Econ 104 Project 2

Jiarui Song, Yiyang Sun, Leming Zhu

2024-02-16

Group Members: Yiyang Sun(705767597), Jiarui Song(905724494), Leming Zhu(605750248)

```r
# Load necessary libraries
library(MASS)
library(car)
```

```
## Loading required package: carData
```

```r
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```r
library(leaps)
library(Boruta)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```r
library(tseries)
library(vars)
```

```
## Loading required package: strucchange
```

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

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

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

```
## Loading required package: sandwich
```

```
## Loading required package: urca

## Loading required package: lmtest
```

```r
library(Metrics)
```

```
##
## Attaching package: 'Metrics'

## The following object is masked from 'package:forecast':
##
##     accuracy
```

```r
library(ggplot2)
```

Import Data

```r
annual <- read.csv("Annual.csv")
data <- annual
head(annual)
```

```
##   Year Bond.Yield stock.price Implicit.Price.Index Velocity.of.Money
## 1 1900       3.30        7.84               24.317              2.49
## 2 1901       3.25        8.42               24.154              2.44
## 3 1902       3.30        7.21               24.971              2.31
## 4 1903       3.45        7.05               25.220              2.29
## 5 1904       3.60        8.99               25.530              2.14
## 6 1905       3.50        9.64               26.064              2.13
```

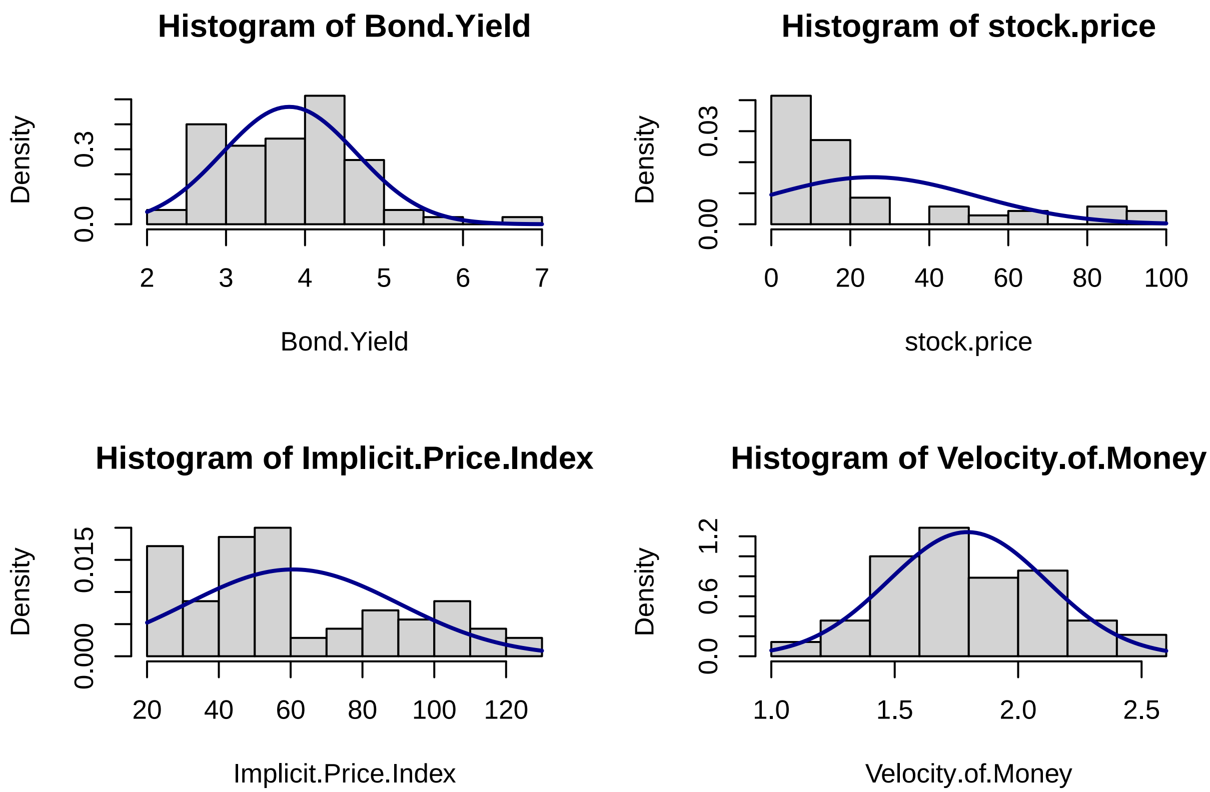
Data Introduction

In this project, we consider the following four time series datasets: Annual US bond yield from 1900 to 1969; Annual US common stock price from 1900 to 1969; Annual US velocity of money from 1900 to 1969; and Annual US implicit price index from 1900 to 1969. We aim to predict annual US common stock price using annual US bond yield, and predict annual US implicit price index using annual US velocity of money. The four datasets are downloaded from Kaggle, which in turn is incorporated from the Time Series Data Library, created by Rob Hyndman, Professor of Statistics at Monash University, Australia. The link is here: https://www.kaggle.com/datasets/krish525/open-time-series-data.

PART 1 Provide a descriptive analysis of the variables.

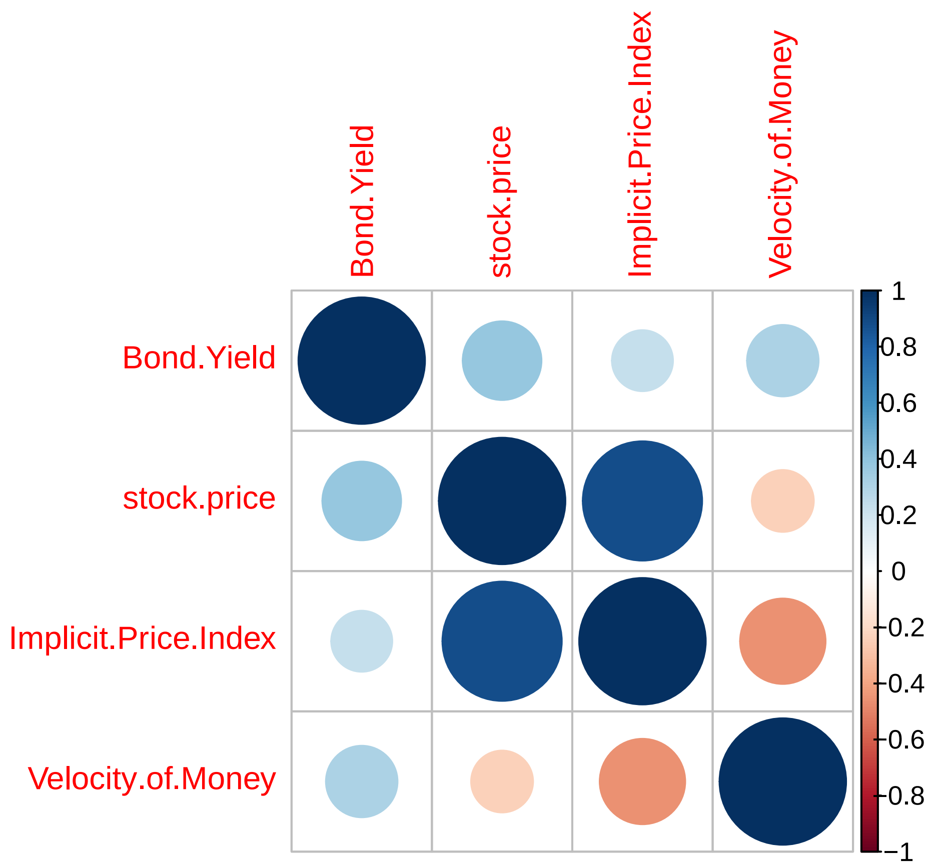
```r
# histograms and fitted distribution
par(mfrow=c(2, 2))
for (i in 2:5) {
  # store the fitted values
  fit_dist <- fitdistr(annual[, i], "normal")
  # plot the histograms
```

```
  hist(annual[, i], probability = TRUE, main = paste("Histogram of", colnames(annual)[i]),
       xlab = paste(colnames(annual)[i]))
  # plot the fitted distribution
  curve(dnorm(x, mean = fit_dist$estimate[1], sd = fit_dist$estimate[2]),
        col = "darkblue", lwd = 2, add = TRUE)
}
```



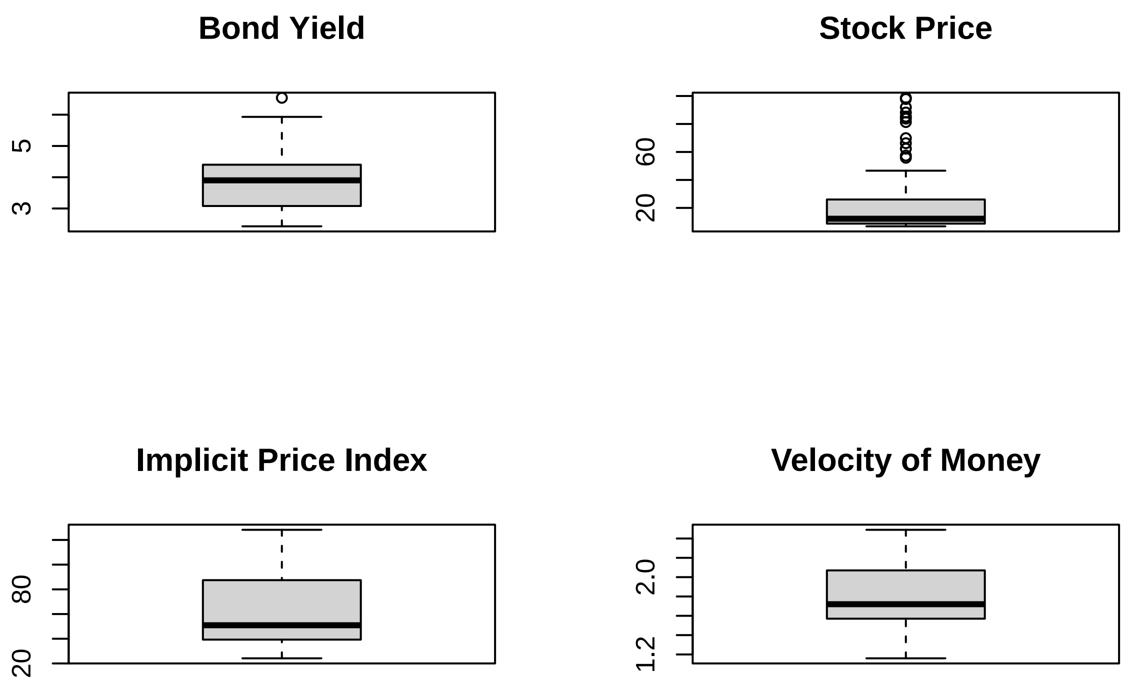
- The histogram for Bond Yield shows the distribution of average bond yields over time. We seem that the most common yield rates are from approximately 2.5% to 5%. The distribution of this dataset also looks like to be a normal distribution, but very slightly right-skewed. It indicates that high yield rates are less common, and most of the times the economy remains from 2.5% to 5%.
- The histogram for average Stock Price suggests how stock prices are distributed over time, under the influence of market fluctuations. We see that the distribution is heavily right-skewed, where most of the times the prices are from 0 - 20, where also occasionally rises all the way to 80 - 100. They are affected by many possible issues: market growth; economic recession and instability; inflation, and so on.
- The histogram for the Implicit Price Index could provide insights into inflation trends over the years. We see that index from 20 - 60 are generally more common, and the overall distribution is also right-skewed. It suggests two things: the GDP, no matter real or nominal, generally increases through time; and the inflation rate is most of the times controlled at a low level. High implicit price indices indicate times when inflation rates are higher.
- The histogram for the Velocity of Money looks like a normal distribution with no obvious skewedness. It is the measurement of overall economy's activity on how quickly money is circulated around the economy.

```r
corr_matrix <- cor(annual[,-1]) # 'Year' is excluded
corrplot(corr_matrix, method="circle")
```



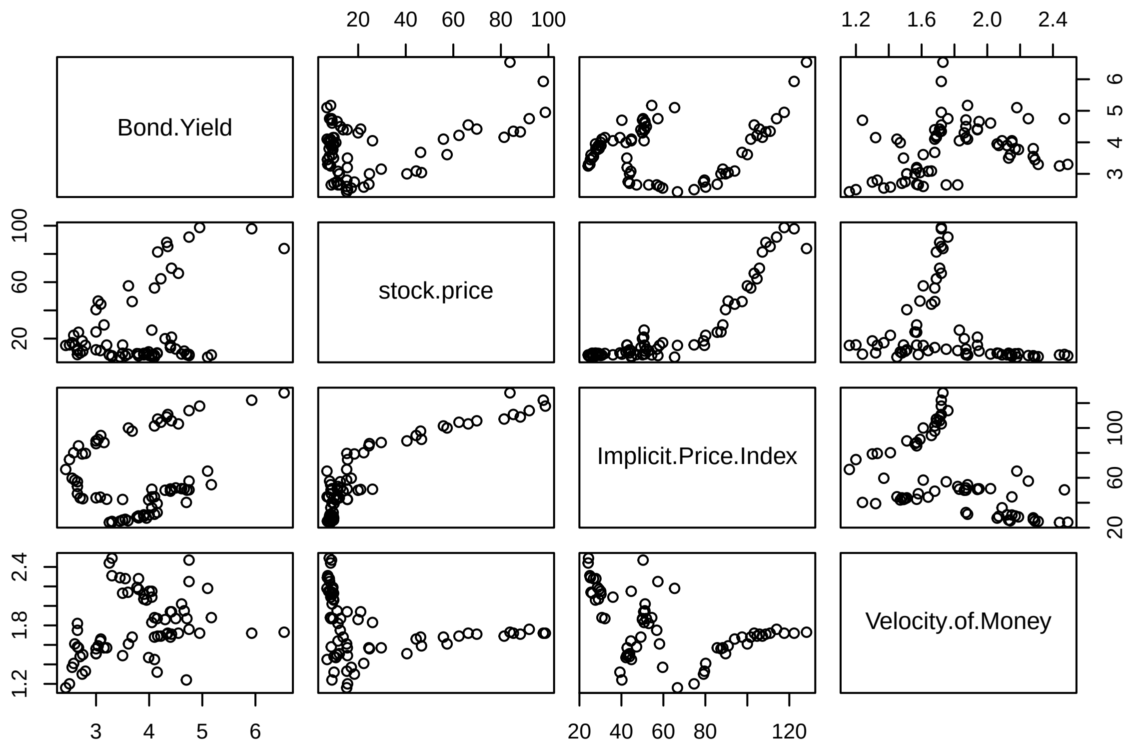
- The correlation plot provides a visual summary of how each of the variables relates to one another. High positive correlation coefficients would suggest that variables move together in the same direction, whereas high negative coefficients indicate an inverse relationship. We can see that there are moderate positive linear relationships between Bond Yield and all other variables; and there are moderate negative linear relationships between velocity of money and stock price and implicit price index. Among all variables, the strongest relationship is the positive linear relationship between stock price and implicit price index.

```r
par(mfrow=c(2, 2))
boxplot(annual$Bond.Yield, main="Bond Yield")
boxplot(annual$stock.price, main="Stock Price")
boxplot(annual$Implicit.Price.Index, main="Implicit Price Index")
boxplot(annual$Velocity.of.Money, main="Velocity of Money")
```



- The box plot for Bond Yield seems relatively symmetrical which corresponds to the histogram plotted earlier. There is only one outlier, and the spread of the data (interquartile range) is moderate, which indicates a close range of values over time.
- The box plot for Stock Price shows numerous large outliers, and again corresponds with the histogram, which suggests a heavily-right-skewed distribution. This implies more variability in Stock Price.
- The Implicit Price Index box plot has a median line close to lower quartile range, and has no outliers. It again indicates a right-skewed distribution corresponding to what we've seen in histogram.
- The box plot for Velocity of Money, again, shows no outliers and corresponds to what we've seen in histogram, a right-skewed distribution.

```
plot(annual[,-1])
```



- The scatterplot matrix is shown above. There are generally linear relation patterns between all variables. For example, we could observe some linear patterns between Bond.Yield and the other variables, as we would expect from the correlation plot. The patterns between the velocity of money and the other variables are less obvious, though. Hence, we need to examine the correlation between the variables more in the following sections of the project.

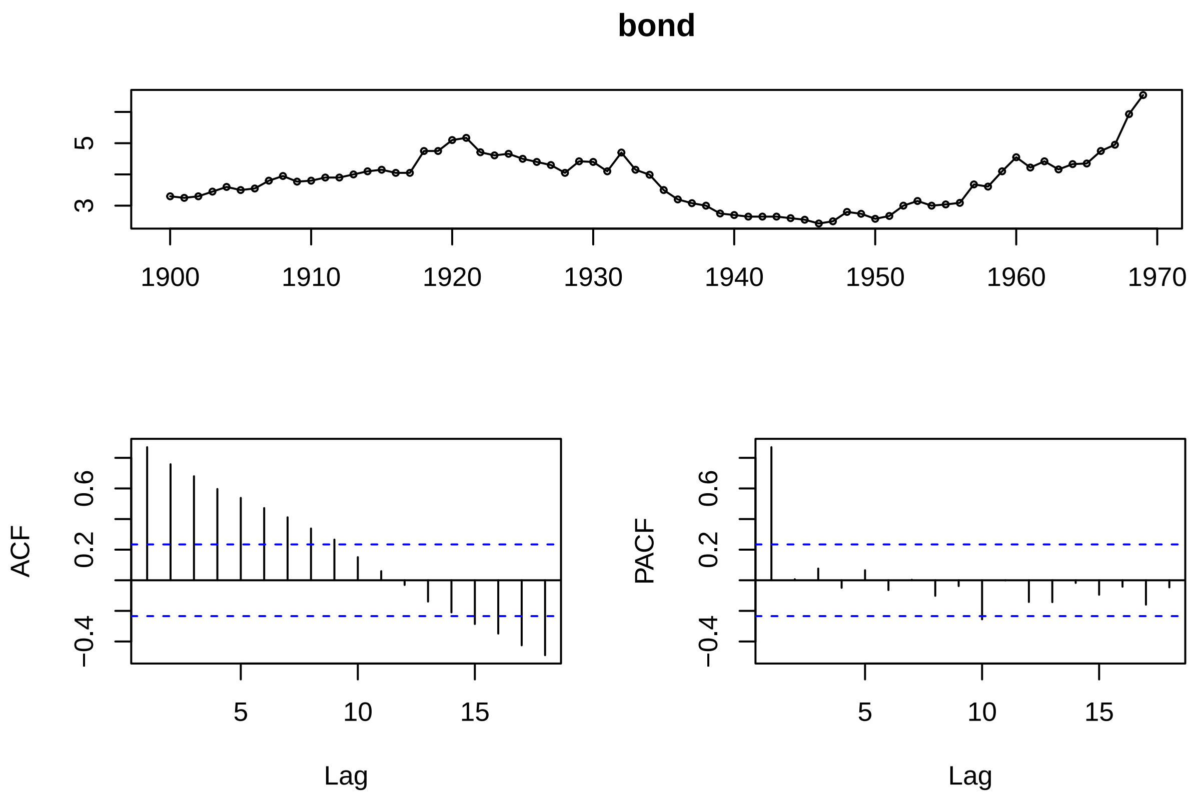
```
# summary of the data
summary(annual[, -1])
```

```
##    Bond.Yield      stock.price      Implicit.Price.Index Velocity.of.Money
##  Min.   :2.430   Min.   : 6.860   Min.   : 24.15       Min.   :1.160    
##  1st Qu.:3.083   1st Qu.: 8.825   1st Qu.: 39.53       1st Qu.:1.570    
##  Median :3.900   Median :12.265   Median : 50.93       Median :1.720    
##  Mean   :3.801   Mean   :25.443   Mean   : 60.67       Mean   :1.795    
##  3rd Qu.:4.388   3rd Qu.:25.698   3rd Qu.: 87.00       3rd Qu.:2.067    
##  Max.   :6.540   Max.   :98.700   Max.   :128.21       Max.   :2.490
```

- We have the five-number summaries for all the four variables, and the results shown are consistent with all the graphs above, especially the histograms and the boxplots.

