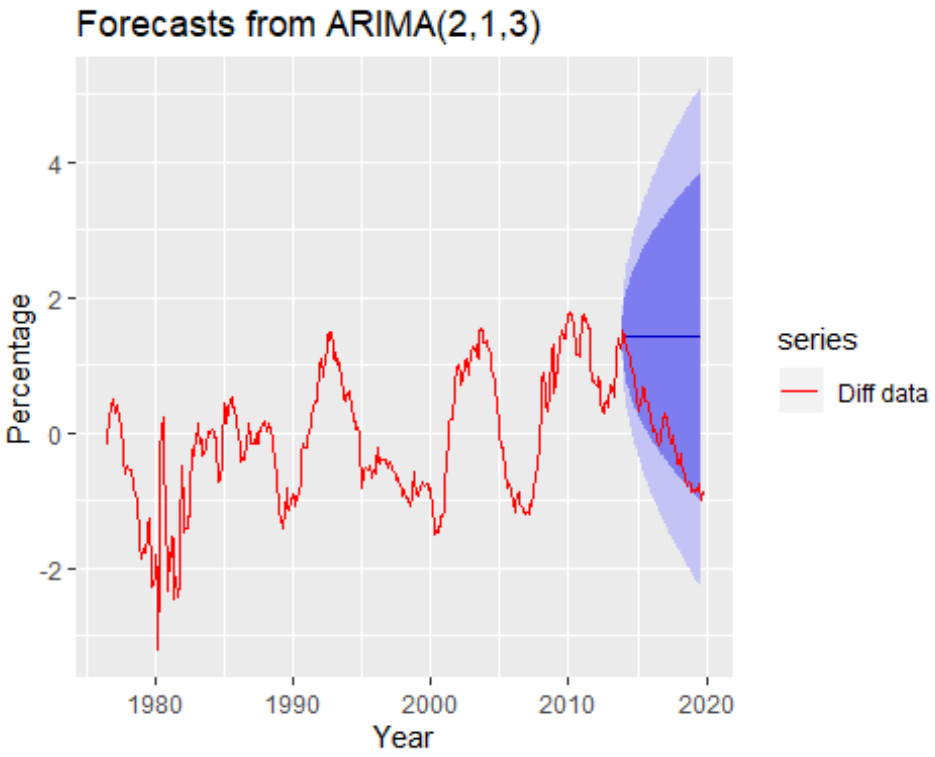



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,3)
## Q* = 47.423, df = 19, p-value = 0.0003106
##
## Model df: 5.   Total lags used: 24
```

The ACF plot of the residuals shows that nearly all autocorrelations are within the threshold limits, thus we could believe the residuals behave quite like white noises. The Ljung-Box test returns a small p-value, but it is much because the sudden change in 1980. So we could accept our model.

Forecast

Point forecasts can be calculated using the following three steps. 1. Expand the ARIMA equation so that y_t is on the left hand side and all other terms are on the right. 2. Rewrite the equation by replacing t with $T + h$ 3. On the right hand side of the equation, replace future observations with their forecasts, future errors with zero, and past errors with the corresponding residuals. Beginning with $h = 1$, repeat these steps until all forecasts have been calculated.



