Assignment 2

Antoine CLOUTE, Leopold Granger, Clara Besnard

Load the dataset

Note: Some modifications were made on the original file prior to processing ie separate data into separated columns.

```
#load Data
data=read.csv("Inflation_France_OECD.csv",header = TRUE, stringsAsFactors=F)
data$Time<- as.Date(data$Time,format="%m/%d/%Y")
x=data$Time
y=data$CPI
```

Question 1

Key Statistics of CPI

```
summary(y)
```

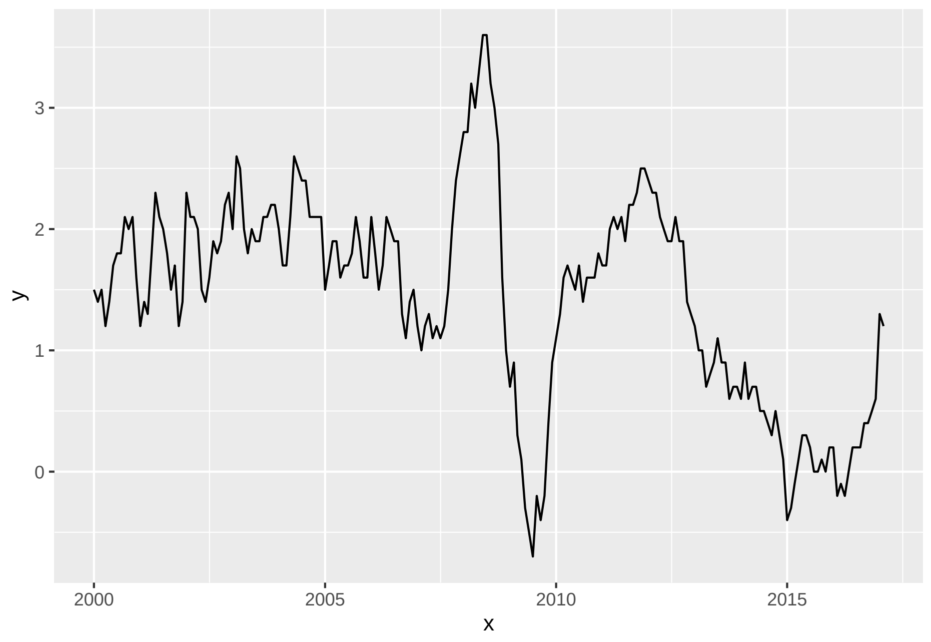
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -0.700   0.900   1.600   1.441   2.000   3.600
```

Plot of CPI time serie

```
require(ggplot2)
```

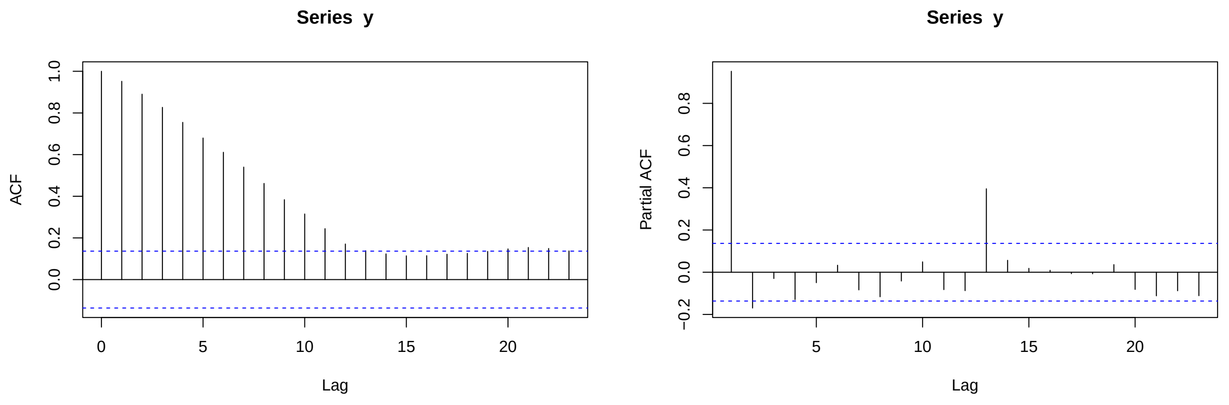
```
## Loading required package: ggplot2
```

```
#Plot CPI across period
p <- ggplot(data, aes(x=x, y=y))+
       geom_line()
p
```



Autocorrelation plot

```
rho = acf(y)
rho1 = pacf(y)
```



The autocorrelation plot shows that there is autocorrelation up to lag 12 (i.e CPI value at t-12 months has some correlation with the CPI value at time t). Indeed, the ACF is significant, geometrically declining up to lag 12. This means that the monthly inflation rate is positively correlated to the 12 month before, noting that the more recent lags have a more sizeable influence than the latter. We can conclude that the residuals are autocorrelated.

Question 2

Split the dataset in two subsets (50% split)

```
# split data in half

df_top <- data[row.names(data) %in% 1:103, ]
df_bottom <- data[row.names(data) %in% (103+1):nrow(data), ]
```

Fit an AR(1) model to the top 50% of the dataset

```r
# fit AR(1)
n = length(df_top$CPI)
arOLS = lm(df_top$CPI[2:n]~df_top$CPI[1:n-1])
summary(arOLS)
```