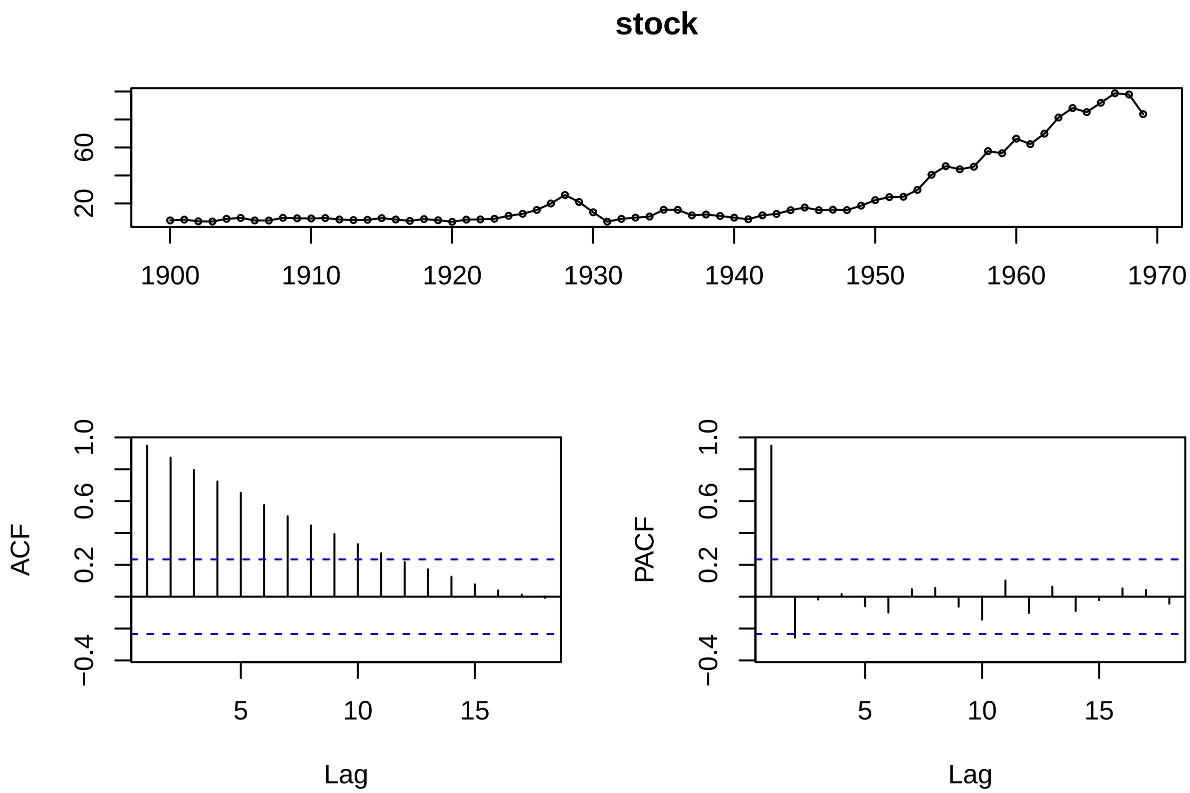
PART 2 Show the tsdisplay Plots and the ACF / PACF Plots.

```r
# convert the data into time series format
bond <- ts(annual$Bond.Yield, start = 1900, frequency = 1)
stock <- ts(annual$stock.price, start = 1900, frequency = 1)
price_index <- ts(annual$Implicit.Price.Index, start = 1900, frequency = 1)
velocity_money <- ts(annual$Velocity.of.Money, start = 1900, frequency = 1)

# the tsdisplay plots
tsdisplay(bond)
```



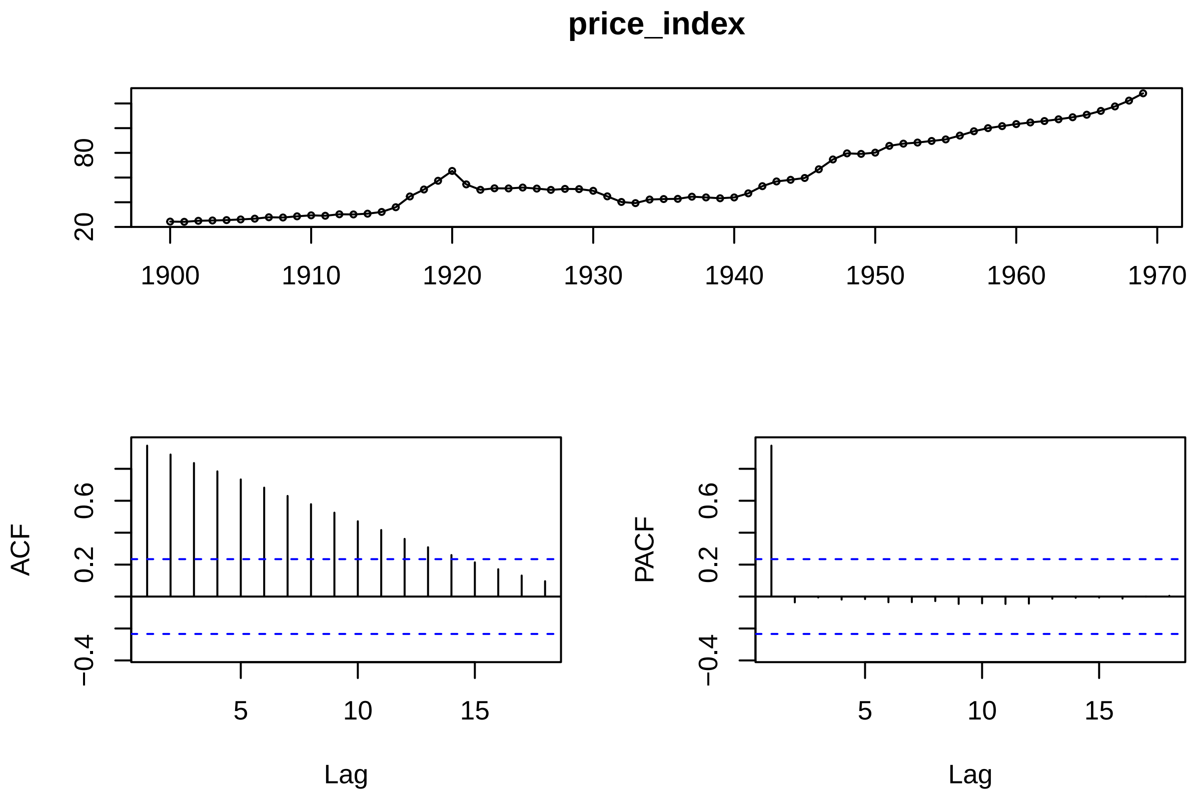
- The plots of bond yield are shown above. We can see that the mean is kind of stationary but the variance is certain not. We can also observe some cycle patterns and a slight upward trend. The ACF plot has decreasing spikes, and the PACF plot has only one strong spike at lag 1. They suggests an AR(1) model latter.

```r
tsdisplay(stock)
```



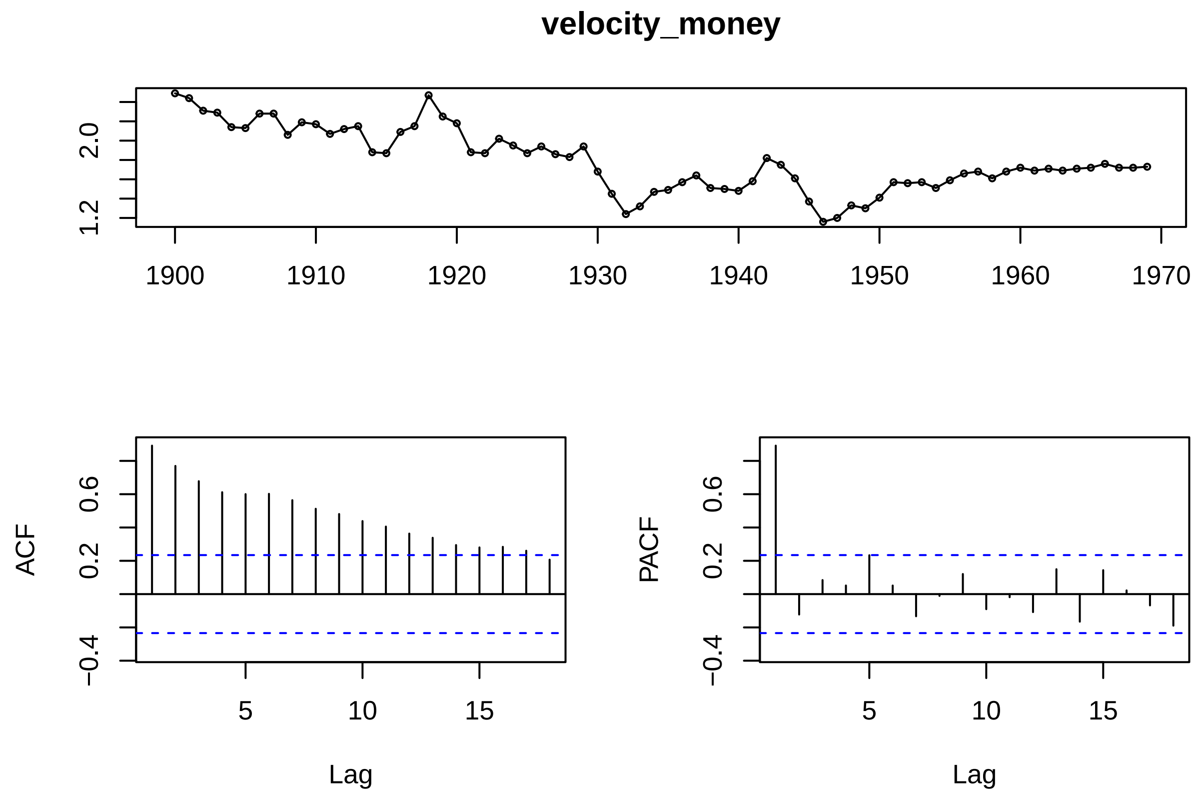
- The plots of stock price are shown above. We can observe an obvious upward trend from the data beginning 1940, and therefore it does not to be mean stationary nor variance stationary. The ACF plot has decreasing spikes towards zero, and the PACF plot has a strong spike at lag 1. According to the ACF and PACF plots, we should use AR(1) or AR(2) latter.

```
tsdisplay(price_index)
```



- The plots of price index are shown above. We observe an obvious upward trend from the data, and therefore the data is not mean or variance stationary. ACF plot shows decreasing spikes towards zero, and PACF plot has only 1 strong spike at lag 1. Again, this is a clear signal for an AR(1) model latter.

```
tsdisplay(velocity_money)
```



- The plots of velocity of money are shown above. We observe a slightly decreasing trend, and both the mean and variance changes through time so no mean or variance stationality. The ACF plot shows decreasing spikes towards zero, and PACF plot shows 1 strong spike at lag 1. According to the ACF and PACF plots, we should use AR(1) for the velocity of money.

PART 3. Fit two AR models to each variable and evaluate model performance.

```r
# for the Bond Yield
# based on the tsdisplay plots, we should fit an AR(1) for bond
bond_ar1 <- arma(bond, order = c(1, 0))
summary(bond_ar1)
```

Bond Yield

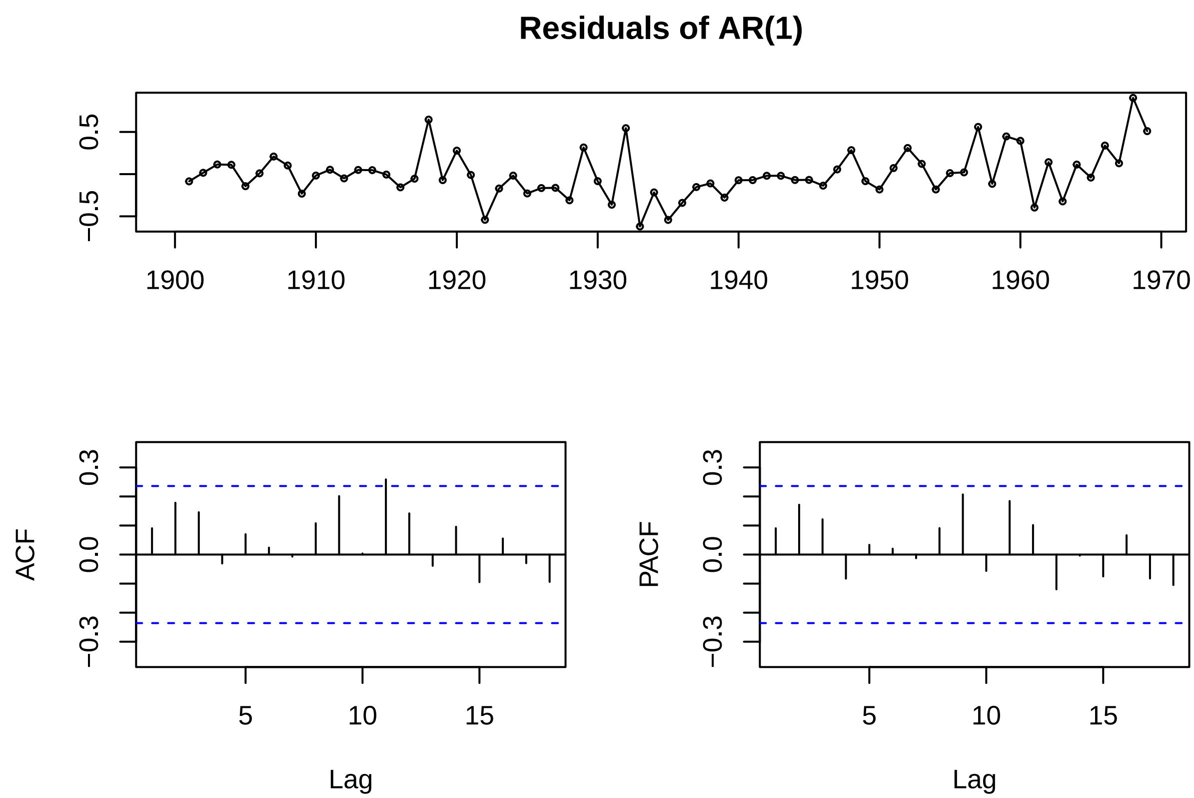
```
##
## Call:
## arma(x = bond, order = c(1, 0))
##
## Model:
## ARMA(1,0)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -0.62010 -0.15654 -0.01955  0.11322  0.90374
##
## Coefficient(s):
##            Estimate  Std. Error  t value Pr(>|t|)
## ar1         1.02466     0.04168   24.586   <2e-16 ***
## intercept  -0.04578     0.16018   -0.286    0.775
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 0.07665,  Conditional Sum-of-Squares = 5.21,  AIC = 22.85
```

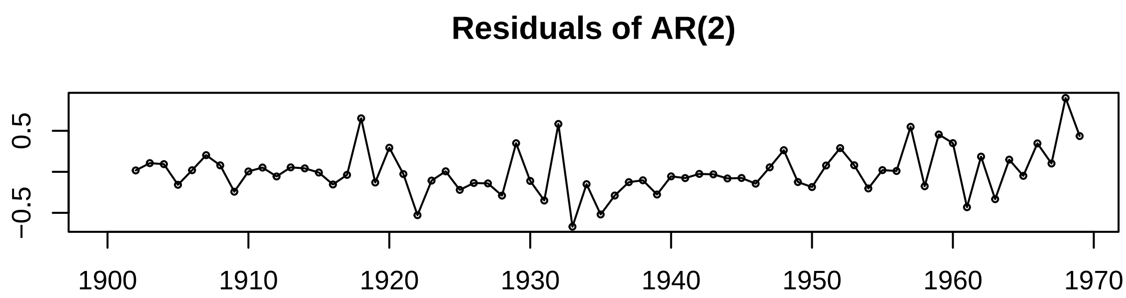
```r
# another possible model for bond is AR(2)
bond_ar2 <- arma(bond, order = c(2, 0))
summary(bond_ar2)
```

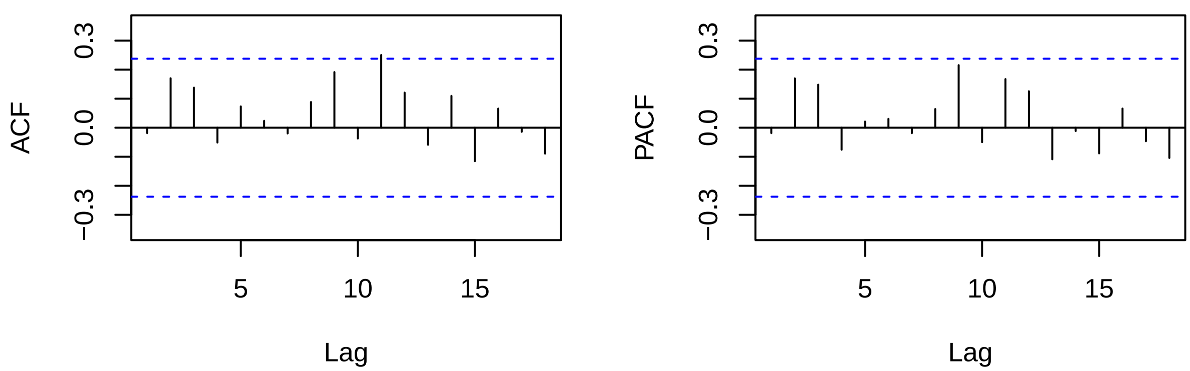
```
##
## Call:
## arma(x = bond, order = c(2, 0))
##
## Model:
## ARMA(2,0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66898 -0.14594 -0.02909  0.09564  0.90038
##
## Coefficient(s):
##            Estimate  Std. Error  t value Pr(>|t|)
## ar1        1.117881    0.122403    9.133   <2e-16 ***
## ar2       -0.105892    0.129513   -0.818    0.414
## intercept -0.000899    0.167403   -0.005    0.996
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 0.07694,  Conditional Sum-of-Squares = 5.16,  AIC = 25.12
```

```r
# plot the residual plots
tsdisplay(resid(bond_ar1), main = "Residuals of AR(1)")
```




```
tsdisplay(resid(bond_ar2), main = "Residuals of AR(2)")
```





- We fit AR(1) and AR(2) for the bond yield. Based on the summary, we found that the lag 2 in AR(2) is not statistically significant, indicating the term is redundant. Furthermore, both the ACF and PACF plots of AR(1) and AR(2) show no strong spikes, therefore they both resemble a white-noise process and thus the models are valid in that sense. Therefore, the residual plots of AR(2) model is not better than those of AR(1) model. Now we split the dataset for AR(1) model:

```
# split the data into training and testing
bond_tr <- bond[1:46]
bond_ts <- bond[47:70]

# fit AR(1) based on the training data
bond_tr1 <- arima(bond_tr, order = c(1, 0, 0))
# compute the training RMSE from the summary
summary(bond_tr1)
```

```
##
## Call:
## arima(x = bond_tr, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.9521     3.4025
## s.e.  0.0382     0.5574
##
## sigma^2 estimated as 0.05425:  log likelihood = 0.57,  aic = 4.87
```

```
##
## Training set error measures:
##                      ME     RMSE       MAE        MPE     MAPE     MASE
## Training set 0.00366517 0.232925 0.1663388 -0.3486646 4.242028 1.012888
##                     ACF1
## Training set 0.003767147
```

```r
# predict the testing data based on AR(1) model
bond_fc1 <- forecast(bond_tr1, h = 24)
# compute the testing RMSE
(bond_ts_rmse <- sqrt(mean((bond_ts - bond_fc1$mean)^2)))
```

```
## [1] 1.264111
```

- The training RMSE of AR(1) model is 0.232925, and the testing RMSE is 1.264111. They are overall consistent and the magnitude is not high. Then we test the AR(2) model:

```r
# fit AR(2) based on the training data
bond_tr2 <- arima(bond_tr, order = c(2, 0, 0))
# compute the training RMSE from the summary
summary(bond_tr2)
```

```
##
## Call:
## arima(x = bond_tr, order = c(2, 0, 0))
##
## Coefficients:
##          ar1     ar2  intercept
##       0.9356  0.0174     3.3947
## s.e.  0.1455  0.1475     0.5662
##
## sigma^2 estimated as 0.05424:  log likelihood = 0.57,  aic = 6.85
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE     MASE
## Training set 0.003461208 0.2328893 0.1673151 -0.3561147 4.267108 1.018833
##                    ACF1
## Training set 0.02257402
```

```r
# predict the testing data based on AR(1) model
bond_fc2 <- forecast(bond_tr2, h = 24)
# compute the testing RMSE
(bond_ts_rmse <- sqrt(mean((bond_ts - bond_fc2$mean)^2)))
```

```
## [1] 1.274338
```

- The training RMSE of AR(2) model is 0.2328893, and the testing RMSE is 1.274338.Therefore there is not too much of a difference in terms of traning or testing RMSE between the AR(1) and AR(2) model, though for testing RMSE AR(1) is slightly lower. We also check AIC and BIC here:

```r
AIC(bond_tr1, bond_tr2)
```

```
##          df      AIC
## bond_tr1  3 4.865386
## bond_tr2  4 6.851378
```

```r
BIC(bond_tr1, bond_tr2)
```