We could find the true data is within the confidence interval of our predicted data.

Spectral analysis

The Fourier transformation has the form:

$$X_t = Acox(\omega t) + Bsin(\omega t) + \varepsilon_t \qquad t = 1,..,n$$

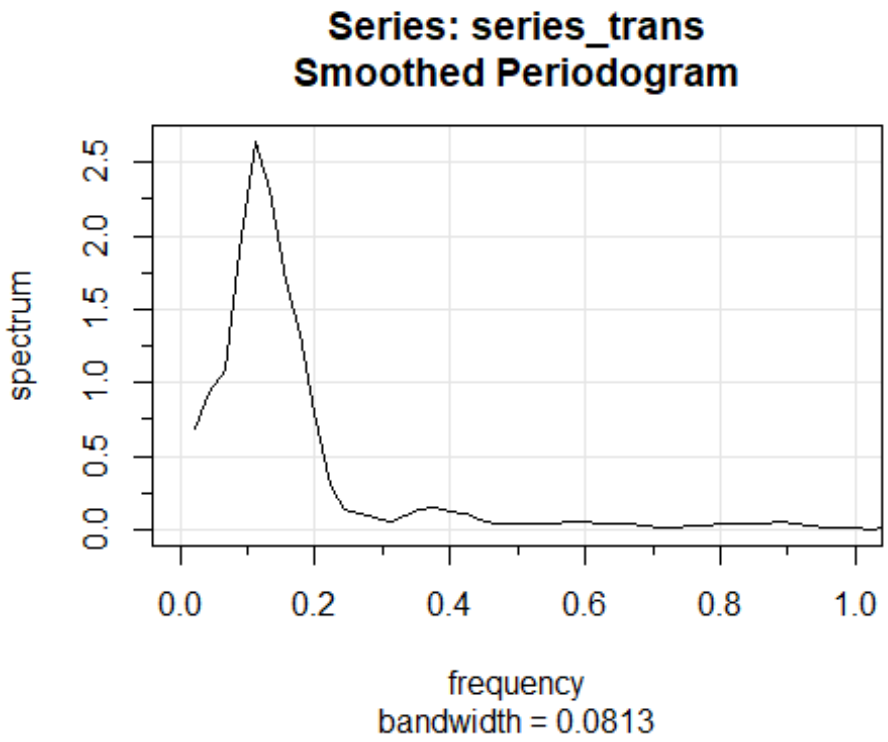
Where ε_t are iid random variable with zeromean and variance σ^2 and ω is unknown. We could estimate the frequency ω by taking the Fourier transform $J_n(\omega) = \frac{1}{\sqrt{n}}\sum_{t=1}^{n} X_t\, e^{it\omega}$ and use this as an estimator of ω, the value which maximized $|J_n(\omega)^2|$. The periodogram is the absolute square of the discrete Fourier Transform (DFT), where

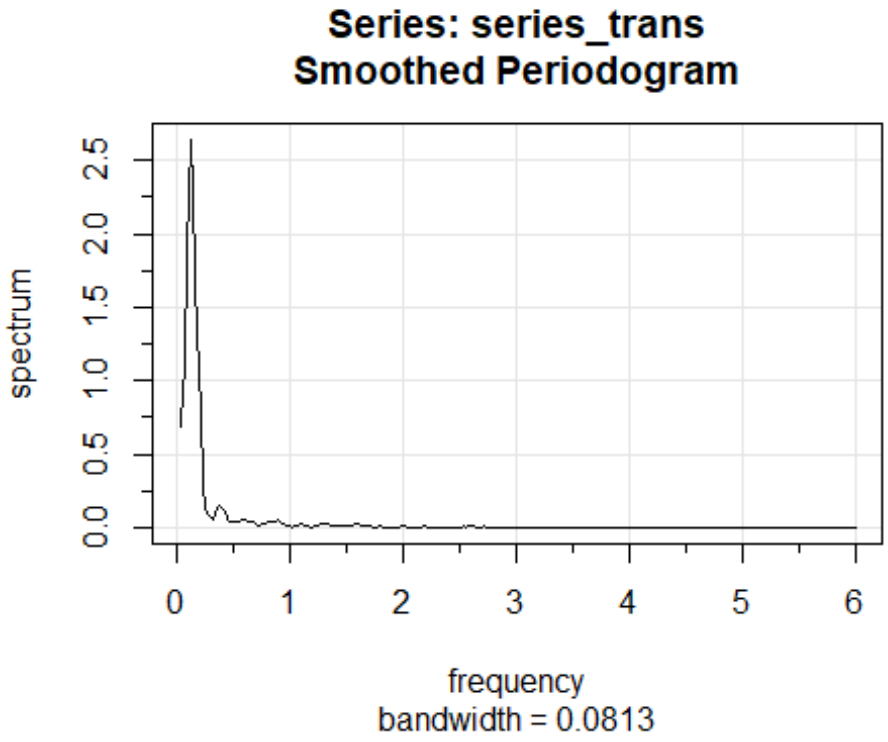
$$J_n(\omega) = \frac{1}{\sqrt{n}}\sum_{t=1}^{n} X_t\, e^{it\omega}$$