```
##
## Call:
## lm(formula = df_top$CPI[2:n] ~ df_top$CPI[1:n - 1])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.61859 -0.17776  0.00183  0.15134  0.83038
##
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)           0.21251    0.10757   1.976    0.051 .
## df_top$CPI[1:n - 1]  0.89794    0.05535  16.223   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2744 on 100 degrees of freedom
## Multiple R-squared:  0.7247, Adjusted R-squared:  0.7219
## F-statistic: 263.2 on 1 and 100 DF,  p-value: < 2.2e-16
```

An AR(1) model is under the form: $y_t = c + \phi y_{t-1} + \epsilon_t$

where y_t is the CPI at date t, and ϵ is white noise with variance σ^2.

Here the AR(1) model can be estimated as: $y_t = 0.21251 + 0.89794 CPI_{t-1} + \epsilon_t$

Question 3-4

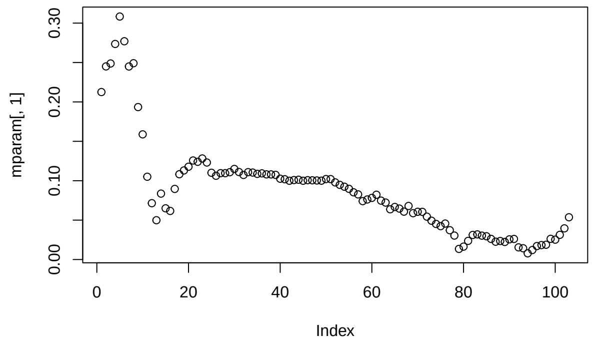
We first define two functions: AR1_forecast: Build recursively n forecast (AR(1) model) for t horizon forecast_performance: evaluate the forecast against the actual values of the test set

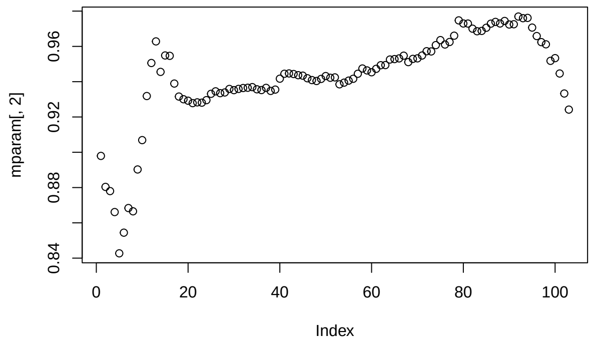
```r
AR1_forecast<-function(vy,nfor,hor)
{
  T=length(vy);
  Tout=round(nfor*T);
  Tin=T-Tout;
  mforecast=array(0,dim=c(Tout,hor));
  mparam=array(0,dim=c(Tout,2)); #2 parameters in AR1 model

  for(t in 1:Tout)
  {
    vyestim=vy[1:Tin+t-1];
    Testim=length(vyestim);
    AR1_estim=lm(vyestim[2:Testim]~vyestim[1:Testim-1]);
    mparam[t,]=AR1_estim$coefficients;
    vyfor=c(1,vyestim[Testim]);

    for(h in 1:hor)
    {
      mforecast[t,h]=vyfor %*% mparam[t,];
      vyfor=c(1,mforecast[t,h]);
    }
  }

  plot(mparam[,1]);
  plot(mparam[,2]);


  return(mforecast)
}

forecast_performance<-function(vy,nfor,mforecasts)
{
  T=length(vy);
  Tout=round(nfor*T);
  Tin=T-Tout;
  vyestim=vy[1:Tin];
  vyout=vy[Tin+1:T];
  hor=length(mforecasts[1,]);

  vrmse=array(0,dim=c(hor,1));
  vmae=array(0,dim=c(hor,1));

  for(h in 1:hor)
  {
    Teval=Tout-h+1;
    verror=array(0,dim=c(Teval,1));
    verror_sq=array(0,dim=c(Teval,1));

    for(t in 1:Teval)
    {
      verror[t,1]=(mforecasts[t,h]-vyout[t+h-1]);
      verror_sq[t,1]=(mforecasts[t,h]-vyout[t+h-1])^2;
    }

    vrmse[h,1]=sqrt(mean(verror_sq));
    vmae[h,1]=mean(verror);

  }

  print("Forecast horizons: ");
  print(hor)
  print("Mean Error: ");
  print(vmae);
  print("Root Mean Squared Error: ");
  print(vrmse)

  return(c(vmae,vrmse));
}
```

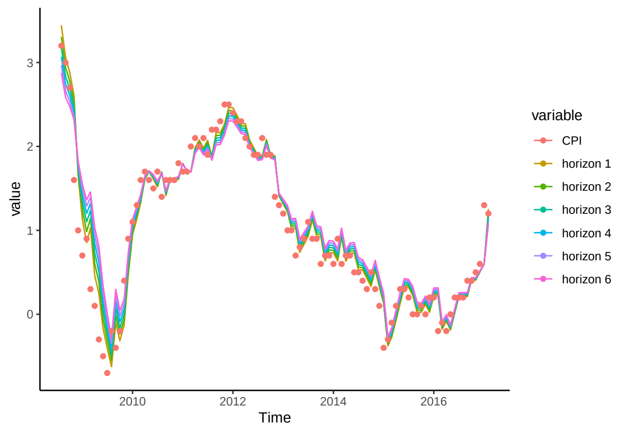
We forecast CPI value for horizon 1 to 6, parameters are re-estimated after each forecast.