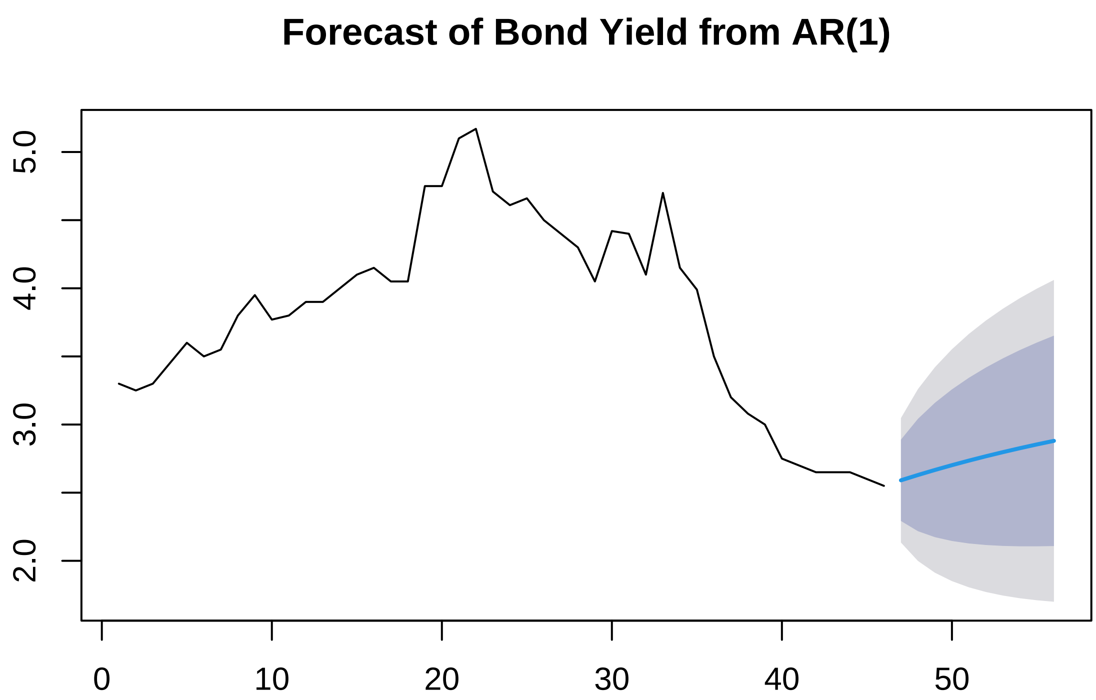
```
##          df      BIC
## bond_tr1  3 10.35131
## bond_tr2  4 14.16594
```

- For both AIC and BIC, the AR(1) model has lower value. In conclusion, since the lag2 term is insignificant, the testing RMSE for AR(1) is lower, and its AIC/BIC is lower, we conclude that AR(1) is the better model for bond yield.

```r
# the 10-step-ahead forecast for AR(1)
bond_fc1_10 <- forecast(bond_tr1, h = 10)
bond_fc1_10
```

```
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 47       2.590796 2.292291 2.889301 2.134272 3.047321
## 48       2.629640 2.217467 3.041813 1.999276 3.260004
## 49       2.666625 2.173552 3.159697 1.912535 3.420714
## 50       2.701839 2.145501 3.258178 1.850993 3.552686
## 51       2.735369 2.127337 3.343401 1.805465 3.665273
## 52       2.767294 2.115934 3.418654 1.771126 3.763463
## 53       2.797691 2.109405 3.485978 1.745048 3.850334
## 54       2.826634 2.106505 3.546762 1.725292 3.927976
## 55       2.854191 2.106365 3.602016 1.710490 3.997891
## 56       2.880429 2.108353 3.652506 1.699640 4.061219
```

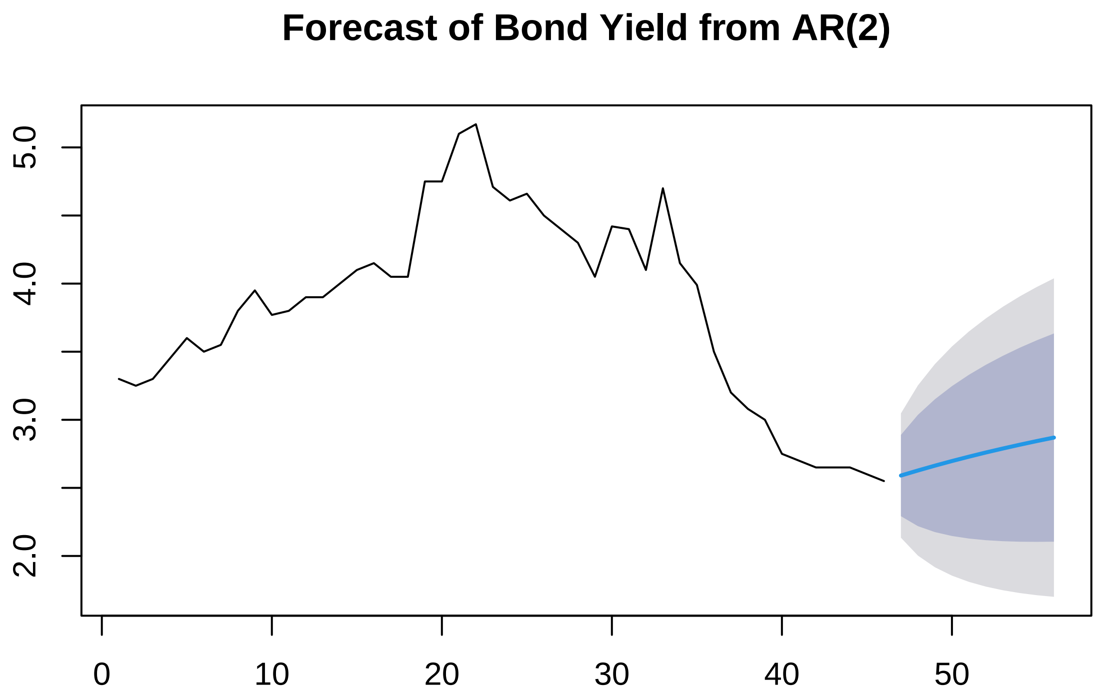
```r
plot(bond_fc1_10, main = "Forecast of Bond Yield from AR(1)")
```




```r
# the 10-step-ahead forecast for AR(2)
bond_fc2_10 <- forecast(bond_tr2, h = 10)
bond_fc2_10
```

```
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 47       2.590609 2.292150 2.889069 2.134155 3.047064
## 48       2.627732 2.219021 3.036443 2.002662 3.252802
## 49       2.663169 2.175290 3.151047 1.917022 3.409315
## 50       2.696967 2.146878 3.247056 1.855679 3.538256
## 51       2.729204 2.128088 3.330321 1.809876 3.648532
## 52       2.759952 2.115917 3.403986 1.774986 3.744918
## 53       2.789279 2.108546 3.470011 1.748189 3.830368
## 54       2.817250 2.104774 3.529727 1.727611 3.906889
## 55       2.843930 2.103756 3.584103 1.711932 3.975927
## 56       2.869376 2.104877 3.633875 1.700176 4.038577
```

```r
plot(bond_fc2_10, main = "Forecast of Bond Yield from AR(2)")
```



- The 10 step forecasts for both models are shown above, we've calculated the point forecasts and confidence intervals, then plotted them out.

```
# for the stock price
# based on the tsdisplay plots, we should fit an AR(1) for stock
stock_ar1 <- arma(stock, order = c(1,0))
summary(stock_ar1)
```

Stock Price.

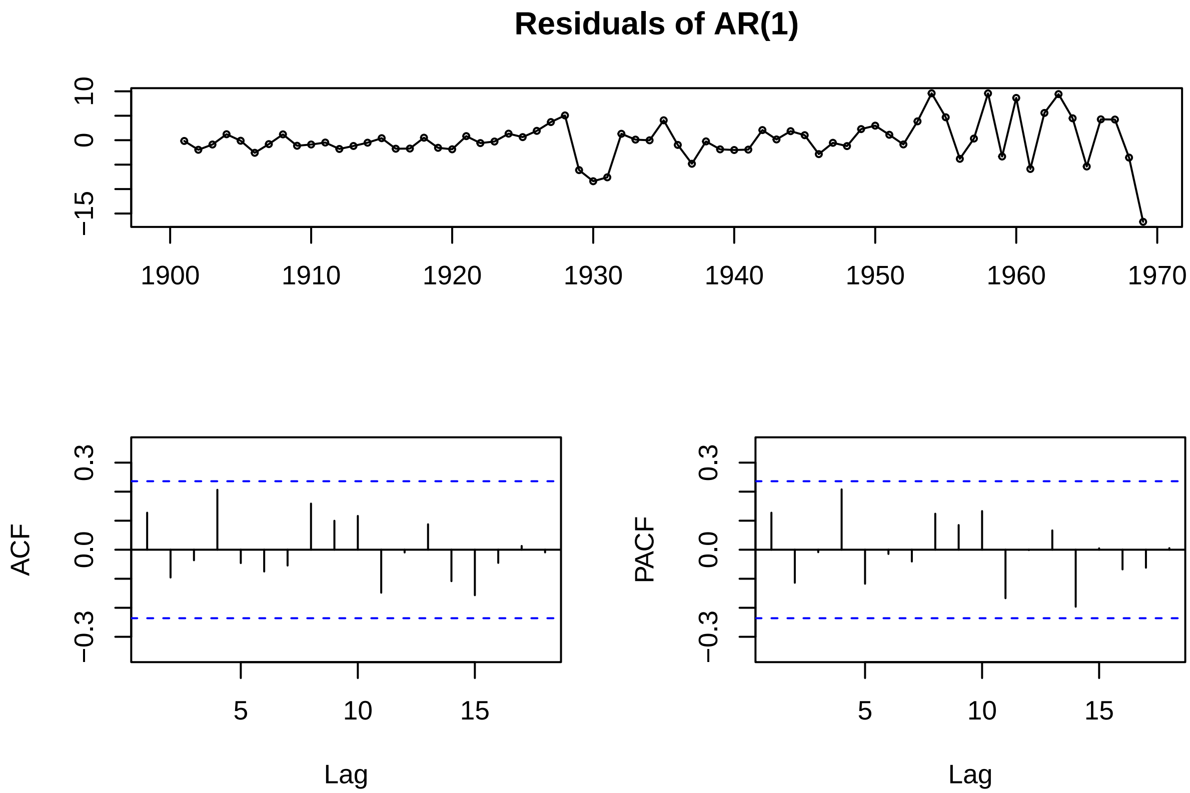
```
##
## Call:
## arma(x = stock, order = c(1, 0))
##
## Model:
## ARMA(1,0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.6973  -1.7964  -0.2486   1.8503   9.5896
##
## Coefficient(s):
```

```
##            Estimate  Std. Error  t value Pr(>|t|)
## ar1          1.02152     0.01942    52.60   <2e-16 ***
## intercept    0.57130     0.68840     0.83    0.407
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 17.45,  Conditional Sum-of-Squares = 1186.83,  AIC = 402.82
```

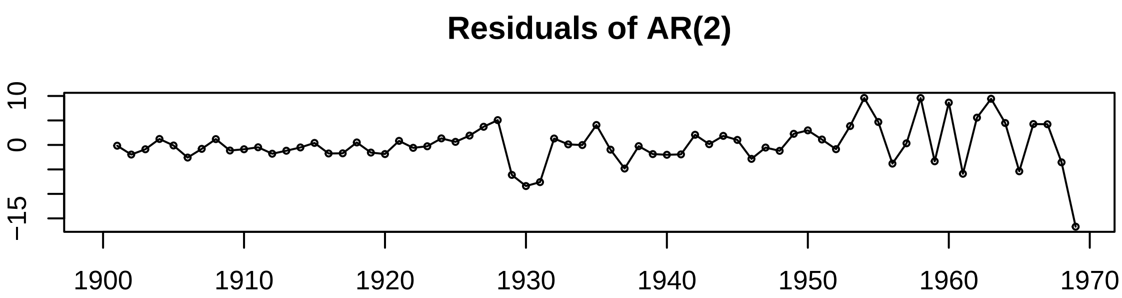
```r
# another possible model for stock is AR(2)
stock_ar2 <- arma(stock, order = c(1,0))
summary(stock_ar2)
```

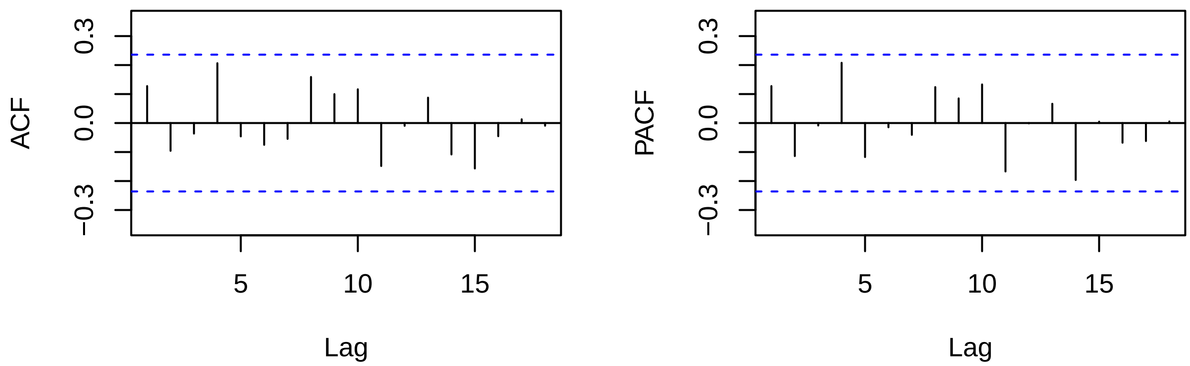
```
##
## Call:
## arma(x = stock, order = c(1, 0))
##
## Model:
## ARMA(1,0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.6973  -1.7964  -0.2486   1.8503   9.5896
##
## Coefficient(s):
##            Estimate  Std. Error  t value Pr(>|t|)
## ar1          1.02152     0.01942    52.60   <2e-16 ***
## intercept    0.57130     0.68840     0.83    0.407
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 17.45,  Conditional Sum-of-Squares = 1186.83,  AIC = 402.82
```

```r
# plot the residual plots
tsdisplay(resid(stock_ar1), main = "Residuals of AR(1)")
```




```r
tsdisplay(resid(stock_ar2), main = "Residuals of AR(2)")
```





- We fit AR(1) and AR(2) for the stock price. Based on the summary, we found that the lag 2 term of AR(2) is not significant and the residual plots of AR(2) model is not better than those of AR(1) model, since both have no significant spikes in either ACF or PACF plot and generally are two white noise processes.

```
# split the data into training and testing
stock_tr <- stock[1:46]
stock_ts <- stock[47:70]

# fit AR(1) based on the training data
stock_tr1 <- arima(stock_tr, order = c(1, 0, 0))
# compute the training RMSE from the summary
summary(stock_tr1)
```

```
##
## Call:
## arima(x = stock_tr, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.8047    11.0905
## s.e.  0.0874     1.6819
##
## sigma^2 estimated as 5.825:  log likelihood = -106.32,  aic = 218.64
##
```

```
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE     MASE      ACF1
## Training set 0.08557157 2.413476 1.718493 -3.070685 15.35847 0.964723 0.3897004
```

```r
# predict the testing data based on AR(1) model
stock_fc1 <- forecast(stock_tr1, h = 24)
# compute the testing RMSE
(stock_ts_rmse <- sqrt(mean((stock_ts - stock_fc1$mean)^2)))
```

```
## [1] 50.62308
```

- The training RMSE of AR(1) model is 2.413476, and the testing RMSE is 50.62308.

```r
# fit AR(2) based on the training data
stock_tr2 <- arima(stock_tr, order = c(2, 0, 0))
# compute the training RMSE from the summary
summary(stock_tr2)
```

```
##
## Call:
## arima(x = stock_tr, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       1.1919  -0.4877    10.9416
## s.e.  0.1275   0.1282     1.0321
##
## sigma^2 estimated as 4.412:  log likelihood = -100.2,  aic = 208.39
##
## Training set error measures:
##                      ME     RMSE      MAE       MPE     MAPE      MASE
## Training set 0.02670503 2.100555 1.605626 -2.731925 14.47256 0.9013618
##                    ACF1
## Training set 0.06737594
```

```r
# predict the testing data based on AR(2) model
stock_fc2 <- forecast(stock_tr2, h = 24)
# compute the testing RMSE
(stock_ts_rmse <- sqrt(mean((stock_ts - stock_fc2$mean)^2)))
```

```
## [1] 50.99426
```

- The training RMSE of AR(2) model is 2.100555, and the testing RMSE is 50.99426.

```r
AIC(stock_tr1, stock_tr2)
```

```
##           df      AIC
## stock_tr1  3 218.6434
## stock_tr2  4 208.3950
```

```r
BIC(stock_tr1, stock_tr2)
```

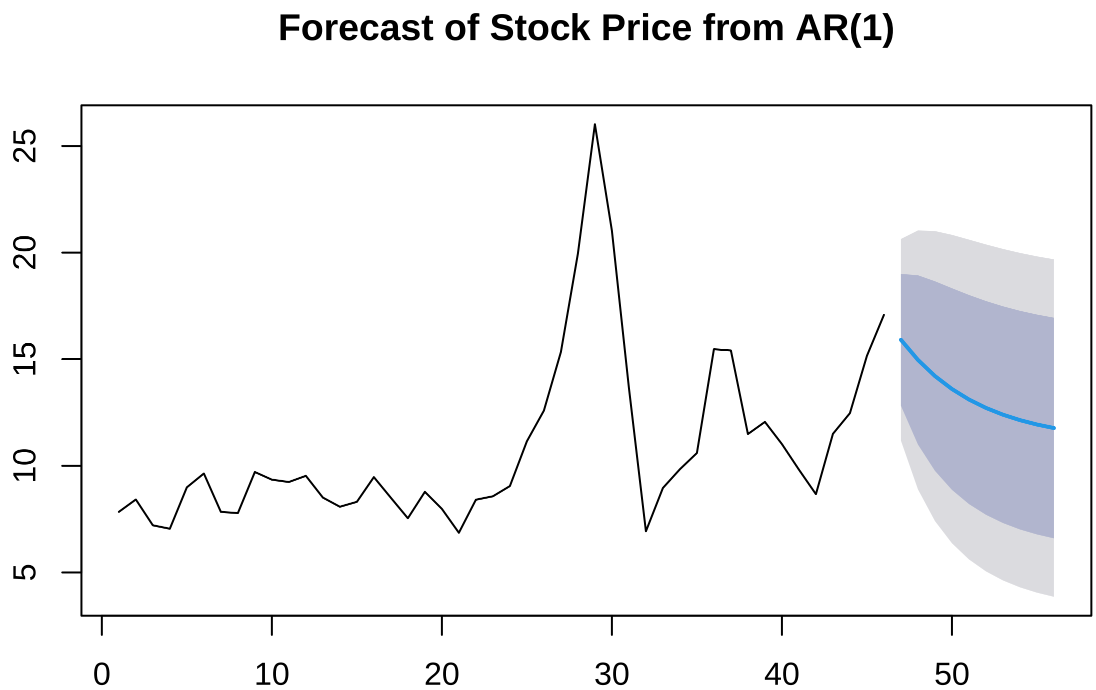
```
##           df      BIC
## stock_tr1  3 224.1293
## stock_tr2  4 215.7095
```

- We have mixed results: on one hand, the lag 2 term of AR(2) model is not statistically significant; on the other hand, AR(2)'s testing RMSE is almost the same as AR(1)'s testing RMSE; its residual plots are good, and it has generally lower AIC and BIC value. Therefore, we should pick AR(2) over AR(1).

```r
# the 10-step-ahead forecast for AR(1)
stock_fc1_10 <- forecast(stock_tr1, h = 10)
stock_fc1_10
```

```
##    Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## 47       15.91020 12.817206 19.00319 11.179873 20.64053
## 48       14.96887 10.998830 18.93892  8.897216 21.04053
## 49       14.21140  9.764777 18.65802  7.410878 21.01191
## 50       13.60186  8.872193 18.33153  6.368457 20.83527
## 51       13.11138  8.207132 18.01562  5.610981 20.61177
## 52       12.71669  7.702642 17.73074  5.048365 20.38501
## 53       12.39909  7.315205 17.48297  4.623961 20.17421
## 54       12.14351  7.014921 17.27211  4.300008 19.98702
## 55       11.93786  6.780520 17.09520  4.050389 19.82533
## 56       11.77237  6.596502 16.94824  3.856563 19.68818
```