Spectral density is defined as

$$f(\omega) = \frac{1}{2\pi}\sum_{r=-\infty}^{\infty} c\,(r)exp(2\pi ir)$$

We will perform a periodogram analysis identifying the predominant periods and obtaining confidence intervals for the identified periods.



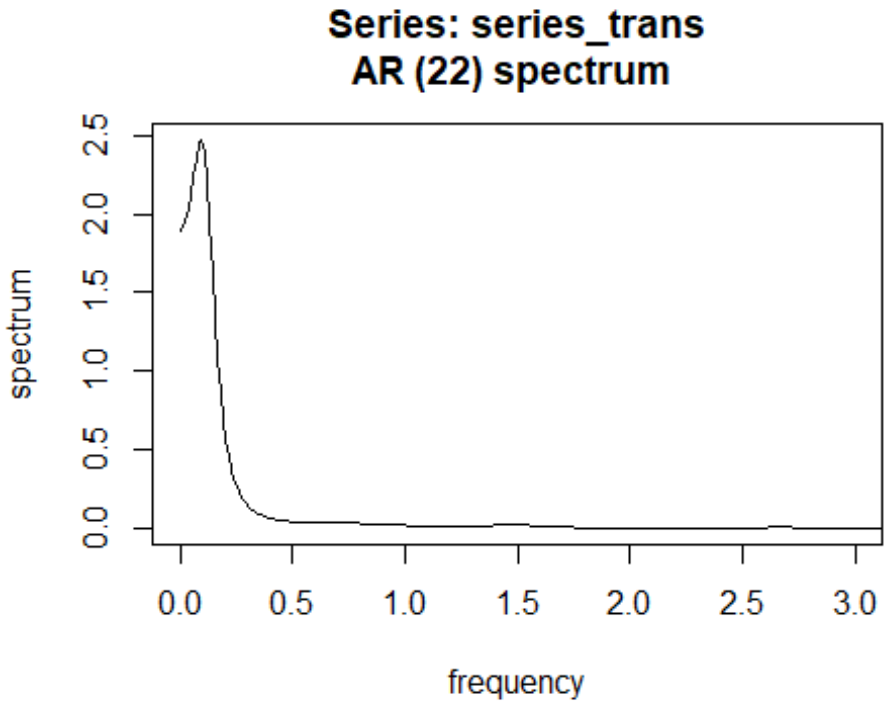


The period in years are:

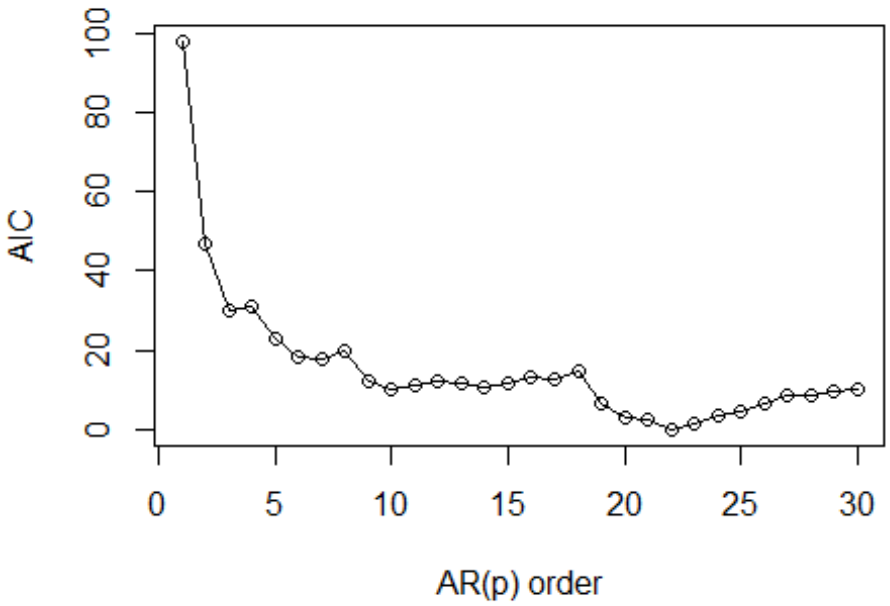
```
## [1] 0.4063492
```

Noting $\chi_2^2(.025) = .05$ and $\chi_2^2(.975) = 7.38$, we can obtain approximate 95% confidence intervals for the frequencies of interest. The periodogram of the sunspotz series is at the 0.406349206349206 year circle. An approximate 95% confidence interval for the spectrum is then: (0.716423596685905 , 104.385035540871)

Next we plot a AR(p) spectral representation of the preiodogram.

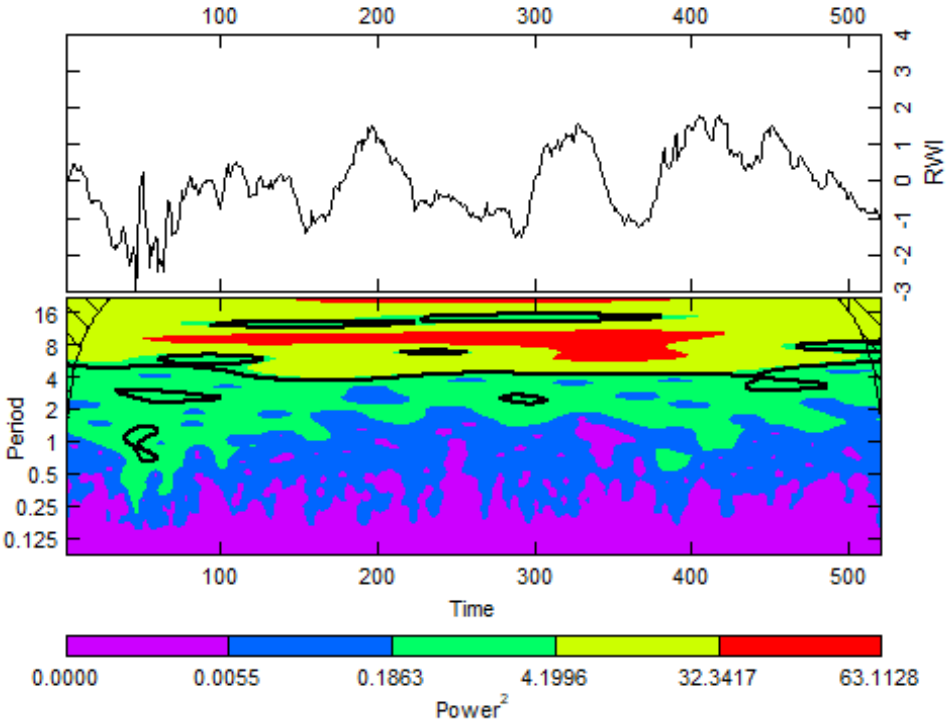