```
#Build the forecasts for horizon 1 to 6 (50% of the dataset)
forecast1= AR1_forecast(data$CPI,0.5,6)

# Plot the forecasted values and the actual values
```

```r
library(reshape)
library(ggplot2)
cl <- rainbow(6)
data_forecast=as.data.frame(cbind.data.frame(df_bottom$Time,forecast1))
colnames(data_forecast) = c("Time","horizon 1", "horizon 2", "horizon 3","horizon 4","horizon 5","horiz
data_forecast=merge.data.frame(df_bottom,data_forecast, by='Time')

data_long<- melt(data_forecast, id="Time")  # convert to long format

p<-ggplot(data=data_long,
       aes(x=Time, y=value, colour=variable)) +
       geom_line(data = subset(data_long, variable != "CPI")) +
      geom_point(data = subset(data_long, variable == "CPI"))+
      theme_classic()
p
```







Based on the graph plotting the autoregressive coefficient (params2), we can observe that there are 3 breaks:

- At index 5: inflexion of the downward trend
- At index 15: parameter coefficient go down
- At index 90: parameter coefficent go down after staying relatively constant (slight increase) from index 20 to 90

Compute MSE and RMSE for each forecast (horizon 1 to 6)

```r
AR_1=forecast_performance(df_bottom$CPI,1,forecast1)
```

```
## [1] "Forecast horizons: "
## [1] 6
## [1] "Mean Error: "
##             [,1]
## [1,] 0.03976279
## [2,] 0.07528754
## [3,] 0.11511093
## [4,] 0.15273059
## [5,] 0.18001438
## [6,] 0.20086159
## [1] "Root Mean Squared Error: "
##            [,1]
## [1,] 0.2570463
## [2,] 0.4016912
## [3,] 0.5315013
## [4,] 0.6435342
## [5,] 0.7294470
## [6,] 0.7945825
```

```
data_forecast=as.data.frame(cbind.data.frame(df_bottom$Time,forecast1))
colnames(data_forecast) = c("Time","horizon 1", "horizon 2", "horizon 3","horizon 4","horizon 5","horiz
data_forecast=merge.data.frame(df_bottom,data_forecast, by='Time')
```

MFE: The mean forecast error is the average of the forecast error values. Technically with a perfect forecast technique, this number would be zero.

$MFE = \frac{1}{n}\sum_{i=1}^{n}(F_i - \hat{F}_i)$

MSFE: The mean squared forecast error is the squared difference between the actual CPI and the forecast CPI value.

$MSFE = \frac{1}{n}\sum_{i=1}^{n}(F_i - \hat{F}_i)^2$

RMSE: This is the root of the MSFE.

$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(F_i - \hat{F}_i)^2}$

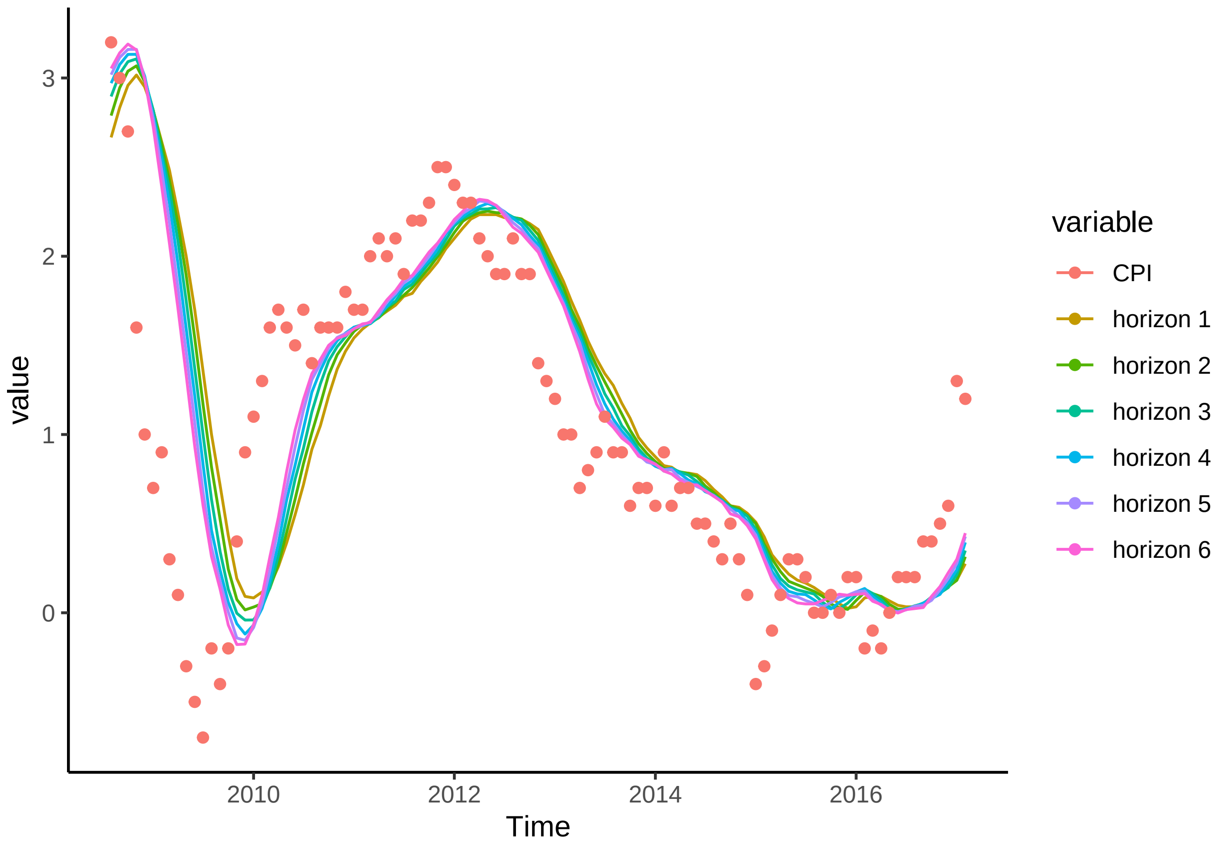
Question 5

We build a function which forecast the CPI based on the mean of the last 12 CPI observations.