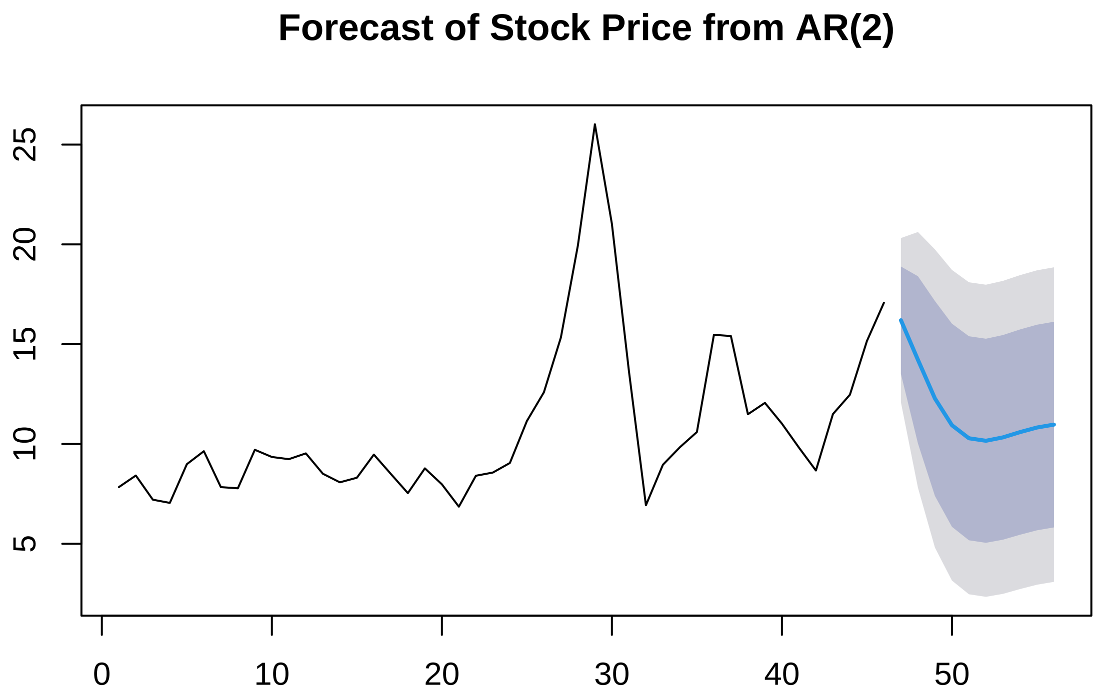
```r
plot(stock_fc1_10, main = "Forecast of Stock Price from AR(1)")
```




```r
# the 10-step-ahead forecast for AR(2)
stock_fc2_10 <- forecast(stock_tr2, h = 10)
stock_fc2_10
```

```
##    Point Forecast      Lo 80    Hi 80     Lo 95    Hi 95
## 47       16.20074 13.508768 18.89271 12.083726 20.31775
## 48       14.21640 10.028135 18.40466  7.811002 20.62180
## 49       12.28006  7.396515 17.16361  4.811321 19.74881
## 50       10.93986  5.851624 16.02810  3.158073 18.72165
## 51       10.28678  5.176138 15.39743  2.470725 18.10284
## 52       10.16197  5.049745 15.27419  2.343498 17.98044
## 53       10.33169  5.205033 15.45835  2.491145 18.17223
## 54       10.59485  5.452902 15.73681  2.730917 18.45879
## 55       10.82575  5.675911 15.97559  2.949752 18.70175
## 56       10.97261  5.820699 16.12453  3.093440 18.85179
```

```r
plot(stock_fc2_10, main = "Forecast of Stock Price from AR(2)")
```



- The point forecast of two models, along with confidence interval and graphs are shown above. We can see that the forecast from AR(2) is different from forecast from AR(1).

```
# for the implicit price index
# based on the tsdisplay plots, we should fit an AR(1) for price index
price_index_ar1 <- arma(price_index, order = c(1, 0))
summary(price_index_ar1)
```

Implicit Price Index.

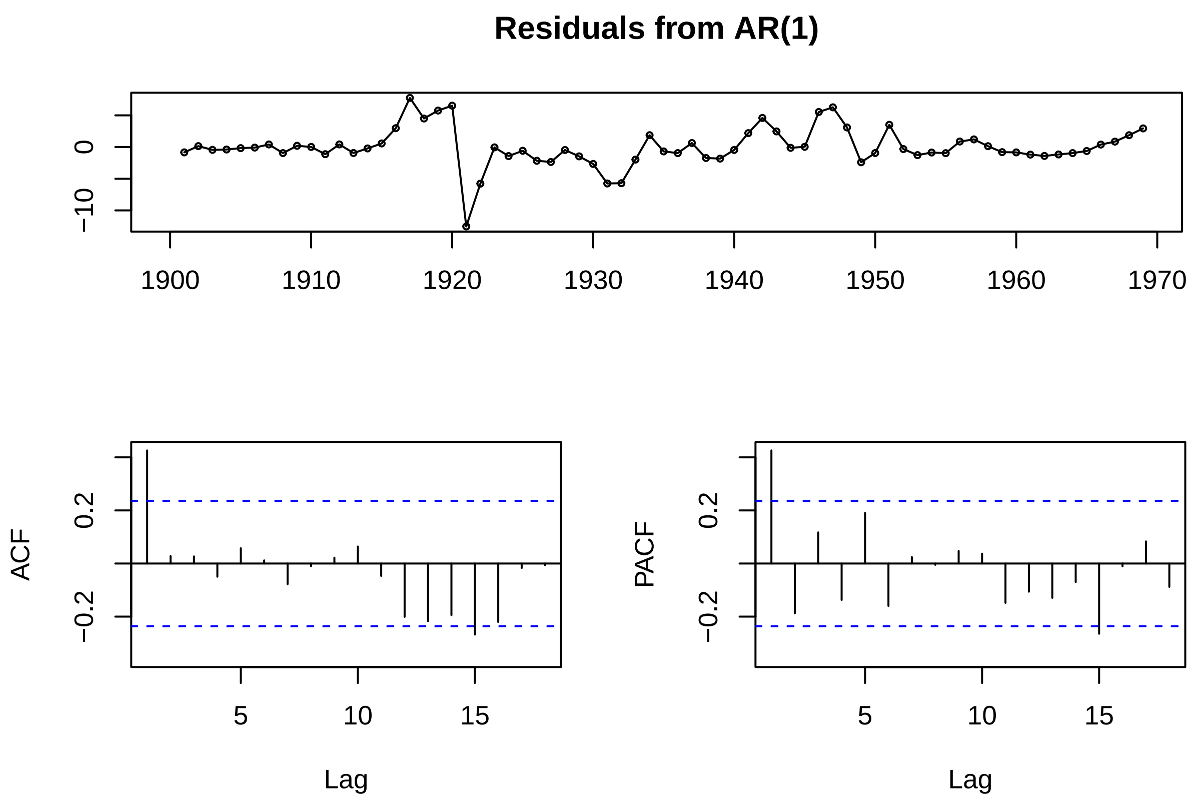
```
##
## Call:
## arma(x = price_index, order = c(1, 0))
##
## Model:
## ARMA(1,0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.5351  -1.1320  -0.3905   0.8568   7.7595
##
## Coefficient(s):
```

```
##              Estimate  Std. Error  t value Pr(>|t|)
## ar1          1.02336     0.01254   81.625   <2e-16 ***
## intercept    0.11142     0.82984    0.134    0.893
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 9.129,  Conditional Sum-of-Squares = 620.74,  AIC = 357.45
```

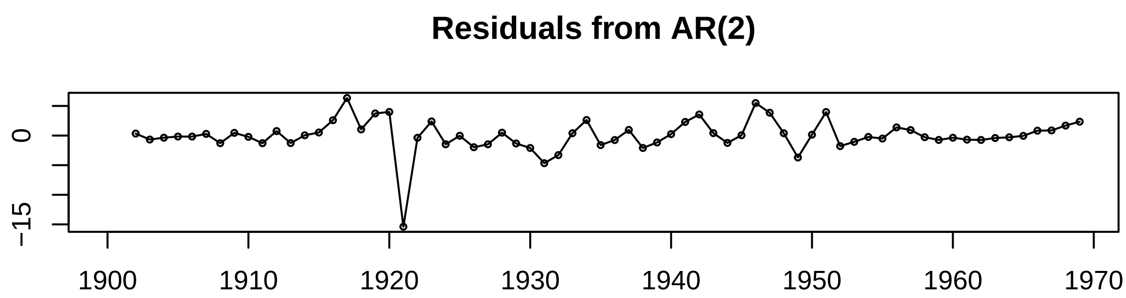
```
# another possible model for bond is AR(2)
price_index_ar2 <- arma(price_index, order = c(2, 0))
summary(price_index_ar2)
```

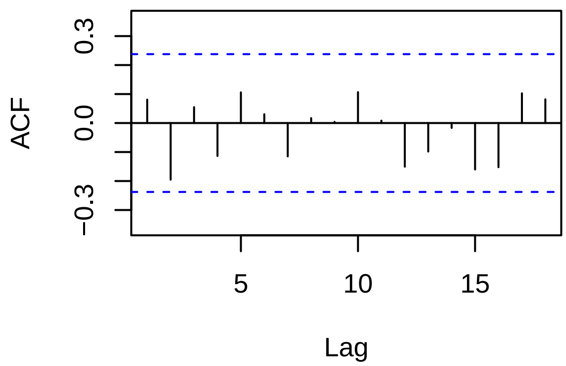
```
##
## Call:
## arma(x = price_index, order = c(2, 0))
##
## Model:
## ARMA(2,0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.3851  -1.0889  -0.1103   0.8799   6.3418
##
## Coefficient(s):
##              Estimate  Std. Error  t value Pr(>|t|)
## ar1           1.4543      0.1086   13.389  < 2e-16 ***
## ar2          -0.4450      0.1114   -3.995 6.47e-05 ***
## intercept     0.3268      0.7629    0.428    0.668
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 7.535,  Conditional Sum-of-Squares = 504.86,  AIC = 346.02
```

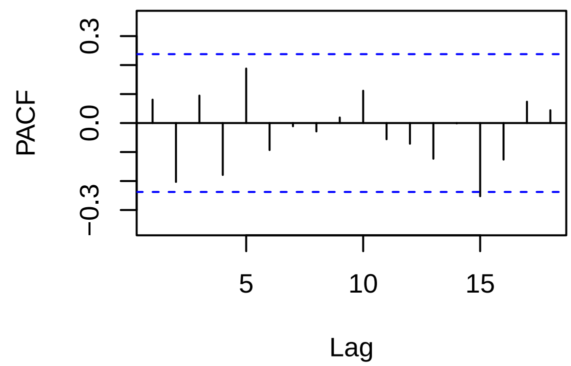
```
# plot the residual plots
tsdisplay(resid(price_index_ar1), main = "Residuals from AR(1)")
```




```r
tsdisplay(resid(price_index_ar2), main = "Residuals from AR(2)")
```







- We fit AR(1) and AR(2) for the implicit price index. Based on the summary, we found that all terms in AR(1) and AR(2). The residual plots of AR(1) shows a strong spike at lag 1 in ACF and PACF, implying some leftover dynamics in the residuals. But the residual plots of AR(2) is a standard white noise process with no strong spikes anywhere.

```
# split the data into training and testing
price_index_tr <- price_index[1:46]
price_index_ts <- price_index[47:70]

# fit AR(1) based on the training data
price_index_tr1 <- arima(price_index_tr, order = c(1, 0, 0))
# compute the training RMSE from the summary
summary(price_index_tr1)
```

```
##
## Call:
## arima(x = price_index_tr, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.9732    41.8202
## s.e.  0.0292    11.4640
##
## sigma^2 estimated as 11.17:  log likelihood = -122.24,  aic = 250.49
##
```

```
## Training set error measures:
##                     ME     RMSE      MAE      MPE     MAPE      MASE      ACF1
## Training set 0.6620773 3.341899 2.156533 1.096709 4.751414 0.9989087 0.4007648
```

```r
# predict the testing data based on AR(1) model
price_index_fc1 <- forecast(price_index_tr1, h = 24)
# compute the testing RMSE
(price_index_ts_rmse <- sqrt(mean((price_index_ts - price_index_fc1$mean)^2)))
```

```
## [1] 46.20935
```

- The training RMSE of AR(1) model is 3.341899, and the testing RMSE is 46.20935.

```r
# fit AR(2) based on the training data
price_index_tr2 <- arima(price_index_tr, order = c(2, 0, 0))
# compute the training RMSE from the summary
summary(price_index_tr2)
```

```
##
## Call:
## arima(x = price_index_tr, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       1.4131  -0.4640    41.8821
## s.e.  0.1283   0.1334     7.2412
##
## sigma^2 estimated as 8.821:  log likelihood = -116.93,  aic = 241.86
##
## Training set error measures:
##                     ME    RMSE      MAE       MPE     MAPE      MASE       ACF1
## Training set 0.3079404 2.97008 1.844907 0.2093397 4.226491 0.8545633 0.01824014
```

```r
# predict the testing data based on AR(1) model
price_index_fc2 <- forecast(price_index_tr2, h = 24)
# compute the testing RMSE
(price_index_ts_rmse <- sqrt(mean((price_index_ts - price_index_fc2$mean)^2)))
```

```
## [1] 52.45451
```

- The training RMSE of AR(2) model is 2.97008, and the testing RMSE is 52.45451.

```r
AIC(price_index_tr1, price_index_tr2)
```

```
##                 df      AIC
## price_index_tr1  3 250.4854
## price_index_tr2  4 241.8615
```

```r
BIC(price_index_tr1, price_index_tr2)
```

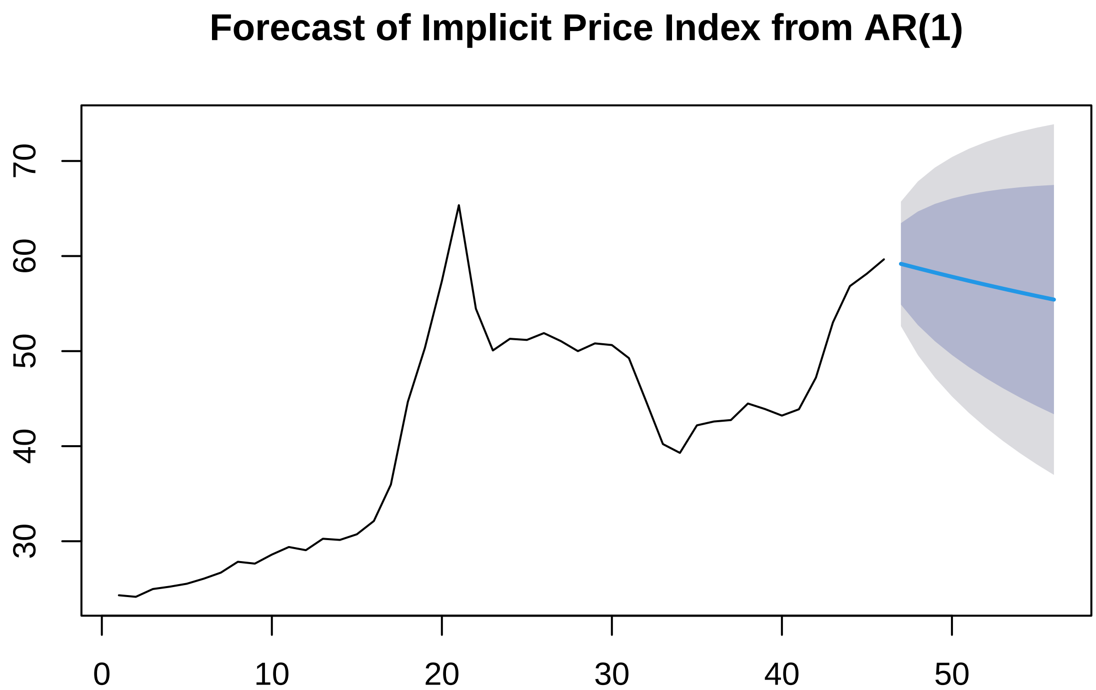
```
##                 df      BIC
## price_index_tr1  3 255.9713
## price_index_tr2  4 249.1761
```

- First of all, the AR(2) model's residual is a standard white noise process while the AR(1)'s is not. Despite a little higher testing RMSE for AR(2), it also has a lower AIC and BIC than AR(1). Therefore, AR(2) is overall the better model for implicit price index. Next we forecast 10 step ahead for both models:

```r
# the 10-step-ahead forecast for AR(1)
price_index_fc1_10 <- forecast(price_index_tr1, h = 10)
price_index_fc1_10
```

```
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 47       59.17982 54.89701 63.46264 52.62982 65.72983
## 48       58.71542 52.73907 64.69176 49.57538 67.85545
## 49       58.26343 51.04029 65.48657 47.21658 69.31028
## 50       57.82354 49.59176 66.05531 45.23412 70.41295
## 51       57.39541 48.31094 66.47988 43.50192 71.28890
## 52       56.97874 47.15461 66.80287 41.95403 72.00345
## 53       56.57321 46.09651 67.04991 40.55049 72.59594
## 54       56.17854 45.11917 67.23790 39.26470 73.09237
## 55       55.79442 44.21014 67.37869 38.07780 73.51104
## 56       55.42057 43.36015 67.48100 36.97574 73.86541
```

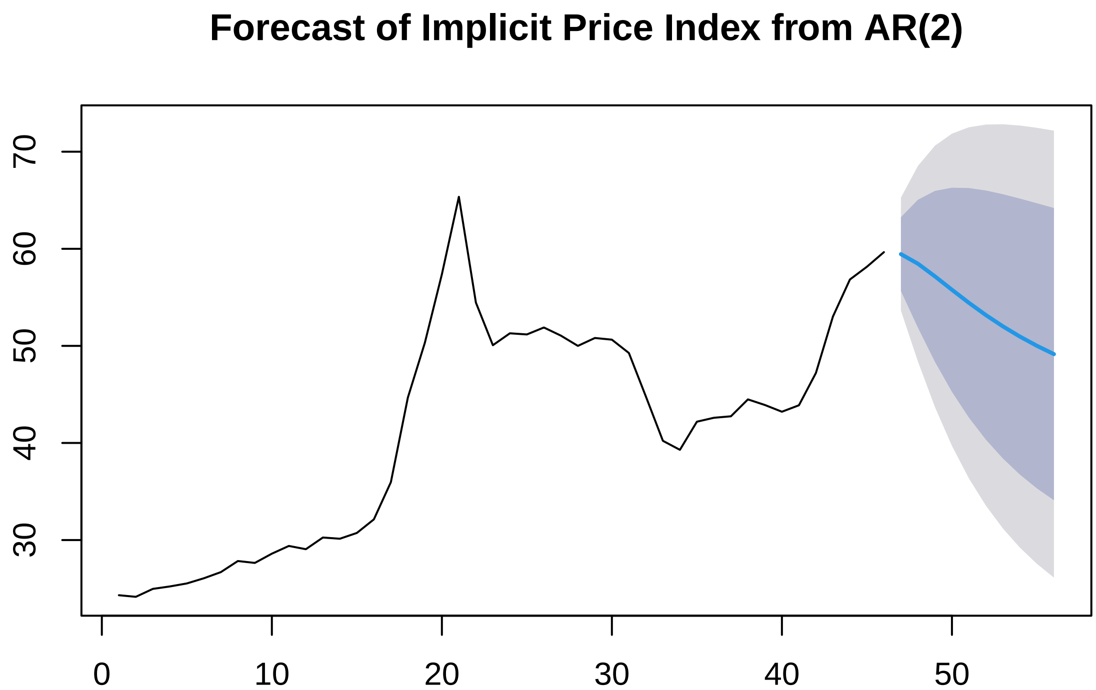
```r
plot(price_index_fc1_10, main = "Forecast of Implicit Price Index from AR(1)")
```




```
# the 10-step-ahead forecast for AR(2)
price_index_fc2_10 <- forecast(price_index_tr2, h = 10)
price_index_fc2_10
```

```
##    Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 47       59.45170 55.64539 63.25801 53.63045 65.27295
## 48       58.46284 51.87355 65.05214 48.38539 68.54030
## 49       57.16074 48.35944 65.96204 43.70031 70.62116
## 50       55.77951 45.26666 66.29237 39.70149 71.85754
## 51       54.43184 42.60516 66.25852 36.34449 72.51918
## 52       53.16827 40.33274 66.00379 33.53802 72.79851
## 53       52.00799 38.39486 65.62111 31.18850 72.82747
## 54       50.95464 36.73917 65.17011 29.21395 72.69533
## 55       50.00448 35.32005 64.68892 27.54658 72.46239
## 56       49.15053 34.09927 64.20179 26.13162 72.16944
```

```
plot(price_index_fc2_10, main = "Forecast of Implicit Price Index from AR(2)")
```