```
##
## Call:
## ar(x = series_trans, order.max = 30)
##
## Coefficients:
##       1        2        3        4        5        6        7
 8
##  1.3793  -0.6538   0.3621  -0.2878   0.2967  -0.1743   0.0253   0.08
85
##       9       10       11       12       13       14       15
16
##  0.0521  -0.2045   0.1941  -0.1866   0.1056   0.0470  -0.0825   0.08
04
##      17       18       19       20       21       22
## -0.1117   0.1771  -0.1949  -0.0555   0.1979  -0.0929
##
## Order selected 22  sigma^2 estimated as  0.03389
```



We find that our spectrum analysis result of the data is quite similar with the AR(2) curve.

We could also use wave plot to find the cycle of the time series.



In this plot the period is in the unit of year, and we find there is some evidence that there is a 8 year period in our data.

State space model

In state space model, there are two main components which make up state space models, an observed data $y_1, \ldots, y_T$ and the unobserved states $x_1, \ldots, x_T$. The observed data are conditionally independent given the states. The transition probabilities of the states are defined as $P(x_t|x_{t-1})$ implying the Markov property of the probability of moving to the next state is only dependent on the previous state for all $t = 1, \ldots, T$. The probability of observing y at time t is given by $P(y_t|x_t)$ implying the observed data is conditional on the current state.

We will use dynamic linear gaussian model in our data analysis. It is a special case of state space models where the errors of the state and observed components are normally distributed.

$$\begin{aligned} x_0 &\sim N(m_0, C_0) \\ x_t|x_{t-1} &\sim N(G_t x_{t-1}, W_t) \\ y_t|x_t &\sim N(F_t x_t, V_t) \end{aligned}$$

Where

- F_t is a $p \times m$ matrix
- G_t is a $p \times p$ matrix
- V_t is an $m \times m$ matrix
- W_t is a $p \times p$ matrix

The formulation as a set of equations is given by

$$\begin{aligned} y_t &= F_t x_t + \nu_t \\ x_t &= G_t x_{t-1} + \omega_t \end{aligned}$$

where the errors are iid

$$\begin{aligned} \nu_t &\sim N(0, V_t) \\ \omega_t &\sim N(0, W_t) \\ x_0 &\sim N(m_0, C_0) \end{aligned}$$

In r package *dlm*, we could build this model with our dataset. We estimate the parameters using MLE. The result says that the MLE procedure is converged.