```
ARMean_forecast<-function(vy,nfor,hor)
{
  T=length(vy);
  Tout=round(nfor*T);
  Tin=T-Tout;
  mforecast=array(0,dim=c(Tout,hor));

  for(t in 1:Tout)
  {
    vyestim=vy[1:Tin+t-1];
    Testim=length(vyestim);

   for(h in 1:hor)
    {
     ARMean_estim=mean(vyestim[91:102+h]);
     mforecast[t,h]=ARMean_estim;
     vyestim<-append(vyestim,ARMean_estim );
     Testim=length(vyestim);
```

```r
    }
  }
  return(mforecast)
}
```

```r
#Build the forecast for horizon 1 to 6
forecast2= ARMean_forecast(data$CPI,0.5,6)


# Plot the forecasted values and the actual values
data_forecast2=as.data.frame(cbind.data.frame(df_bottom$Time,forecast2))
colnames(data_forecast2) = c("Time","horizon 1", "horizon 2", "horizon 3","horizon 4","horizon 5","hori
data_forecast2=merge.data.frame(df_bottom,data_forecast2, by='Time')

data_long<- melt(data_forecast2, id="Time")  # convert to long format

p<-ggplot(data=data_long,
       aes(x=Time, y=value, colour=variable)) +
       geom_line(data = subset(data_long, variable != "CPI")) +
      geom_point(data = subset(data_long, variable == "CPI"))+
      theme_classic()

p
```



```
#Compute RMSE and MFE
AR_mean=forecast_performance(data$CPI,0.5,forecast2)
```

```
## [1] "Forecast horizons: "
## [1] 6
## [1] "Mean Error: "
##            [,1]
## [1,] 0.1536408
## [2,] 0.1740605
## [3,] 0.1938440
## [4,] 0.2117333
## [5,] 0.2200456
## [6,] 0.2239759
## [1] "Root Mean Squared Error: "
##            [,1]
## [1,] 0.7939373
## [2,] 0.8583786
## [3,] 0.9239741
## [4,] 0.9880102
## [5,] 1.0413515
## [6,] 1.0827896
```

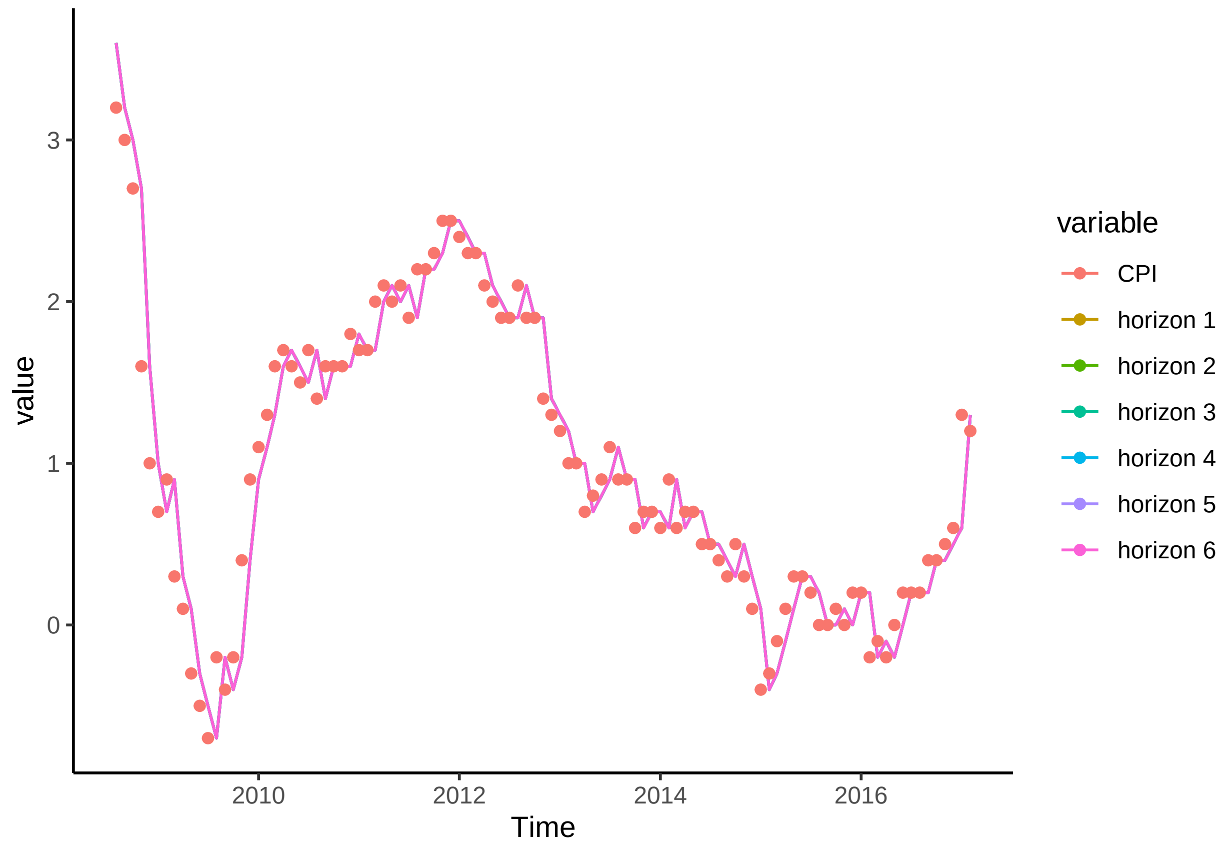
Question 6

Mathematically: $y_t = y_{t-1}$

```
AR_RW_forecast<-function(vy,nfor,hor)
{
  T=length(vy);
  Tout=round(nfor*T);
  Tin=T-Tout;
  mforecast=array(0,dim=c(Tout,hor));

  for(t in 1:Tout)
  {
    vyestim=vy[1:Tin+t-1];
    Testim=length(vyestim);

   for(h in 1:hor)
    {
     AR_RW_estim=vyestim[Testim];
     mforecast[t,h]=AR_RW_estim;
    }
  }
  return(mforecast)
}
```