- The calculated forecasts and graphs are shown above.

```r
# for the velocity of money
# based on the tsdisplay plots, we should fit an AR(1) for the velocity of money
velocity_money_ar1 <- arma(velocity_money, order = c(1, 0))
summary(velocity_money_ar1)
```

Velocity of Money

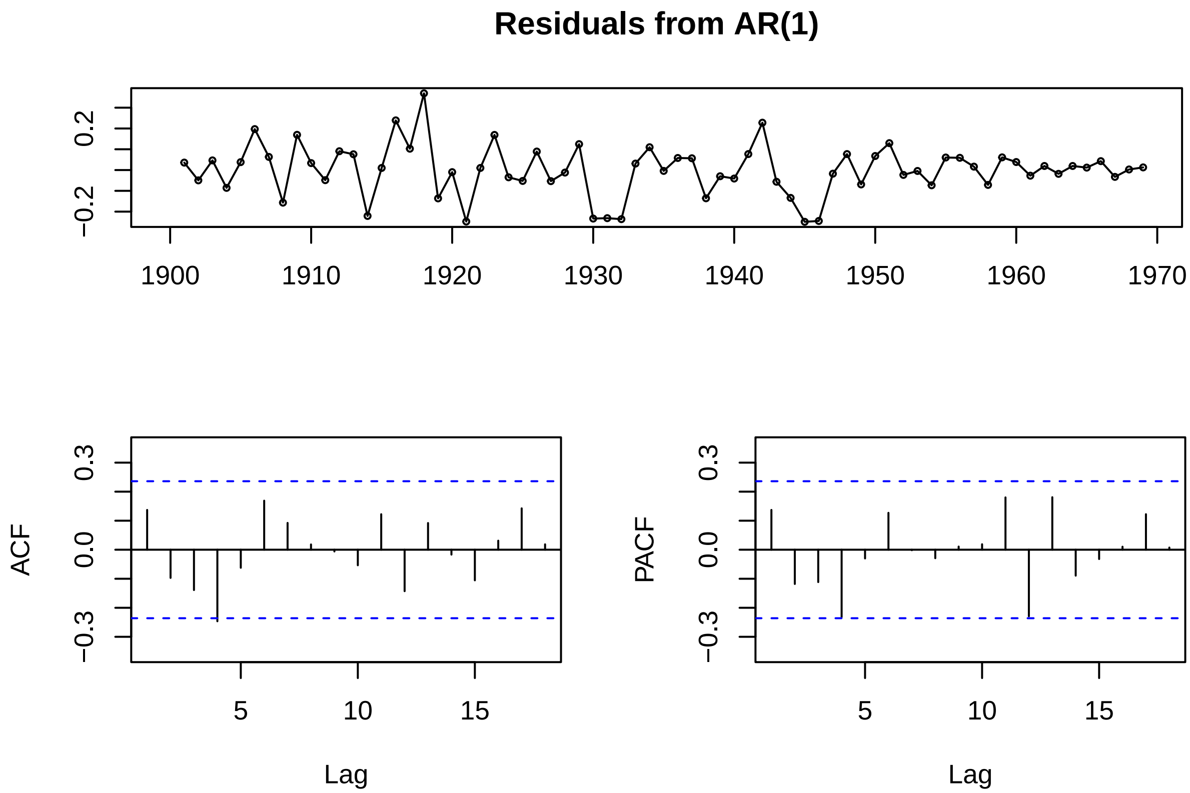
```
##
## Call:
## arma(x = velocity_money, order = c(1, 0))
##
## Model:
## ARMA(1,0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.24903 -0.05239  0.01007  0.06097  0.36917
##
## Coefficient(s):
##            Estimate  Std. Error  t value Pr(>|t|)
```

```
## ar1          0.89223     0.04419   20.192   <2e-16 ***
## intercept   0.18255     0.08063    2.264   0.0236 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 0.01451,  Conditional Sum-of-Squares = 0.99,  AIC = -93.66
```

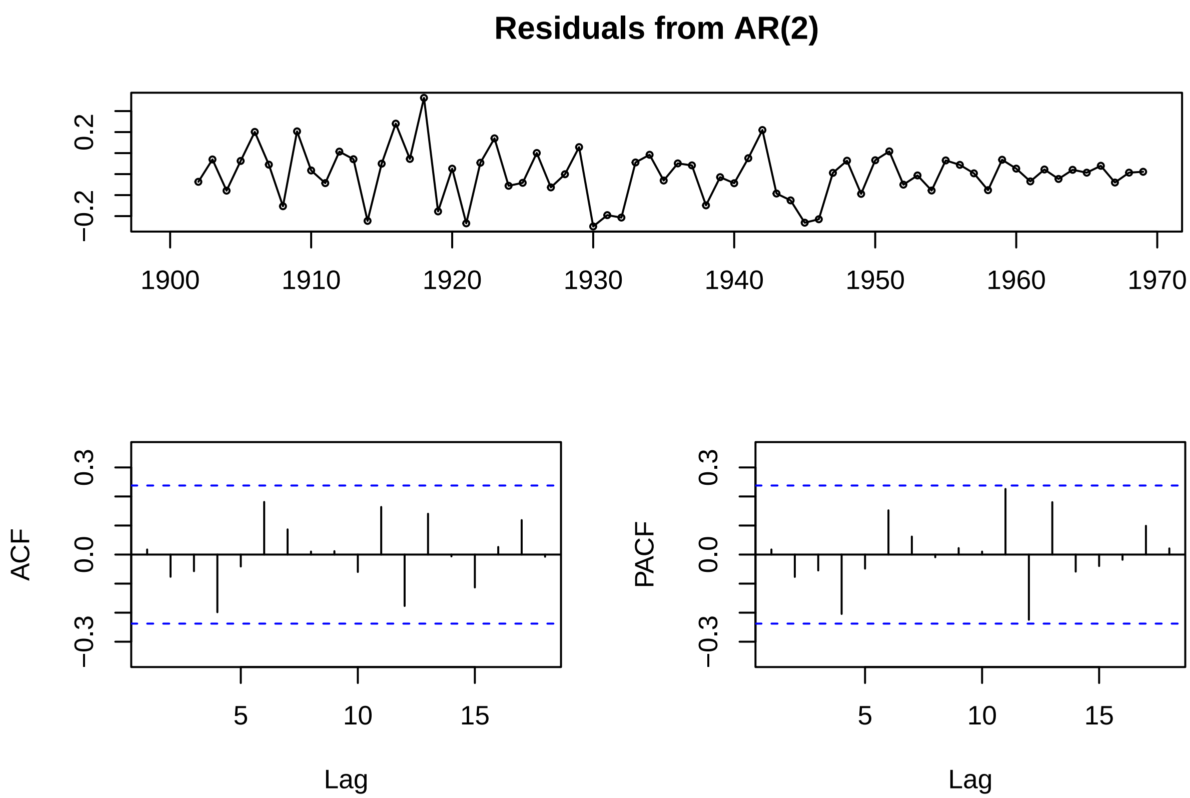
```
# another possible model for bond is AR(2)
velocity_money_ar2 <- arma(velocity_money, order = c(2, 0))
summary(velocity_money_ar2)
```

```
##
## Call:
## arma(x = velocity_money, order = c(2, 0))
##
## Model:
## ARMA(2,0)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.249221 -0.057588  0.008971  0.065294  0.362911
##
## Coefficient(s):
##            Estimate  Std. Error  t value Pr(>|t|)
## ar1         1.02929     0.11805    8.719   <2e-16 ***
## ar2        -0.14724     0.11403   -1.291   0.1966
## intercept   0.20185     0.08258    2.444   0.0145 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Fit:
## sigma^2 estimated as 0.01436,  Conditional Sum-of-Squares = 0.96,  AIC = -92.37
```

```
# plot the residual plots
tsdisplay(resid(velocity_money_ar1), main = "Residuals from AR(1)")
```




```
tsdisplay(resid(velocity_money_ar2), main = "Residuals from AR(2)")
```



- We fit AR(1) and AR(2) for the velocity of money. Based on the summary, we found that the lag 2 term in AR(2) is not significant. For residual plots, both models demonstrate standard white noise processes since there are no spikes in ACF or PACF plots. Thus the residual plots of AR(2) model is not better than those of AR(1) model.

```
# split the data into training and testing
velocity_money_tr <- velocity_money[1:46]
velocity_money_ts <- velocity_money[47:70]

# fit AR(1) based on the training data
velocity_money_tr1 <- arima(velocity_money_tr, order = c(1, 0, 0))
# compute the training RMSE from the summary
summary(velocity_money_tr1)
```

```
##
## Call:
## arima(x = velocity_money_tr, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
##       0.9296     1.9145
## s.e.  0.0571     0.2392
##
## sigma^2 estimated as 0.02052:  log likelihood = 23.12,  aic = -40.23
##
```

```
## Training set error measures:
##                       ME      RMSE       MAE       MPE     MAPE      MASE
## Training set -0.01953879 0.1432464 0.1120792 -1.606729 6.145759 0.9736607
##                    ACF1
## Training set 0.07098079
```

```r
# predict the testing data based on AR(1) model
velocity_money_fc1 <- forecast(velocity_money_tr1, h = 24)
# compute the testing RMSE
(velocity_money_ts_rmse <- sqrt(mean((velocity_money_ts - velocity_money_fc1$mean)^2)))
```

```
## [1] 0.1079826
```

- The training RMSE of AR(1) model is 0.1432464, and the testing RMSE is 0.1079826.

```r
# fit AR(2) based on the training data
velocity_money_tr2 <- arima(velocity_money_tr, order = c(2, 0, 0))
# compute the training RMSE from the summary
summary(velocity_money_tr2)
```

```
##
## Call:
## arima(x = velocity_money_tr, order = c(2, 0, 0))
##
## Coefficients:
##          ar1      ar2  intercept
##       1.0506  -0.1351     1.9078
## s.e.  0.1495   0.1542     0.2097
##
## sigma^2 estimated as 0.02019:  log likelihood = 23.5,  aic = -39
##
## Training set error measures:
##                       ME      RMSE       MAE       MPE     MAPE      MASE
## Training set -0.01635156 0.1420907 0.1125857 -1.431477 6.142057 0.9780609
##                     ACF1
## Training set -0.02599652
```

```r
# predict the testing data based on AR(1) model
velocity_money_fc2 <- forecast(velocity_money_tr2, h = 24)
# compute the testing RMSE
(velocity_money_ts_rmse <- sqrt(mean((velocity_money_ts - velocity_money_fc2$mean)^2)))
```

```
## [1] 0.1302476
```

- The training RMSE of AR(1) model is 0.1217399, and the testing RMSE is 0.1302476. It is worth noticing that for both AR(1) and AR(2) model on training data, the training RMSE is larger than testing RMSE. We believe this is because the training dataset might be more noisy than the testing dataset.

```r
AIC(velocity_money_tr1, velocity_money_tr2)
```

```
##                    df       AIC
## velocity_money_tr1  3 -40.23469
## velocity_money_tr2  4 -38.99632
```

```r
BIC(velocity_money_tr1, velocity_money_tr2)
```

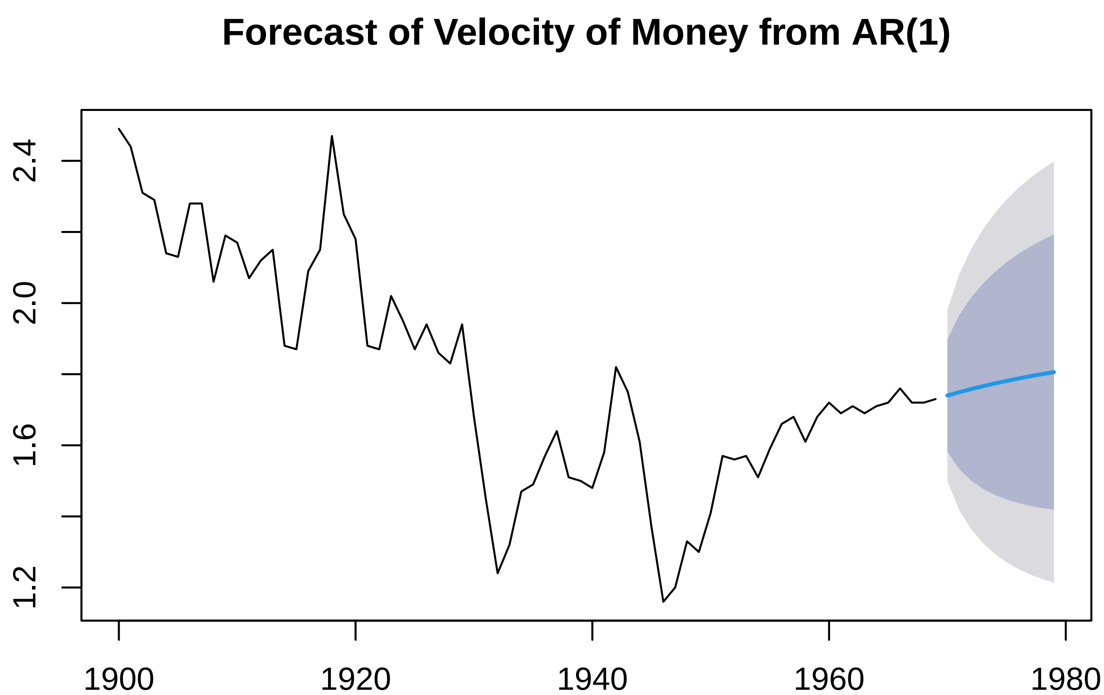
```
##                    df       BIC
## velocity_money_tr1  3 -34.74877
## velocity_money_tr2  4 -31.68175
```

- In conclusion, since the AR(2) model has one insignificant term, larger testing RMSE, larger AIC and BIC values, while both models' residual plots look great, we believe an AR(1) model is more suitable for the data.

```r
# the 10-step-ahead forecast for AR(1)
velocity_money_ar1 <- arima(velocity_money, order = c(1,0,0))
velocity_money_ar2 <- arima(velocity_money, order = c(2,0,0))
velocity_money_fc1_10 <- forecast(velocity_money_ar1, h = 10)
velocity_money_fc1_10
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 1970       1.739950 1.581906 1.897994 1.498242 1.981657
## 1971       1.749280 1.532616 1.965944 1.417921 2.080639
## 1972       1.758029 1.500621 2.015438 1.364357 2.151702
## 1973       1.766234 1.477711 2.054757 1.324977 2.207492
## 1974       1.773929 1.460587 2.087270 1.294714 2.253143
## 1975       1.781144 1.447499 2.114789 1.270878 2.291410
## 1976       1.787910 1.437381 2.138439 1.251823 2.323997
## 1977       1.794255 1.429524 2.158986 1.236447 2.352063
## 1978       1.800205 1.423427 2.176983 1.223973 2.376438
## 1979       1.805785 1.418723 2.192848 1.213824 2.397746
```

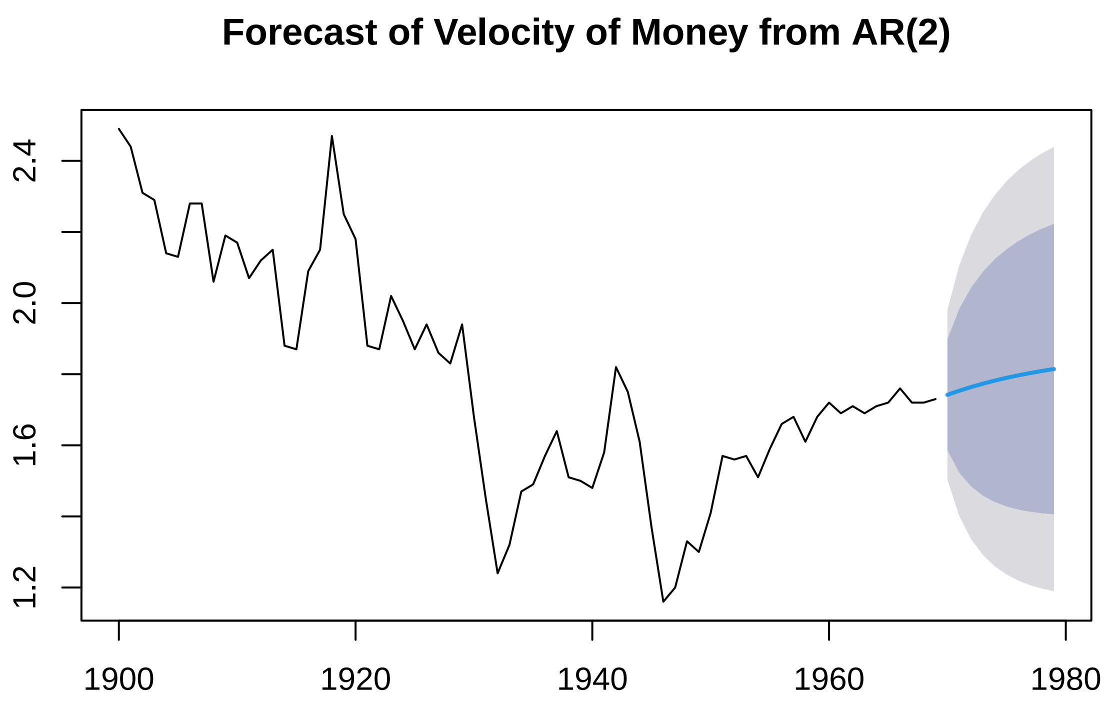
```r
plot(velocity_money_fc1_10, main = "Forecast of Velocity of Money from AR(1)")
```




```r
# the 10-step-ahead forecast for AR(2)
velocity_money_fc2_10 <- forecast(velocity_money_ar2, h = 10)
velocity_money_fc2_10
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 1970       1.741988 1.585972 1.898004 1.503382 1.980594
## 1971       1.753387 1.523085 1.983688 1.401171 2.105602
## 1972       1.763824 1.484139 2.043509 1.336082 2.191565
## 1973       1.773312 1.458318 2.088307 1.291569 2.255056
## 1974       1.781927 1.440521 2.123332 1.259792 2.304061
## 1975       1.789745 1.428024 2.151466 1.236540 2.342949
## 1976       1.796840 1.419205 2.174475 1.219298 2.374382
## 1977       1.803279 1.413024 2.193534 1.206436 2.400122
## 1978       1.809123 1.408773 2.209473 1.196840 2.421405
## 1979       1.814426 1.405949 2.222903 1.189714 2.439138
```

```r
plot(velocity_money_fc2_10, main = "Forecast of Velocity of Money from AR(2)")
```



- The calculated forecasts and plots are shown above.