```
## [1] "converged"
```

These ML estimates are now passed to the model function to build the final parameterisation of the model.

```
model.mle$par

## [1] -0.450138441  0.000000000 -0.001707762  0.449861559
```

```
model.fit <- model.build(model.mle$par)
```

Then we can apply a Kalman filter to the data , using the *dlmFilter* function. In the output of *model.filtered*

a is the state forecast for one time step and is calculated by

$$a_t = G_t m_{t-1}$$

where

$$x_t | y_{t-1} \sim N(a_t, R_t)$$

and

$$R_t = G_t C_{t-1} G_t^T + W_t$$

f is the forecast step of the observations and is calculated by

$$f_t = F_t a_t$$

where

$$y_t | y_{t-1} \sim N(f_t, Q_t)$$

and

$$Q_t = F_t R_{t-1} F_t^T + V_t$$

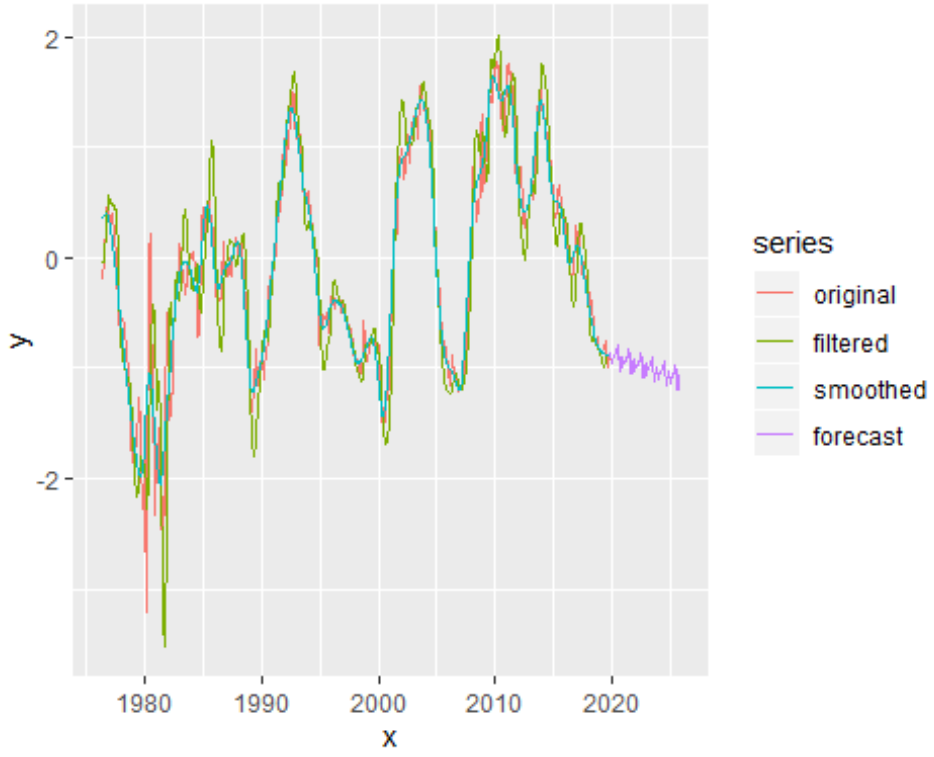
m is the posterior and is calculated by

$$m_t = R_t f_t Q_t^{-1} (y_t - f_t)$$

and

$$C_t = R_t - R_t F_t^T Q_t^{-1} F_t R_t$$

With this filtered model and the smoothed model, we could plot them together and do the



forecasting.

Conclusion

We made a time series analysis for the Economic data from the Federal Reserve Bank of St. Louis. We used ARIMA model and dynamic linear model for this data. From the result we believe those method can do a good job in modeling the time series data. For forecasting, as our data is complex trended, without seasonal effects, the forecasting in the dynamic linear model may not be informative. We can only get a wide confidence interval from the ARIMA model as well. But for the detrended data, we could forecast well. The data has a growing trend, which inflects the growth of the economy. For the periodogram analysis, the result shows that it is coincidence with the AR(2) model. We also obtained the wave plot, and find that there is some cycle in 8 year period of our time series data. This time period may be consist with the economy cycle.

Appendix: Rcode