```
#Build the forecast for horizon 1 to 6
forecast3= AR_RW_forecast(data$CPI,0.5,6)

# Plot the forecasted values and the actual values
data_forecast3=as.data.frame(cbind.data.frame(df_bottom$Time,forecast3))
colnames(data_forecast3) = c("Time","horizon 1", "horizon 2", "horizon 3","horizon 4","horizon 5","hori
data_forecast3=merge.data.frame(df_bottom,data_forecast3, by='Time')
```

```
data_long<- melt(data_forecast3, id="Time")  # convert to long format

p<-ggplot(data=data_long,
       aes(x=Time, y=value, colour=variable)) +
       geom_line(data = subset(data_long, variable != "CPI")) +
      geom_point(data = subset(data_long, variable == "CPI"))+
      theme_classic()

p
```



```
#Compute RMSE and MFE
AR_RW=forecast_performance(df_bottom$CPI,1,forecast3)
```

```
## [1] "Forecast horizons: "
## [1] 6
## [1] "Mean Error: "
##            [,1]
## [1,] 0.02330097
## [2,] 0.04215686
## [3,] 0.06633663
## [4,] 0.08900000
## [5,] 0.10202020
## [6,] 0.10918367
## [1] "Root Mean Squared Error: "
##           [,1]
## [1,] 0.2558064
## [2,] 0.4028088
## [3,] 0.5418158
## [4,] 0.6668583
## [5,] 0.7656119
## [6,] 0.8466621
```

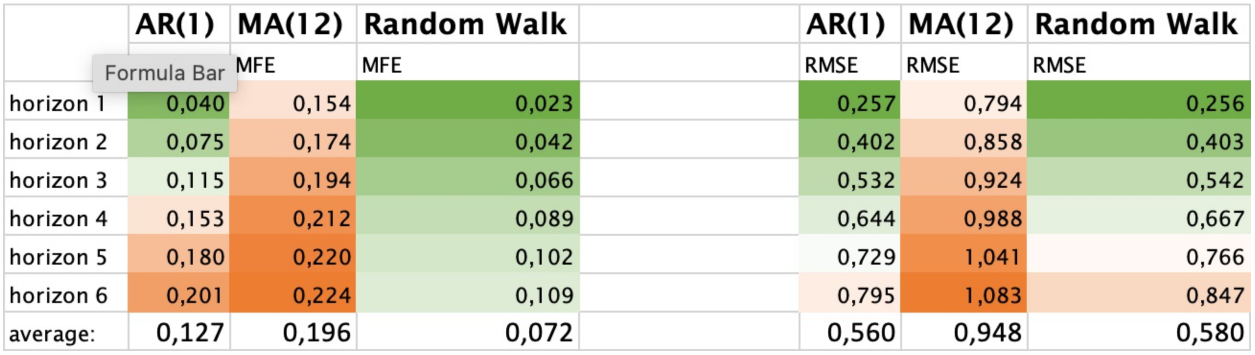
Question 7

Summary of the MFE and RMSE for the 3 different models:

```
df_ME=as.data.frame(cbind(AR_1[1:6],AR_mean[1:6],AR_RW[1:6]))
colnames(df_ME) = c("AR(1)", "MA(12)", "Random Walk")
rownames(df_ME) = c("horizon 1", "horizon 2", "horizon 3","horizon 4","horizon 5","horizon 6")



df_RMSE=as.data.frame(cbind(AR_1[7:12],AR_mean[7:12],AR_RW[7:12]))
colnames(df_RMSE) = c("AR(1)", "MA(12)", "Random Walk")
rownames(df_RMSE) = c("horizon 1", "horizon 2", "horizon 3","horizon 4","horizon 5","horizon 6")


summary_df=cbind(df_ME,df_RMSE)
summary_df=rbind( c('MFE','MFE','MFE','RMSE','RMSE','RMSE'), summary_df)
as.data.frame.matrix(summary_df)
```

```
##                          AR(1)             MA(12)        Random Walk
## 1                          MFE                MFE                MFE
## horizon 1 0.0397627850864675 0.153640776699029 0.0233009708737864
## horizon 2 0.0752875382144245  0.17406045751634  0.042156862745098
## horizon 3  0.115110927405255 0.193843967730106 0.0663366336633663
## horizon 4  0.152730588246532  0.21173331404321              0.089
## horizon 5   0.18001438312297 0.220045559926685  0.102020202020202
## horizon 6  0.200861585705424 0.223975865242413  0.109183673469388
##                        AR(1)             MA(12)       Random Walk
## 1                       RMSE               RMSE              RMSE
## horizon 1 0.257046325275983 0.793937298995145 0.255806358802676
## horizon 2 0.401691168376696  0.85837863803873 0.402808765992976
## horizon 3 0.531501286065167 0.923974087829801 0.541815795668273
## horizon 4 0.643534151928514 0.988010247627184 0.666858305789168
## horizon 5  0.72944699302396  1.04135152354242 0.765611922687739
## horizon 6 0.794582450060927  1.08278963018913 0.846662113652121
```