PART 4

```r
## Part 4
library(dLagM)
```

```
## Loading required package: nardl

## Loading required package: dynlm

##
## Attaching package: 'dLagM'

## The following object is masked from 'package:forecast':
##
##     forecast
```

```r
# using stock price to predict bond.yield
# ARDL(1,1)
data <- as.data.frame(data)
ardl_bond_stock1 <- ardlDlm(Bond.Yield ~ stock.price, data = data, p = 1, q = 1)
summary(ardl_bond_stock1)
```

```
##
## Time series regression with "ts" data:
## Start = 2, End = 70
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.60470 -0.11845 -0.03698  0.12075  0.74592
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.062973   0.155068   0.406   0.6860
## stock.price.t -0.010635   0.007582  -1.403   0.1655
## stock.price.1  0.015443   0.007927   1.948   0.0557 .
## Bond.Yield.1   0.967419   0.042085  22.987   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2565 on 65 degrees of freedom
## Multiple R-squared:  0.9149, Adjusted R-squared:  0.9109
## F-statistic: 232.8 on 3 and 65 DF,  p-value: < 2.2e-16
```

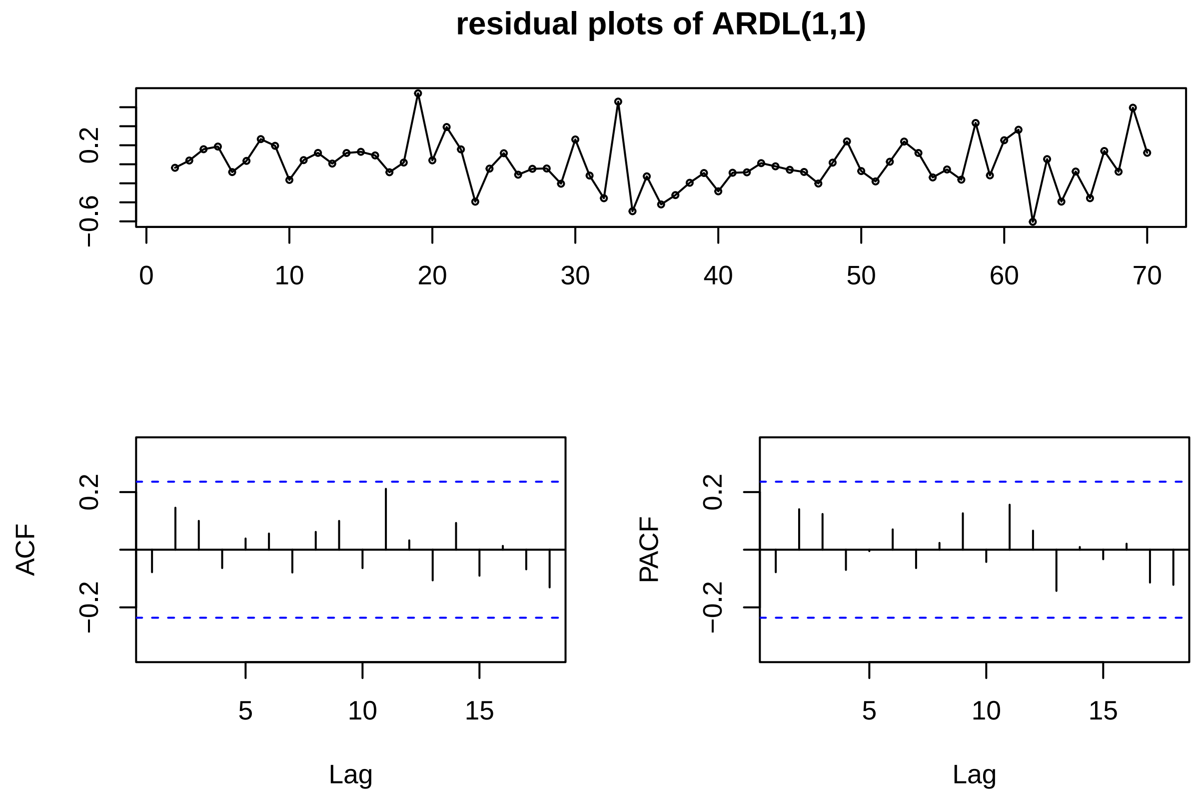
```r
# ARDL(2,2)
ardl_bond_stock2 <- ardlDlm(Bond.Yield ~ stock.price, data = data, p = 2, q = 2)
summary(ardl_bond_stock2)
```

```
##
## Time series regression with "ts" data:
## Start = 3, End = 70
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.53322 -0.13634 -0.02932  0.12451  0.72194
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.088124   0.155887   0.565   0.5739
## stock.price.t -0.008214   0.007406  -1.109   0.2716
## stock.price.1 -0.010377   0.012166  -0.853   0.3969
## stock.price.2  0.025347   0.009201   2.755   0.0077 **
## Bond.Yield.1   0.906344   0.121026   7.489 3.05e-10 ***
## Bond.Yield.2   0.050408   0.123124   0.409   0.6836
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2478 on 62 degrees of freedom
## Multiple R-squared:  0.9237, Adjusted R-squared:  0.9176
## F-statistic: 150.1 on 5 and 62 DF,  p-value: < 2.2e-16
```

- According to the summaries of the two ARDL models of stock price predicted by bond.yield, the coefficients are not significant or too close to 0, meaning that bond.yield perfictly predicts stock price. Because of the unrealistic perfectness, we consider the model overfitting.

```r
# residual plots
tsdisplay(resid(ardl_bond_stock1), main = "residual plots of ARDL(1,1)")
```

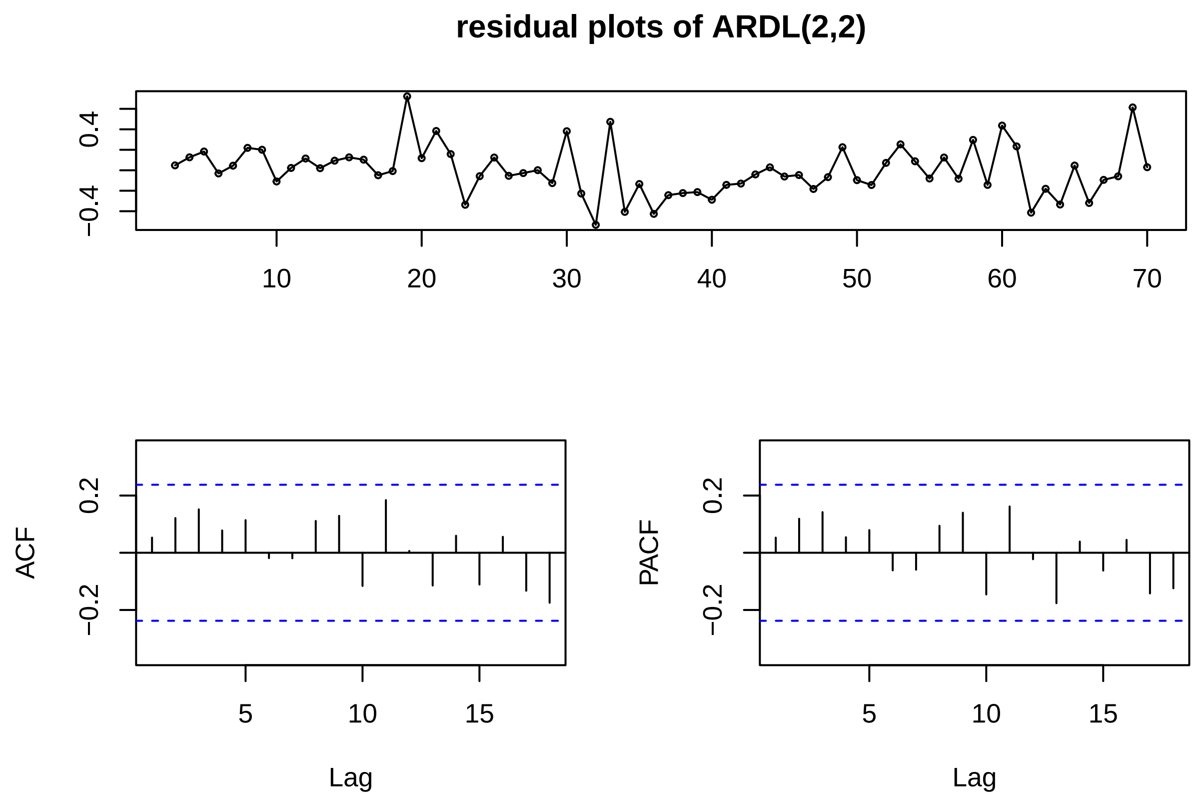
```
## Time Series:
## Start = 2
## End = 70
## Frequency = 1
##           2           3           4           5           6           7
## -0.03698001  0.03956557  0.15817875  0.18616895 -0.08199009  0.03557070
##           8           9          10          11          12          13
##  0.26435864  0.19395628 -0.16478987  0.04373515  0.11949548  0.00742725
##          14          15          16          17          18          19
##  0.11860576  0.13095032  0.09299336 -0.08360737  0.01790429  0.74591699
##          20          21          22          23          24          25
##  0.04106626  0.39150908  0.15669277 -0.39326127 -0.04561434  0.11604894
##          26          27          28          29          30          31
## -0.10943721 -0.04764098 -0.04433852 -0.20423228  0.26081528 -0.11845166
##          32          33          34          35          36          37
## -0.35686471  0.65888046 -0.49356104 -0.12698732 -0.42214340 -0.32395241
##          38          39          40          41          42          43
## -0.19448996 -0.09180186 -0.28427129 -0.08911818 -0.08444633  0.01178141
##          44          45          46          47          48          49
## -0.02160555 -0.05797641 -0.08072699 -0.20231938  0.01709533  0.24062602
##          50          51          52          53          54          55
## -0.07125342 -0.18025927  0.02665524  0.23867717  0.11862745 -0.13822170
##          56          57          58          59          60          61
## -0.05469723 -0.16188109  0.43412122 -0.11690384  0.25251104  0.36292163
##          62          63          64          65          66          67
## -0.60470177  0.05427657 -0.39256917 -0.07631295 -0.35673357  0.13979322
##          68          69          70
## -0.07817772  0.59464449  0.12074909
```




```r
tsdisplay(resid(ardl_bond_stock2), main = "residual plots of ARDL(2,2)")
```

```
## Time Series:
## Start = 3
## End = 70
## Frequency = 1
##            3            4            5            6            7            8
##  0.047787219  0.126417891  0.182890937 -0.031094908  0.044764530  0.218839961
##            9           10           11           12           13           14
##  0.200589149 -0.109372154  0.022648684  0.114897409  0.020170142  0.093661517
##           15           16           17           18           19           20
##  0.126308328  0.103447521 -0.048670086 -0.007910654  0.721940095  0.118129423
##           21           22           23           24           25           26
##  0.383911299  0.158077570 -0.337220905 -0.057516411  0.124480750 -0.054341520
##           27           28           29           30           31           32
## -0.027544303  0.001059791 -0.125271132  0.381507815 -0.227417237 -0.533220101
##           33           34           35           36           37           38
##  0.473335642 -0.406465863 -0.134301698 -0.425978821 -0.243023210 -0.222683317
##           39           40           41           42           43           44
## -0.213276539 -0.287985538 -0.142464202 -0.130082742 -0.040516748  0.028489077
##           45           46           47           48           49           50
## -0.061082283 -0.046665711 -0.182776550 -0.067025833  0.225264287 -0.096367566
##           51           52           53           54           55           56
## -0.144245299  0.072072571  0.253002684  0.087750117 -0.080482601  0.124613459
##           57           58           59           60           61           62
```

```
## -0.082616420  0.296702828 -0.142976554  0.436618537  0.233380399 -0.414211895
##          63           64           65           66           67           68
## -0.180767578 -0.334611971  0.046300244 -0.319503159 -0.093973016 -0.058931323
##          69           70
##  0.613760962  0.029775011
```



- For both ARDL(1, 1) and ARDL(2, 2) models, there is no obvious pattern and the residuals look like white noise.

```r
# data splitting
data_tr <- data[1:46, ]
data_ts <- data[47:70, ]

# ARDL(1, 1) from training
ardl_bond_stock1_tr <- ardlDlm(Bond.Yield ~ stock.price, data = data_tr, p = 1, q = 1)
# training RMSE
(ardl_bond_stock1_tr_rmse <- sqrt(mean(resid(ardl_bond_stock1_tr)^2)))
```

```
## Time Series:
## Start = 2
## End = 46
## Frequency = 1
##             2             3             4             5             6
## -0.0816423289  0.0008230136  0.1005407847  0.1305546467 -0.1069067890
##             7             8             9            10            11
##  0.0162933212  0.2174766154  0.1503406970 -0.1803737333  0.0228626602
```

```
##            12            13            14            15            16 
##  0.0975994855 -0.0128551840  0.0817660089  0.0879035917  0.0560755646 
##            17            18            19            20            21 
## -0.1054523334 -0.0208074278  0.6957158544  0.0059465042  0.3414548144 
##            22            23            24            25            26 
##  0.0926140953 -0.4335830039 -0.0811589207  0.0932702488 -0.0968718794 
##            27            28            29            30            31 
## -0.0059654245  0.0508168515 -0.0237219456  0.5216940469  0.0473898015 
##            32            33            34            35            36 
## -0.3196262190  0.5997402198 -0.5211926226 -0.1379872117 -0.4090807024 
##            37            38            39            40            41 
## -0.2358681028 -0.1163222012 -0.0714330857 -0.2576362327 -0.0806341225 
##            42            43            44            45            46 
## -0.0969194589 -0.0109828722  0.0010134372 -0.0139337602  0.0090632987
```

```
## [1] 0.2291805
```

```r
# testing RMSE
ardl_bond_stock1_ts <- forecast(model = ardl_bond_stock1_tr, x = data_ts$stock.price, h = 24)
(ardl_bond_stock1_ts_rmse <- sqrt(mean((data_ts$Bond.Yield - ardl_bond_stock1_ts$forecasts)^2)))
```

```
## [1] 6.448766
```

- The training RMSE of ARDL(1, 1) is 0.2291805, and the testing RMSE is 6.448766.

```r
# ARDL(2, 2) from training
ardl_bond_stock2_tr <- ardlDlm(Bond.Yield ~ stock.price, data = data_tr, p = 2, q = 2)
# training RMSE
(ardl_bond_stock2_tr_rmse <- sqrt(mean(resid(ardl_bond_stock2_tr)^2)))
```

```
## Time Series:
## Start = 3 
## End = 46 
## Frequency = 1 
##            3            4            5            6            7            8 
##  0.033541526  0.098710309  0.138786930 -0.053033494  0.044738976  0.197626428 
##            9           10           11           12           13           14 
##  0.162251554 -0.123077751  0.008611598  0.100383078  0.013394729  0.072863647 
##           15           16           17           18           19           20 
##  0.094192783  0.066068216 -0.065944599 -0.031748705  0.676800199  0.092166875 
##           21           22           23           24           25           26 
##  0.341369514  0.089955699 -0.395225809 -0.111279569  0.078963427 -0.070985473 
##           27           28           29           30           31           32 
## -0.027468753  0.030943279 -0.040342989  0.597521874 -0.021616527 -0.435448677 
##           33           34           35           36           37           38 
##  0.432741991 -0.441448400 -0.165096368 -0.451472571 -0.191422307 -0.125206662 
##           39           40           41           42           43           44 
## -0.169876814 -0.245803608 -0.106902280 -0.106371434 -0.049812044  0.052796555 
##           45           46 
## -0.022218861  0.027374509
```

```
## [1] 0.2263987
```

```r
# testing RMSE
ardl_bond_stock2_ts <- forecast(model = ardl_bond_stock2_tr, x = data_ts$stock.price, h = 24)
(ardl_bond_stock2_ts_rmse <- sqrt(mean((data_ts$Bond.Yield - ardl_bond_stock2_ts$forecasts)^2)))
```

```
## [1] 5.045455
```

- Bacause of the perfect fit of the models, both the training and testing RMSE are very small. The training RMSE of ARDL(2, 2) is 0.2263987, and the testing RMSE is 5.045455.
- ARDL(2, 2) has a smaller testing RMSE, so we should choose ARDL(2, 2) over ARDL(1, 1).

```r
# AIC&BIC of two models
AIC_BIC_1 <- data.frame("(AIC_of_ARDL(1, 1)" = AIC(ardl_bond_stock1_tr),
                        "AIC_of_ARDL(2, 2)" = AIC(ardl_bond_stock2_tr),
                        "BIC_of_ARDL(1, 1)" = BIC(ardl_bond_stock1_tr),
                        "BIC_of_ARDL(2, 2)" = BIC(ardl_bond_stock2_tr))
```

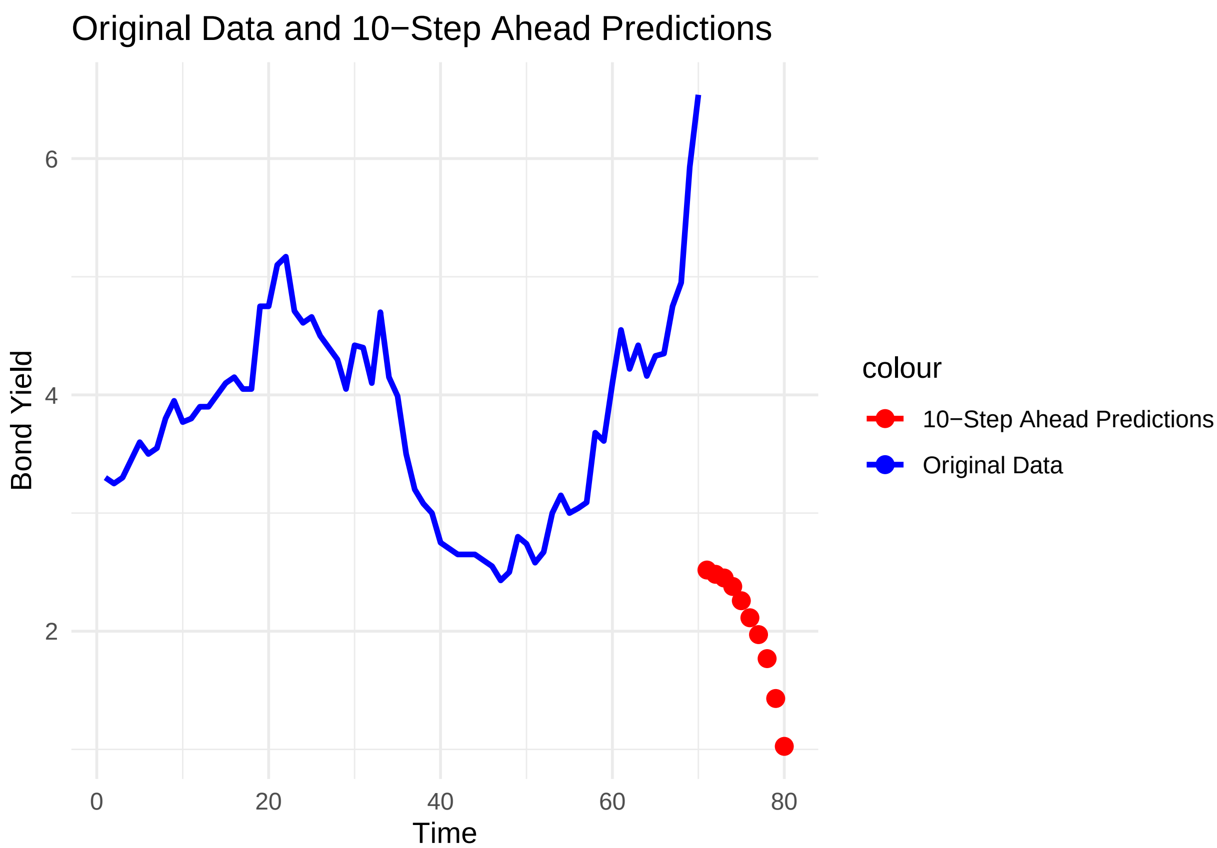
```
## [1] 5.112381
## [1] 8.14631
## [1] 14.14569
## [1] 20.63564
```

```r
AIC_BIC_1
```

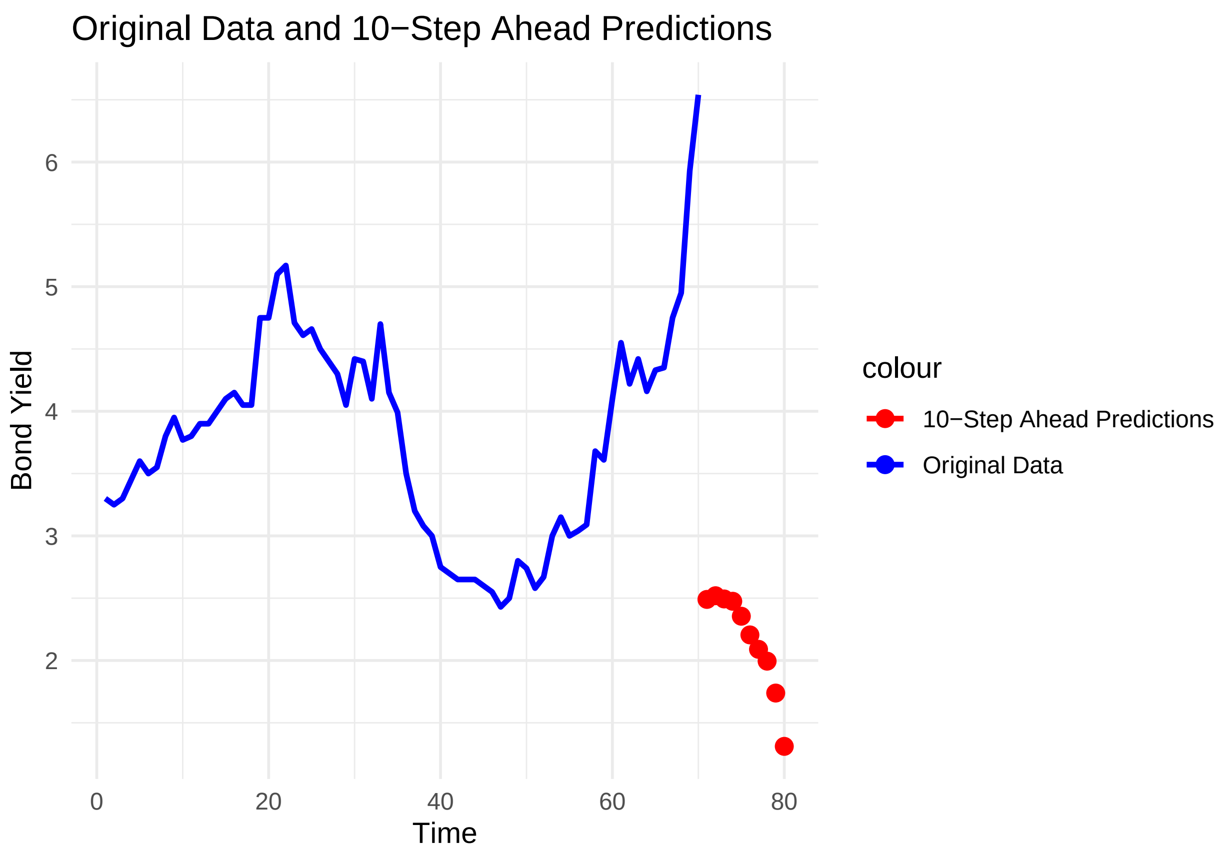
```
##   X.AIC_of_ARDL.1..1. AIC_of_ARDL.2..2. BIC_of_ARDL.1..1. BIC_of_ARDL.2..2.
## 1            5.112381           8.14631          14.14569          20.63564
```