```
knitr::opts_chunk$set(echo=FALSE, message=FALSE, warning=FALSE, paged.p
rint=FALSE)
library(astsa)
data = read.csv("T10Y2YM.csv")
knitr::kable(head(data))
series<-ts(data$T10Y2YM,frequency=12,start=c(1976,6))
end(series)
tsplot(series,ylab = "percentage")
abline(reg=lm(series~time(series)))
summary(series)
boxplot(series~cycle(series),ylab = "percentage",xlab = "Months")
library(ggplot2)
library(forecast)
ggseasonplot(ts(series[1:240],frequency = 12,start = c(1976,6)), year.l
abels=TRUE, year.labels.left=TRUE) +
  ylab("percentage") +
  ggtitle("Seasonal plot: 10-Year Treasury Constant Maturity, 1976-1996
")
ggseasonplot(ts(series[241:521],frequency = 12,start = c(1996,7)), year
.labels=TRUE, year.labels.left=TRUE) +
  ylab("percentage") +
  ggtitle("Seasonal plot: 10-Year Treasury Constant Maturity, 1996-2019
")
library(seasonal)
fit = series %>% stl(t.window=13, s.window="periodic", robust=TRUE)
autoplot(fit)
autoplot(series, series="Data") +
  autolayer(trendcycle(fit), series="Trend") +
  autolayer(seasadj(fit), series="Seasonally Adjusted") +
  xlab("Year") + ylab("Percentage") + xlim(1976, 1990) +
  ggtitle("plots of different components from 1976-1996") +
  scale_colour_manual(values=c("green","blue","red"),
             breaks=c("Data","Seasonally Adjusted","Trend"))
autoplot(series, series="Data") +
  autolayer(ma(series,15), series="15-MA") +
  xlab("Year") + ylab("Percentage") +
  ggtitle("15-MA plot against data") +
  scale_colour_manual(values=c("Data"="grey50","15-MA"="red"),
                      breaks=c("Data","15-MA"))
par(mfrow=c(1,1))
ts_lowess <- lowess(series,f=15/521)$y
ts_lowess = ts(ts_lowess,frequency = 12,start = c(1976,6))
autoplot(series, series="Data") +
  autolayer(ma(series,15), series="15-MA") + xlim(1996,2019) +
  autolayer(ts_lowess, series="lowess f=15/521") +
  xlab("Year") + ylab("Percentage") +
  ggtitle("15-MA plot and lowess plot against data") +
  scale_colour_manual(values=c("Data"="grey50","15-MA"="red","lowess f=
```

```
15/521" = "blue"),
                        breaks=c("Data","15-MA","lowess f=15/521"))
lambda <- BoxCox.lambda(series)
print(lambda)
series_trans = BoxCox(series, lambda = lambda)
autoplot(series, series="original") +
  autolayer(series_trans, series="transformed") + xlim(1996,2019) +
  xlab("Year") + ylab("Percentage") +
  ggtitle("boxcox transform") +
  scale_colour_manual(values=c("original"="red","transformed" = "blue")
,
                        breaks=c("original","transformed"))
y1 = diff(series_trans)
y1_ma = ma(y1,15)
autoplot(diff(series_trans), series="Diff data") +
  autolayer(y1_ma, series="15-MA") +
  xlab("Year") + ylab("Percentage") +
  ggtitle("15-MA plot and lowess plot against data") +
  scale_colour_manual(values=c("Diff data"="grey50","15-MA"="red"),
                        breaks=c("Diff data","15-MA"))
fit = y1 %>% stl(t.window=13, s.window="periodic", robust=TRUE)
autoplot(fit)
library(urca)
print("Unit root test for original data")
series_trans %>% ur.kpss() %>% summary()
print("Unit root test for differenced data")
series_trans %>% diff() %>% ur.kpss() %>% summary()
series_trans %>% diff() %>% ggtsdisplay(main="")
series.train = ts(series_trans[1:450],frequency = 12, start = c(1976,6)
)
arima_2_1_1 = arima(series.train,c(2,1,1))
arima_3_1_1 = arima(series.train,c(3,1,1))
arima_2_1_2 = arima(series.train,c(2,1,2))
arima_2_1_3 = arima(series.train,c(2,1,3))
arima_2_1_4 = arima(series.train,c(2,1,4))
arima_3_1_3 = arima(series.train,c(3,1,3))
arima_3_1_2 = arima(series.train,c(3,1,2))
df = data.frame("aic_2,1,1" = -arima_2_1_1$aic,"aic_3,1,1" = -arima_3_1
_1$aic,"aic_2,1,2" = -arima_2_1_2$aic,"aic_2,1,3" = -arima_2_1_3$aic,"a
ic_2,1,4" = -arima_2_1_4$aic,"aic_3,1,2" = -arima_3_1_2$aic,"aic_3,1,3"
 = -arima_3_1_3$aic)
df
arima = arima_2_1_3
checkresiduals(arima)
autoplot(forecast(arima,h=70,),series = "ARIMA model") +
  autolayer(series_trans, series="Diff data") +
  xlab("Year") + ylab("Percentage") +
  scale_colour_manual(values=c("ARIMA model"="blue","Diff data"="red"),
                        breaks=c("ARIMA model","Diff data"))
library(astsa)
```