```
knitr::include_graphics("table.png")
```


Formula Bar

	AR(1)	MA(12)	Random Walk		AR(1)	MA(12)	Random Walk
		MFE	MFE		RMSE	RMSE	RMSE
horizon 1	0,040	0,154	0,023		0,257	0,794	0,256
horizon 2	0,075	0,174	0,042		0,402	0,858	0,403
horizon 3	0,115	0,194	0,066		0,532	0,924	0,542
horizon 4	0,153	0,212	0,089		0,644	0,988	0,667
horizon 5	0,180	0,220	0,102		0,729	1,041	0,766
horizon 6	0,201	0,224	0,109		0,795	1,083	0,847
average:	0,127	0,196	0,072		0,560	0,948	0,580

Interpretations

MFE: All three models seem to overestimate inflation since all MFEs are positive.

In absolute terms, the random walk model is more accurate that the AR(1) and MA(12) model since it has the lowest mean error. All models get less accurate as the model forecasts further into the future.

RMSE: The AR(1) model seems to be slightly more accurate than the random walk model (except for Horizon 1). The MA(12) model is less accurate than both the AR(1) and random walk model.

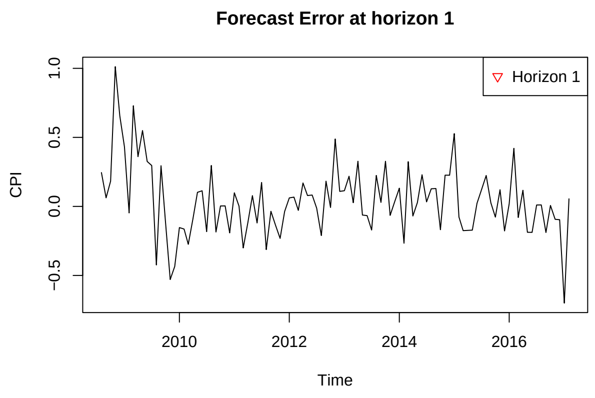
Question 8

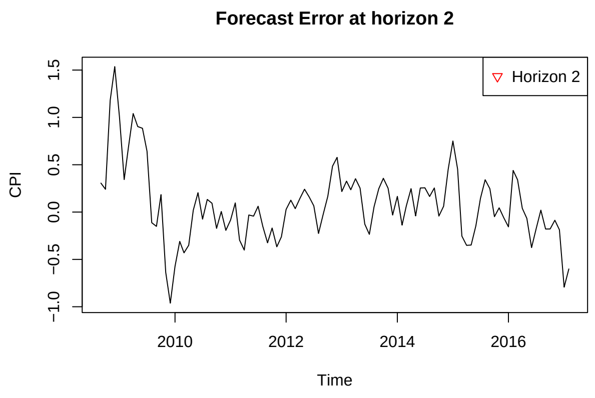
Define function to plot the forecast error

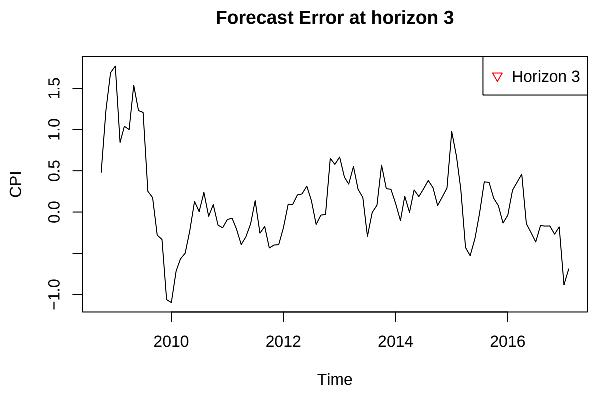
```
plot_forecast_error<-function(vy,nfor,mforecasts)
{
  T=length(vy);
  Tout=round(nfor*T);
  Tin=T-Tout;
  vyestim=vy[1:Tin];
  vyout=vy[Tin+1:T];
  hor=length(mforecasts[1,]);


  for(h in 1:hor)
  {
    Teval=Tout-h+1;
    verror=array(0,dim=c(Teval,1));


    for(t in 1:Teval)
    {
      verror[t,1]=(mforecasts[t,h]-vyout[t+h-1]);
    }
    x= 1+h-1
    plot(df_bottom$Time[x:103],verror,col="Black", type="l", xlab="Time", ylab="CPI") # Actual values
    legend("topright", legend = paste0("Horizon ", h), col=cl, pch=6)
    title(paste0("Forecast Error at horizon ", h))
  }