- ARDL(1, 1) has smaller AIC and BIC, so according AIC and BIC, we should keep ARDL(1, 1).

```r
# 10-step ahead for ARDL(1, 1)
ardl_bond_stock1_10 <- forecast(model = ardl_bond_stock1_tr, x = data_ts[1:10, ]$stock.price, h = 10)

# Combine the original data and the predictions into a new data frame
prediction_data <- data.frame(
  Time = (length(data$Bond.Yield) + 1):(length(data$Bond.Yield) + length(ardl_bond_stock1_10$forecasts))
  Prediction = ardl_bond_stock1_10$forecasts)

# Create the ggplot
ggplot() +
  geom_line(data = data, aes(x = 1:length(Bond.Yield), y = Bond.Yield, color = "Original Data"), size =
  geom_point(data = prediction_data, aes(x = Time, y = Prediction, color = "10-Step Ahead Predictions")
  scale_color_manual(values = c("Original Data" = "blue", "10-Step Ahead Predictions" = "red")) +
  labs(title = "Original Data and 10-Step Ahead Predictions",
       x = "Time",
       y = "Bond Yield") +
  theme_minimal()
```




```
# 10-step ahead for ARDL(2, 2)
ardl_bond_stock2_10 <- forecast(model = ardl_bond_stock2_tr, x = data_ts[1:10, ]$stock.price, h = 10)

# Combine the original data and the predictions into a new data frame
prediction_data <- data.frame(
  Time = (length(data$Bond.Yield) + 1):(length(data$Bond.Yield) + length(ardl_bond_stock2_10$forecasts))
  Prediction = ardl_bond_stock2_10$forecasts)

# Create the ggplot
ggplot() +
  geom_line(data = data, aes(x = 1:length(Bond.Yield), y = Bond.Yield, color = "Original Data"), size =
  geom_point(data = prediction_data, aes(x = Time, y = Prediction, color = "10-Step Ahead Predictions")
  scale_color_manual(values = c("Original Data" = "blue", "10-Step Ahead Predictions" = "red")) +
  labs(title = "Original Data and 10-Step Ahead Predictions",
       x = "Time",
       y = "Bond Yield") +
  theme_minimal()
```




```
# using bond.yield to predict stock price
# ARDL(1,1)
ardl_stock_bond1 <- ardlDlm(stock.price ~ Bond.Yield, data = data, p = 1, q = 1)
summary(ardl_stock_bond1)
```

```
##
## Time series regression with "ts" data:
## Start = 2, End = 70
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.3938  -1.7038  -0.0948   2.5944   9.4265
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     4.32490    2.44409   1.770   0.0815 .
## Bond.Yield.t   -2.76228    1.96938  -1.403   0.1655
## Bond.Yield.1    1.65069    2.03907   0.810   0.4212
## stock.price.1   1.04421    0.02237  46.684   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 4.133 on 65 degrees of freedom
## Multiple R-squared:  0.9769, Adjusted R-squared:  0.9758
## F-statistic: 916.8 on 3 and 65 DF,  p-value: < 2.2e-16
```

```r
# ARDL(2,2)
ardl_stock_bond2 <- ardlDlm(stock.price ~ Bond.Yield, data = data, p = 2, q = 2)
summary(ardl_stock_bond2)
```

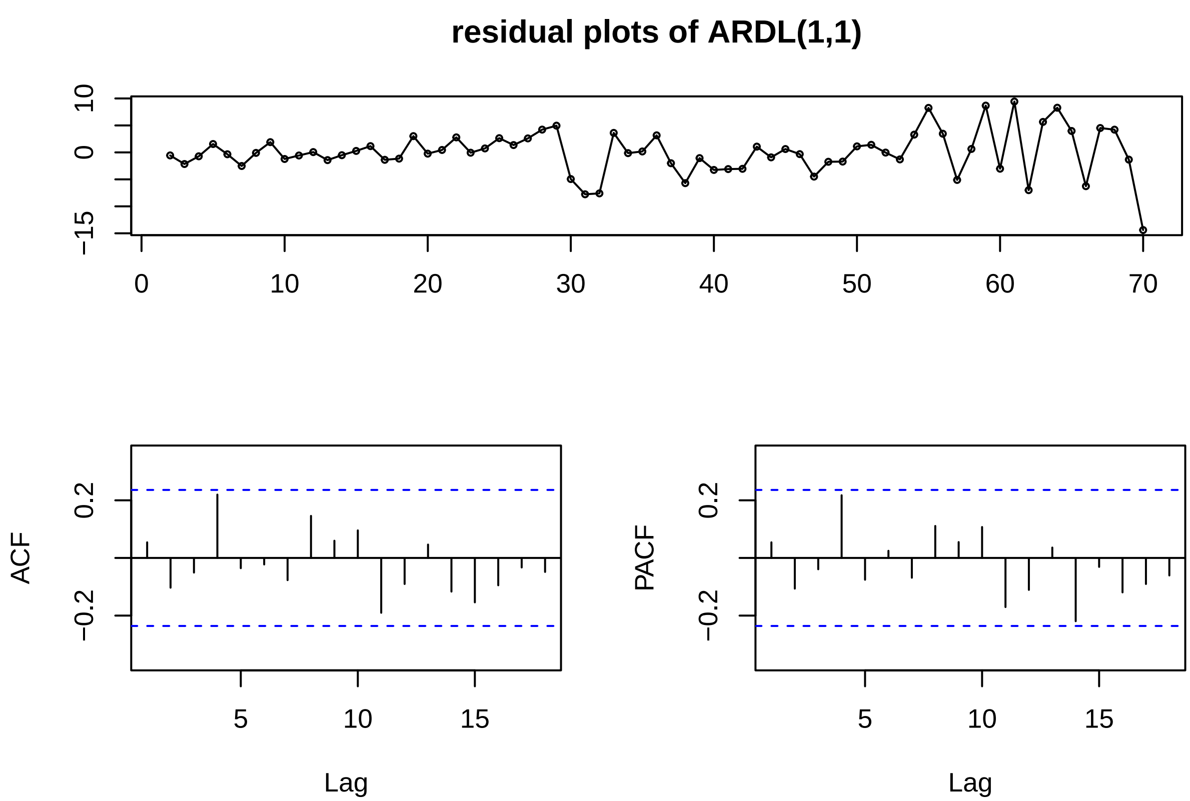
```
##
## Time series regression with "ts" data:
## Start = 3, End = 70
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2745  -1.5974  -0.0629   2.2396  10.1602
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)    3.76578    2.61059   1.442    0.154
## Bond.Yield.t  -2.36873    2.13559  -1.109    0.272
## Bond.Yield.1   0.25748    2.83614   0.091    0.928
## Bond.Yield.2   1.16309    2.08845   0.557    0.580
## stock.price.1  1.12758    0.15059   7.488 3.07e-10 ***
## stock.price.2 -0.08902    0.16514  -0.539    0.592
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.208 on 62 degrees of freedom
## Multiple R-squared:  0.977,  Adjusted R-squared:  0.9752
## F-statistic: 527.4 on 5 and 62 DF,  p-value: < 2.2e-16
```

- According to the summaries of the two ARDL models of stock price predicted by bond.yield, the lag 1 of stock price is significant, and all the other coefficients are not significant.

```r
# residual plots
tsdisplay(resid(ardl_stock_bond1), main = "residual plots of ARDL(1,1)")
```

```
## Time Series:
## Start = 2
## End = 70
## Frequency = 1
##           2           3           4           5           6           7
## -0.56138839 -2.15638176 -0.72107938  1.55273232 -0.34686871 -2.52242209
##           8           9          10          11          12          13
## -0.09480728  1.89951388 -1.22062757 -0.57471823  0.05685225 -1.43103832
##          14          15          16          17          18          19
## -0.51971531  0.27045394  1.16333009 -1.37671716 -1.15876334  3.01727590
##          20          21          22          23          24          25
## -0.23303165  0.44913538  2.78426785 -0.06045677  0.73556097  2.63752341
```

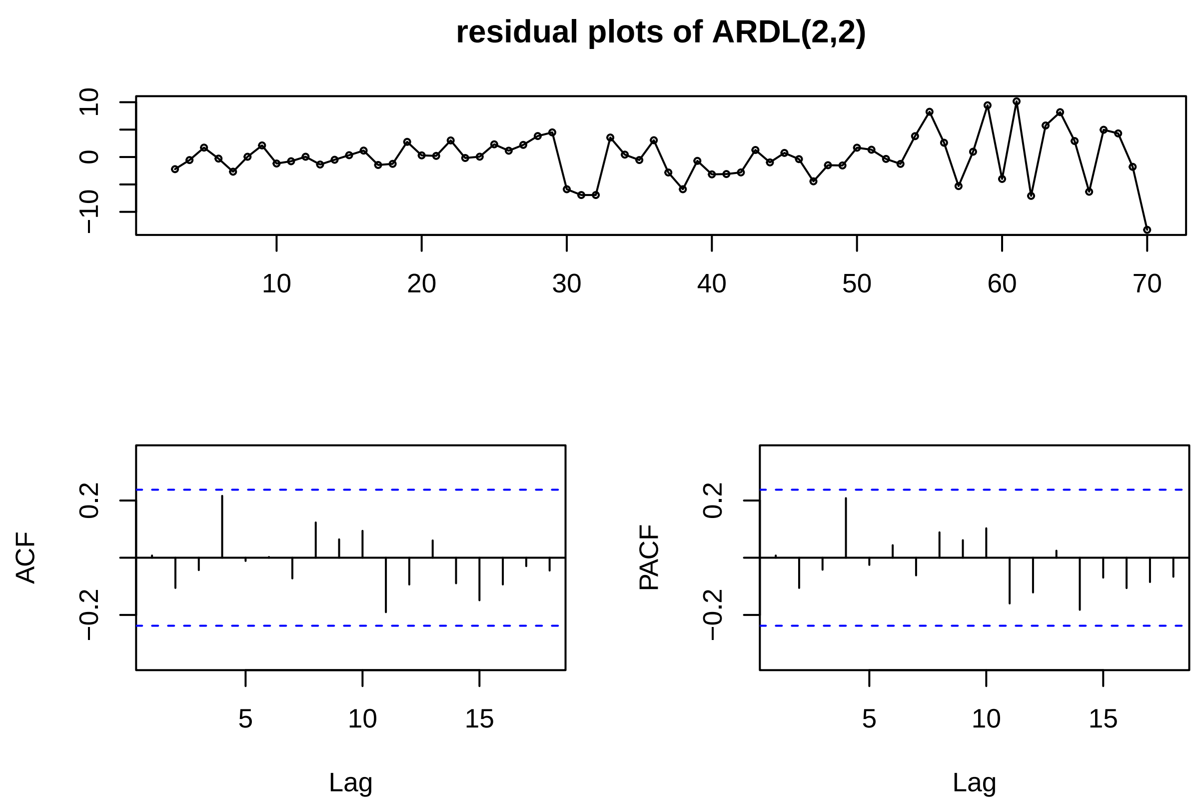
```
##            26           27           28           29           30           31
##    1.36018125   2.59440092   4.22166295   4.95235093  -4.94128994  -7.76668145
##            32           33           34           35           36           37
##   -7.59651944   3.61359559  -0.13582370   0.17118801   3.15818117  -2.00696874
##            38           39           40           41           42           43
##   -5.70058144  -1.06017480  -3.25388966  -3.09335095  -3.04587750   1.06749950
##            44           45           46           47           48           49
##   -0.91761666   0.62138493  -0.32312108  -4.48694458  -1.74105915  -1.70383918
##            50           51           52           53           54           55
##    1.11847870   1.40540757  -0.03606593  -1.29857061   3.29087388   8.24964259
##            56           57           58           59           60           61
##    3.46026295  -5.10866210   0.63758109   8.66807965  -3.02536054   9.42646798
##            62           63           64           65           66           67
##   -6.99857261   5.65059242   8.28112246   3.97146837  -6.26453658   4.51601511
##            68           69           70
##    4.21330840  -1.33910089 -14.39376892
```




```r
tsdisplay(resid(ardl_stock_bond2), main = "residual plots of ARDL(2,2)")
```

```
## Time Series:
## Start = 3
## End = 70
## Frequency = 1
##             3            4            5            6            7            8
##   -2.21031000  -0.55371538   1.71751222  -0.28419665  -2.67470033   0.04842561
```

```
##           9          10          11          12          13          14
##  2.11862369 -1.17871470 -0.76802823  0.05246274 -1.36496922 -0.49845763
##          15          16          17          18          19          20
##  0.33672351  1.17547893 -1.44809547 -1.24348307  2.76706022  0.30316063
##          21          22          23          24          25          26
##  0.21050037  3.02786418 -0.17431126  0.06341358  2.31562673  1.15487768
##          27          28          29          30          31          32
##  2.21427862  3.82659543  4.49313722 -5.87377438 -6.92865312 -6.92841959
##          33          34          35          36          37          38
##  3.55584532  0.43937646 -0.54741610  3.06418603 -2.81782929 -5.87372191
##          39          40          41          42          43          44
## -0.69861973 -3.16231911 -3.09991065 -2.80418574  1.28673397 -0.97853938
##          45          46          47          48          49          50
##  0.75120421 -0.38119762 -4.42990043 -1.49043561 -1.53423170  1.70530514
##          51          52          53          54          55          56
##  1.33169308 -0.34460485 -1.25482978  3.80377550  8.25369916  2.60628845
##          57          58          59          60          61          62
## -5.30173828  0.96789840  9.43530817 -3.99788238 10.16019949 -7.08285586
##          63          64          65          66          67          68
##  5.75636565  8.18094776  2.91755984 -6.33022444  4.97100025  4.30847417
##          69          70
## -1.78685003 -13.27448051
```



- The residual plots of ARDL(1, 1) and ARDL(2, 2) all look like white noise, meaning that the models have already captured almost all the info from the data.

```r
# using Bond.Yield to predict stock.price
# ARDL(1, 1) from training
ardl_stock_bond1_tr <- ardlDlm(stock.price ~ Bond.Yield, data = data_tr, p = 1, q = 1)
# training RMSE
(ardl_stock_bond1_tr_rmse <- sqrt(mean(resid(ardl_stock_bond1_tr)^2)))
```

```
## Time Series:
## Start = 2
## End = 46
## Frequency = 1
##           2           3           4           5           6           7
## -0.30833775 -1.84405514 -0.88784796  1.19662476 -0.04619043 -2.16166690
##           8           9          10          11          12          13
## -0.49038753  1.37175407 -0.98782159 -0.52870580 -0.04705033 -1.43126212
##          14          15          16          17          18          19
## -0.90246001 -0.31768127  0.59779063 -1.50912052 -1.56133581  1.45016039
##          20          21          22          23          24          25
## -1.26781831 -1.24627119  0.83605655 -0.99548228 -0.17781246  1.74260430
##          26          27          28          29          30          31
##  1.20821789  2.87550984  5.27618820  7.43487064 -1.54559410 -5.44527420
##          32          33          34          35          36          37
## -6.68775241  1.97985894 -0.34616470  0.20904622  3.98048443  0.24594512
##          38          39          40          41          42          43
## -3.40104206  0.34661447 -1.40171923 -1.51110896 -1.70738703  2.10808766
##          44          45          46
##  0.81586507  2.65875292  2.42291800
```

```
## [1] 2.395952
```

```r
# testing RMSE
ardl_stock_bond1_ts <- forecast(model = ardl_stock_bond1_tr, x = data_ts$Bond.Yield)
(ardl_stock_bond1_ts_rmse <- sqrt(mean((data_ts$stock.price - ardl_stock_bond1_ts$forecasts)^2)))
```

```
## [1] 46.623
```

- Bacause of the perfect fit of the models, both the training and testing RMSE are very small. The training RMSE of ARDL(1, 1) is 2.395952, and the testing RMSE is 46.623.

```r
# ARDL(2, 2) from training
ardl_stock_bond2_tr <- ardlDlm(stock.price ~ Bond.Yield, data = data_tr, p = 2, q = 2)
# training RMSE
(ardl_stock_bond2_tr_rmse <- sqrt(mean(resid(ardl_stock_bond2_tr)^2)))
```

```
## Time Series:
## Start = 3
## End = 46
## Frequency = 1
##           3           4           5           6           7           8
## -2.37052617 -0.63764505  1.04245009 -0.75740085 -2.68944908 -0.01863770
##           9          10          11          12          13          14
##  1.35749370 -1.56852018 -0.72485475 -0.14444795 -1.49072291 -0.69322522
```

```
##          15          16          17          18          19          20 
## -0.31097439  0.34794378 -2.00106767 -1.43743968  0.84339406 -1.08941098 
##          21          22          23          24          25          26 
## -1.58744364  1.31050368 -1.50671092 -1.11625738  1.08994579  0.44696501 
##          27          28          29          30          31          32 
##  2.24411610  4.41776418  6.36597946 -3.50866223 -1.22669205 -2.41817619 
##          33          34          35          36          37          38 
##  3.79695233 -0.07803882 -0.93645659  3.94704286 -1.89283080 -2.99370019 
##          39          40          41          42          43          44 
##  2.44615228 -1.19114337 -1.06900842 -1.01791371  2.59278978 -0.25937588 
##          45          46 
##  2.66503113  1.82220852
```

```
## [1] 2.103889
```

```
# testing RMSE
ardl_stock_bond2_ts <- forecast(model = ardl_stock_bond2_tr, x = data_ts$Bond.Yield)
(ardl_stock_bond2_ts_rmse <- sqrt(mean((data_ts$stock.price - ardl_stock_bond2_ts$forecasts)^2)))
```

```
## [1] 46.68779
```

- Bacause of the perfect fit of the models, both the training and testing RMSE are very small. The training RMSE of ARDL(2, 2) is 2.103889, and the testing RMSE is 46.68779.
- ARDL(1, 1) has slightly smaller testing RMSE, so we should choose ARDL(1, 1).

```
# AIC&BIC of two models
AIC_BIC_2 <- data.frame("(AIC_of_ARDL(1, 1)" = AIC(ardl_stock_bond1_tr),
                        "AIC_of_ARDL(2, 2)" = AIC(ardl_stock_bond2_tr),
                        "BIC_of_ARDL(1, 1)" = BIC(ardl_stock_bond1_tr),
                        "BIC_of_ARDL(2, 2)" = BIC(ardl_stock_bond2_tr))
```

```
## [1] 216.3447
## [1] 204.3199
## [1] 225.378
## [1] 216.8092
```

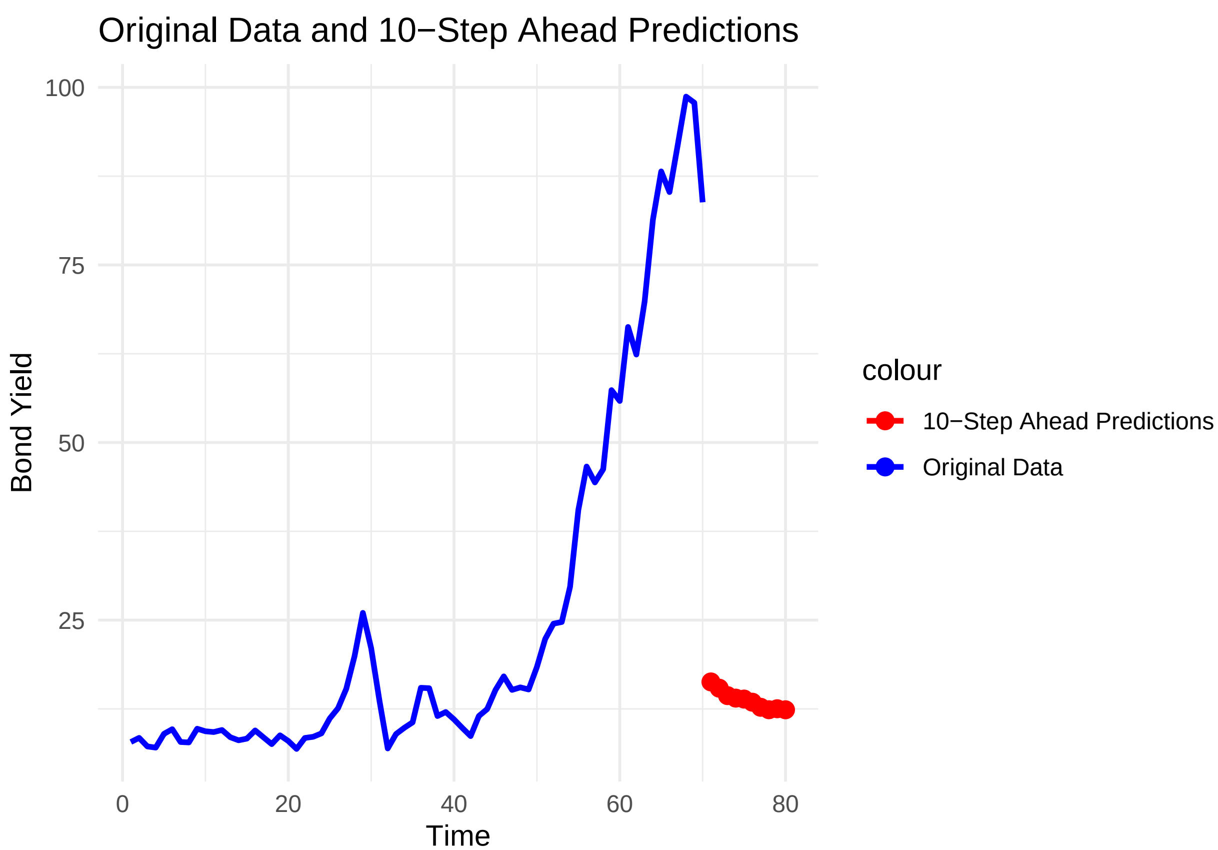
```
AIC_BIC_2
```

```
##   X.AIC_of_ARDL.1..1. AIC_of_ARDL.2..2. BIC_of_ARDL.1..1. BIC_of_ARDL.2..2.
## 1            216.3447          204.3199           225.378          216.8092
```

- ARDL(2, 2) has smaller BIC and AIC, so we should keep ARDL(2, 2) according to the AIC and BIC.