```r
mvspec(series_trans, log="no",spans=c(3,2),xlim = c(0,1))
mvspec(series_trans, log="no",spans=c(3,2))
se.per = mvspec(series_trans, log="no",spans=c(3,2),plot = FALSE)
l2 = which(se.per$spec>1.97)[1]
Period = l2 * se.per$bandwidth
n = length(se.per$spec)
Pr2 = se.per$spec[l2]
# conf intervals - returned value:
U = qchisq(.025,2)
L = qchisq(.975,2)
Period
spaic = spec.ar(series_trans, log="no", xlim=c(0,3))  #best fitting AR
is determined with spec.ar command
(soi.ar = ar(series_trans, order.max=30))                          # est
imates and AICs
plot(1:30, soi.ar$aic[-1], type="o", ylab= 'AIC', xlab='AR(p) order') #
 plot AICs
library(dplR)
wave.out <- morlet(y1 = series_trans, p2 = 8, dj = 0.1, siglvl = 0.95)
# p2=6 <=> estimate out to 2^8 = 256 months dj <=> controls the frequen
cy
# resolution hack the period estimate to be in years, not months
wave.out$period <- wave.out$period/12
levs <- quantile(wave.out$Power, c(0, 0.25, 0.5, 0.75, 0.95, 1))
wavelet.plot(wave.out, wavelet.levels = levs, crn.ylim = c(-3,4))
data(co2)
library(dlm)
library(zoo)
library(gridExtra)

# forecasting using state sapce models
model.build <- function(p) {
    return(
      dlmModPoly(2, dV=p[1], dW=p[2:3]) +
      dlmModSeas(12, dV=p[4])
    )
}

model.mle <- dlmMLE(series_trans, parm=c(0.1, 0, 1, 1), build=model.bui
ld)
if(model.mle$convergence==0) print("converged") else print("did not con
verge")
model.mle$par
model.fit <- model.build(model.mle$par)
model.filtered <- dlmFilter(series_trans, model.fit)
model.smoothed <- dlmSmooth(series_trans, model.fit)
n <- 6*12
model.forecast <- dlmForecast(model.filtered, nAhead=n)

x <- index(series_trans)
```

```
xf <- seq(max(x), max(x)+n/12, 1/12)
aa <- model.forecast$a[,-1]*(-1)
aa <- cbind(model.forecast$a[,1], aa)
a <- drop(model.forecast$a%*%t(FF(model.fit)))
a <- c(tail(series_trans,1), a)
df <- rbind(
  data.frame(x=x, y=as.numeric(series_trans), series="original"),
  data.frame(x=x, y=apply(model.filtered$m[-1,1:2], 1, sum), series="fi
ltered"),
  data.frame(x=x, y=apply(model.smoothed$s[-1,1:2], 1, sum), series="sm
oothed"),
  data.frame(x=xf, y=a, series="forecast")
)
g.dlm <- ggplot(subset(df, x>1970), aes(x=x, y=y, colour=series)) + geo
m_line()
g.dlm
```