}
```

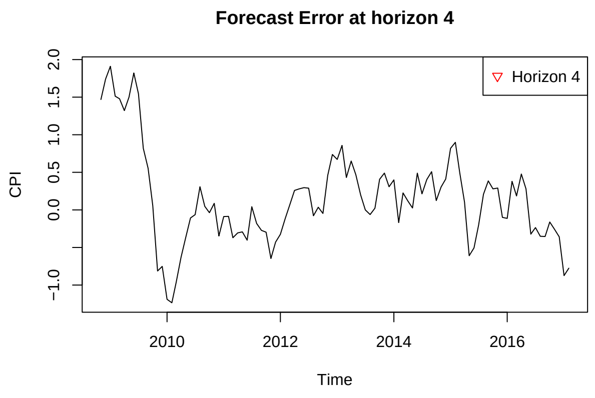
AR(1) model

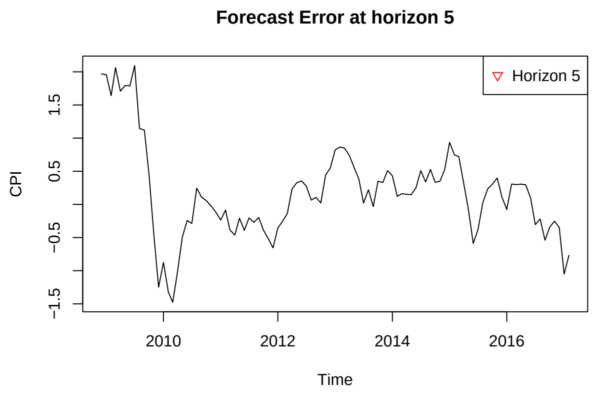
```
plot_forecast_error(data$CPI,0.5,forecast1)
```

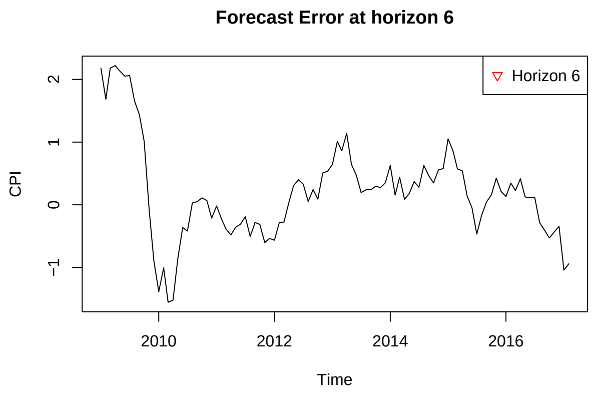






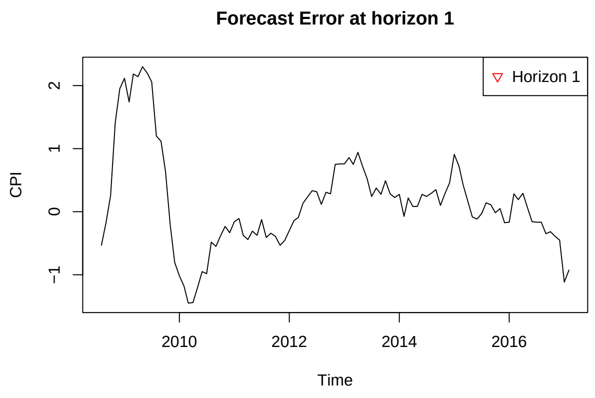


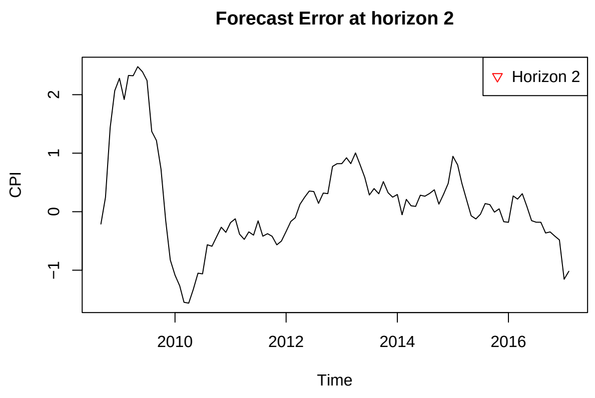


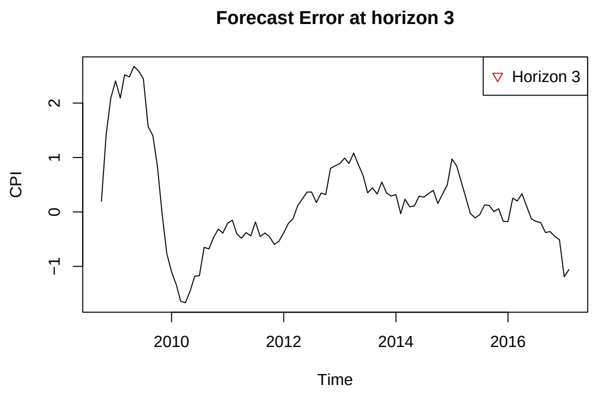


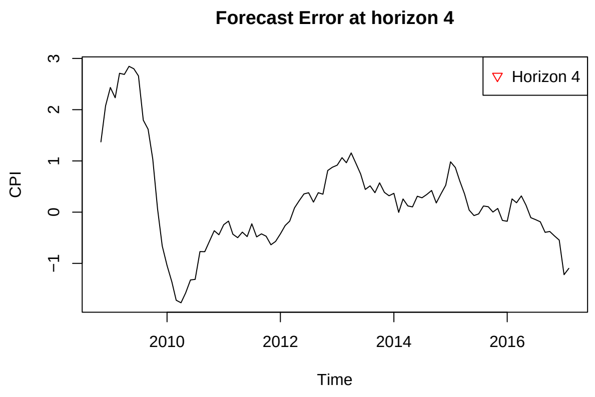
MA(12) model

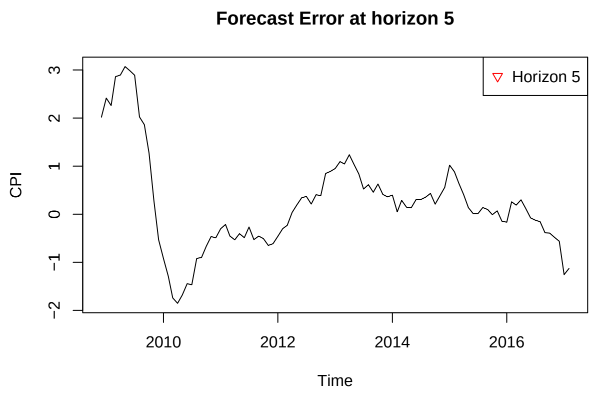
```
plot_forecast_error(data$CPI,0.5,forecast2)
```

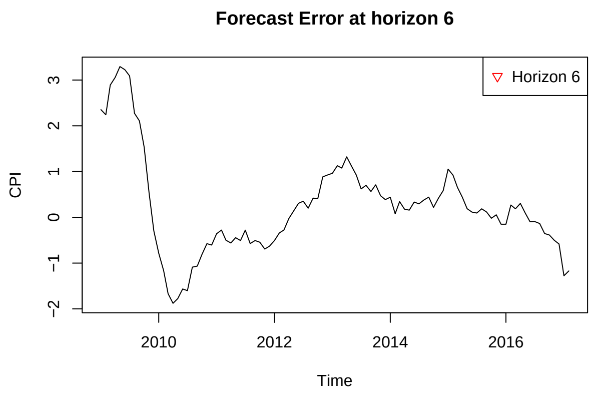






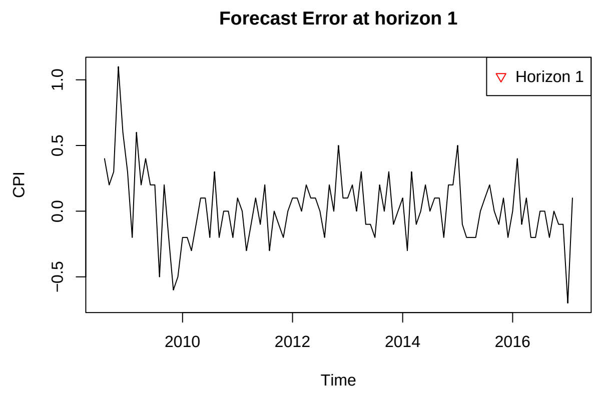


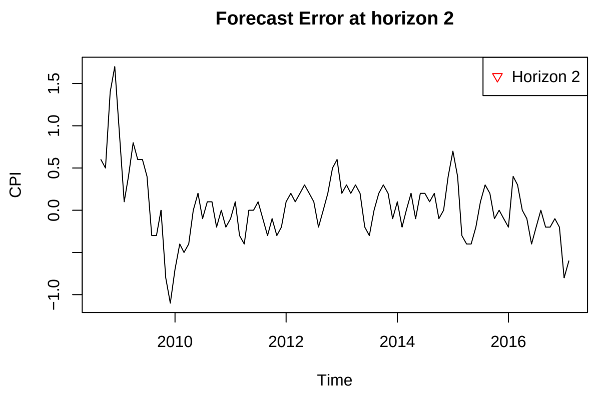


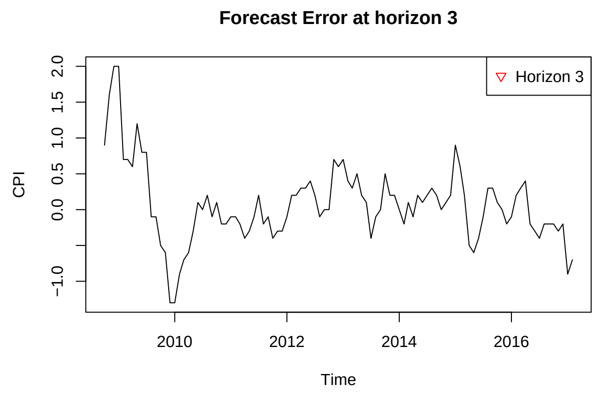


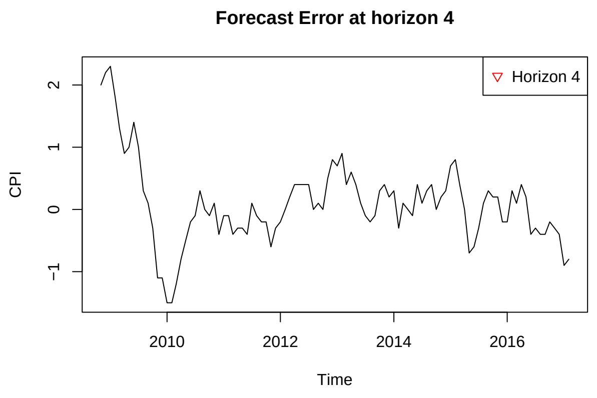
Random Walk model

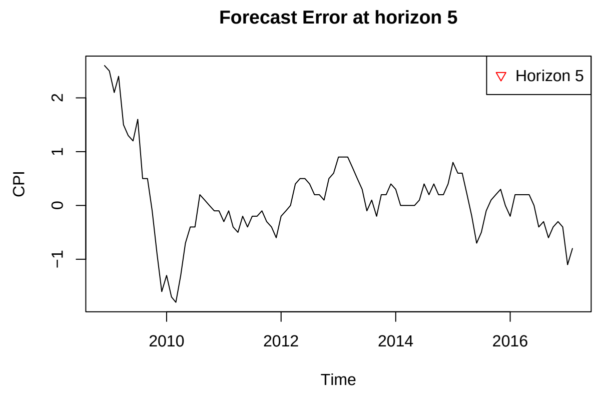
```
plot_forecast_error(df_bottom$CPI,1,forecast3)
```

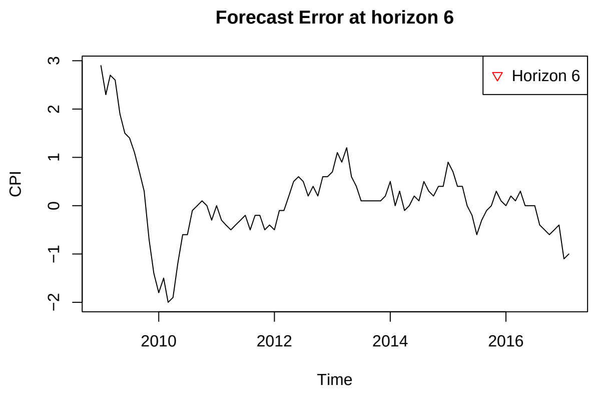












Observations:

- First, as a general observation the prior graphs clearly shows that the forecast errors (amplitude) increases for all models as the horizon increases.
- The comparison of the forecast errors for all 3 models at horizon 1 shows that the naive model (MA(12)) at any time has overall a higher level of error compare to both AR(1) and RW models.

- AR and RW error follows a similar trends across time.
- For all 3 models, the period with the highest level of errors is the period 2009-2010 which can be explained by the financial crisis that happened in that period (large fluctuation of the CPI over month). As the subsequent period has less over month fluctuation, the forecast error is stabilizing and converge toward 0.