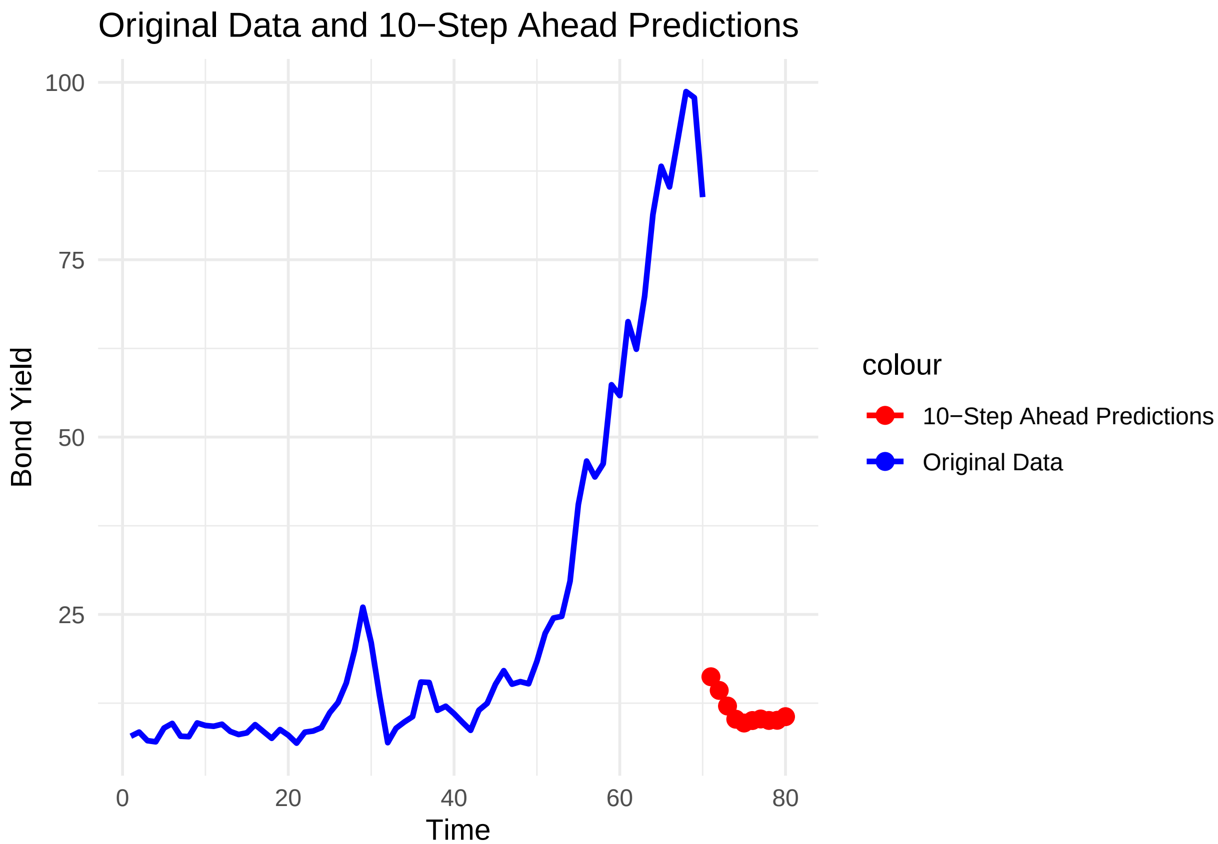
```
# 10-step ahead for ARDL(1, 1)
ardl_stock_bond1_10 <- forecast(model = ardl_stock_bond1_tr, x = data_ts[1:10, ]$Bond.Yield, h = 10)

# Combine the original data and the predictions into a new data frame
prediction_data <- data.frame(
  Time = (length(data$stock.price) + 1):(length(data$stock.price) + length(ardl_stock_bond1_10$forecasts
  Prediction = ardl_stock_bond1_10$forecasts)
```

```r
# Create the ggplot
ggplot() +
  geom_line(data = data, aes(x = 1:length(stock.price), y = stock.price, color = "Original Data"), size
  geom_point(data = prediction_data, aes(x = Time, y = Prediction, color = "10-Step Ahead Predictions")
  scale_color_manual(values = c("Original Data" = "blue", "10-Step Ahead Predictions" = "red")) +
  labs(title = "Original Data and 10-Step Ahead Predictions",
       x = "Time",
       y = "Bond Yield") +
  theme_minimal()
```




```r
# 10-step ahead for ARDL(2, 2)
ardl_stock_bond2_10 <- forecast(model = ardl_stock_bond2_tr, x = data_ts[1:10, ]$Bond.Yield, h = 10)

# Combine the original data and the predictions into a new data frame
prediction_data <- data.frame(
  Time = (length(data$stock.price) + 1):(length(data$stock.price) + length(ardl_stock_bond2_10$forecasts
  Prediction = ardl_stock_bond2_10$forecasts)

# Create the ggplot
ggplot() +
  geom_line(data = data, aes(x = 1:length(stock.price), y = stock.price, color = "Original Data"), size
  geom_point(data = prediction_data, aes(x = Time, y = Prediction, color = "10-Step Ahead Predictions")
  scale_color_manual(values = c("Original Data" = "blue", "10-Step Ahead Predictions" = "red")) +
  labs(title = "Original Data and 10-Step Ahead Predictions",
       x = "Time",
```

```r
       y = "Bond Yield") +
  theme_minimal()
```




```r
# using implicit price index to predict the velocity of money
# ARDL(1,1)
ardl_pi_vm1 <- ardlDlm(Velocity.of.Money ~ Implicit.Price.Index, data = data, p = 1, q = 1)
summary(ardl_pi_vm1)
```

```
##
## Time series regression with "ts" data:
## Start = 2, End = 70
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.33273 -0.05044  0.00530  0.06111  0.29785
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            0.204651   0.105626   1.937  0.05703 .
## Implicit.Price.Index.t  0.015458   0.004553   3.395  0.00117 **
## Implicit.Price.Index.1 -0.016061   0.004681  -3.431  0.00105 **
## Velocity.of.Money.1     0.886982   0.047736  18.581  < 2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1133 on 65 degrees of freedom
## Multiple R-squared:  0.876,  Adjusted R-squared:  0.8702
## F-statistic:   153 on 3 and 65 DF,  p-value: < 2.2e-16
```

```
# ARDL(2,2)
ardl_pi_vm2 <- ardlDlm(Velocity.of.Money ~ Implicit.Price.Index, data = data, p = 2, q = 2)
summary(ardl_pi_vm2)
```

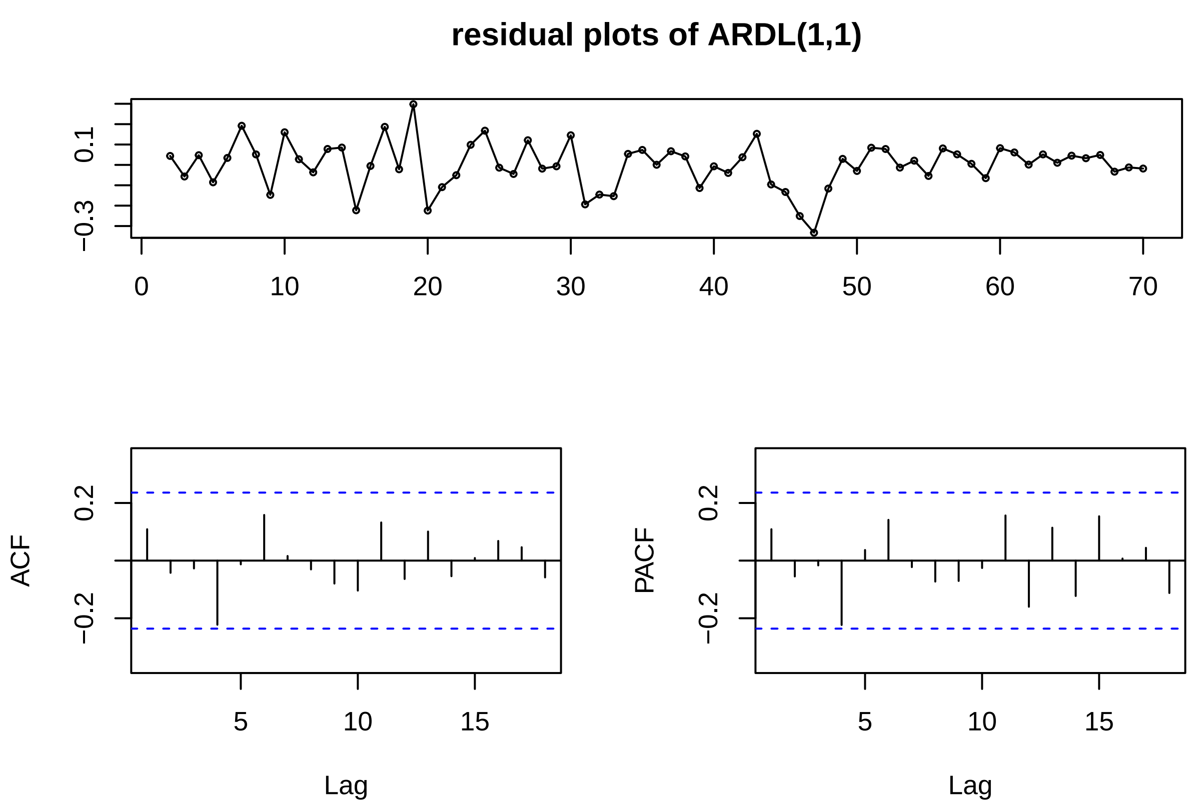
```
##
## Time series regression with "ts" data:
## Start = 3, End = 70
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.31702 -0.05129  0.00811  0.05420  0.33516
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.204356   0.112212   1.821  0.07341 .
## Implicit.Price.Index.t  0.017215   0.005167   3.331  0.00146 **
## Implicit.Price.Index.1 -0.024461   0.008886  -2.753  0.00774 **
## Implicit.Price.Index.2  0.006780   0.005584   1.214  0.22930
## Velocity.of.Money.1     0.993080   0.125425   7.918  5.5e-11 ***
## Velocity.of.Money.2    -0.106281   0.122250  -0.869  0.38799
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1143 on 62 degrees of freedom
## Multiple R-squared:  0.8714, Adjusted R-squared:  0.861
## F-statistic: 84.04 on 5 and 62 DF,  p-value: < 2.2e-16
```

- According to the summaries of the two ARDL models, we observe that Implicit.Price.Index, Implicit.Price.Index Lag 1, and Velocity.of.Money Lag 1 are significant.

```
# residual plots
tsdisplay(resid(ardl_pi_vm1), main = "residual plots of ARDL(1,1)")
```

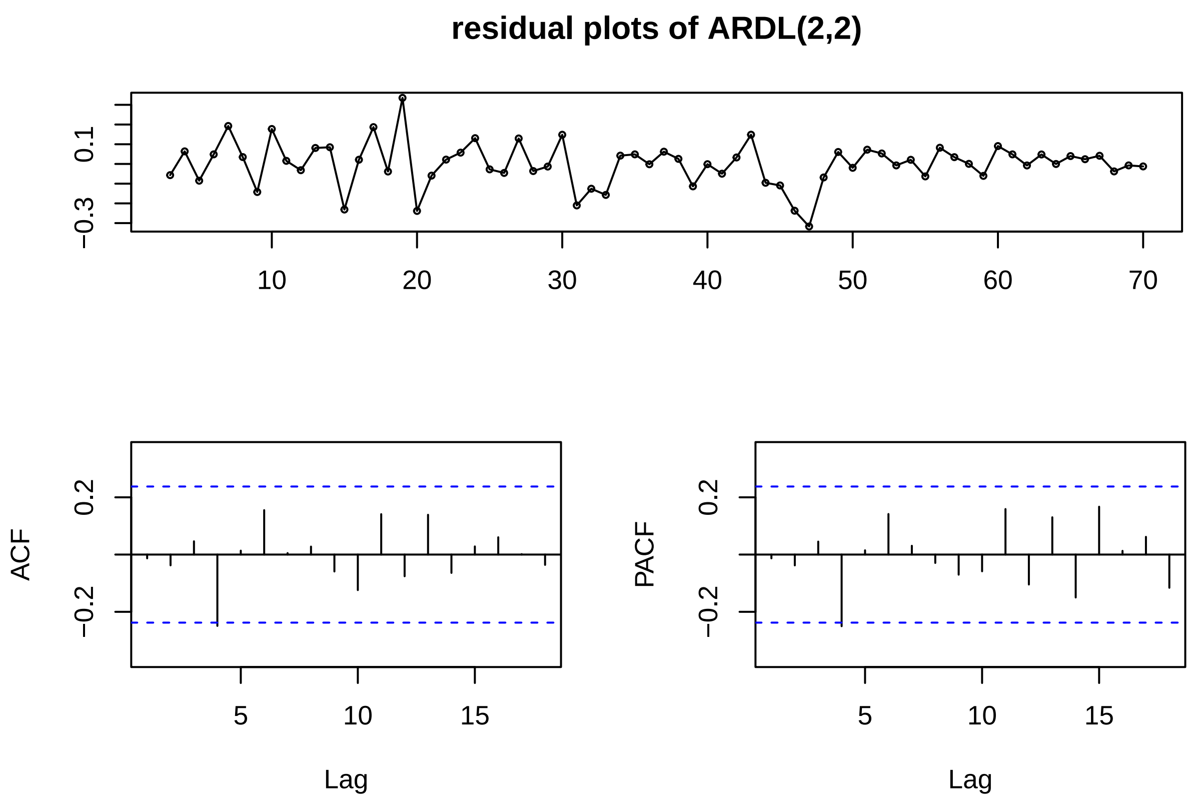
```
## Time Series:
## Start = 2
## End = 70
## Frequency = 1
##             2             3             4             5             6
##  0.0439357746 -0.0569625783  0.0476176719 -0.0854356285  0.0343357288
##             7             8             9            10            11
##  0.1919814523  0.0514787871 -0.1471967791  0.1601224454  0.0277838712
##            12            13            14            15            16
```

```
## -0.0365289014  0.0782084739  0.0850975113 -0.2227012389 -0.0052553106
##            17            18            19            20            21
##  0.1867686592 -0.0214317879  0.2978541796 -0.2241223208 -0.1091470054
##            22            23            24            25            26
## -0.0504420410  0.0985299448  0.1678773096 -0.0136230516 -0.0445131895
##            27            28            29            30            31
##  0.1209769930 -0.0183573274 -0.0068631522  0.1454620653 -0.1936291675
##            32            33            34            35            36
## -0.1456740279 -0.1534466498  0.0540871306  0.0734095727  0.0006462002
##            37            38            39            40            41
##  0.0670924776  0.0416145532 -0.1135456401 -0.0068709951 -0.0393321262
##            42            43            44            45            46
##  0.0375639798  0.1525669001 -0.0960000872 -0.1329197857 -0.2509347073
##            47            48            49            50            51
## -0.3327293786 -0.1160709931  0.0295304398 -0.0296172980  0.0842517423
##            52            53            54            55            56
##  0.0780268446 -0.0132893608  0.0207730854 -0.0541993705  0.0808891140
##            57            58            59            60            61
##  0.0518459873  0.0053038020 -0.0649856548  0.0822087435  0.0611081572
##            62            63            64            65            66
##  0.0014091248  0.0516024394  0.0105365208  0.0452611099  0.0328020532
##            67            68            69            70
##  0.0487694533 -0.0331955607 -0.0123787944 -0.0179303888
```



```r
tsdisplay(resid(ardl_pi_vm2), main = "residual plots of ARDL(2,2)")
```

```
## Time Series:
## Start = 3
## End = 70
## Frequency = 1
##             3             4             5             6             7
## -0.0567618955  0.0638272600 -0.0849129322  0.0486251205  0.1926594435
##             8             9            10            11            12
##  0.0348795060 -0.1422276216  0.1773971387  0.0157887934 -0.0317603323
##            13            14            15            16            17
##  0.0811455429  0.0846885852 -0.2312221799  0.0214429945  0.1867948265
##            18            19            20            21            22
## -0.0383728954  0.3351637195 -0.2383382280 -0.0591641116  0.0219736711
##            23            24            25            26            27
##  0.0574173428  0.1307739290 -0.0274045742 -0.0455021901  0.1293379767
##            28            29            30            31            32
## -0.0360094690 -0.0131503255  0.1481190231 -0.2103742991 -0.1256292481
##            33            34            35            36            37
## -0.1570371477  0.0421869223  0.0486634485 -0.0015831769  0.0621733612
##            38            39            40            41            42
##  0.0257868640 -0.1136654565 -0.0012201168 -0.0494717080  0.0328786841
##            43            44            45            46            47
##  0.1484247074 -0.0954725972 -0.1090964327 -0.2371865572 -0.3170206918
##            48            49            50            51            52
## -0.0685953619  0.0606907903 -0.0195289800  0.0723624138  0.0531246116
##            53            54            55            56            57
## -0.0073387892  0.0210215079 -0.0631389246  0.0822305841  0.0344019786
##            58            59            60            61            62
##  0.0004359407 -0.0605016313  0.0907183290  0.0485666591 -0.0076169717
##            63            64            65            66            67
##  0.0480425678 -0.0001686512  0.0399337123  0.0240443348  0.0414442833
##            68            69            70
## -0.0378876450 -0.0073273532 -0.0124780790
```



- The residual plots of ARDL(1, 1) and ARDL(2, 2) all look like white noise, meaning that the models have already captured almost all the info from the data.

```r
# using implicit price index to predict the velocity of money
# ARDL(1, 1) from training
ardl_pi_vm1_tr <- ardlDlm(Velocity.of.Money ~ Implicit.Price.Index, data = data_tr, p = 1, q = 1)
# training RMSE
(ardl_pi_vm1_tr_rmse <- sqrt(mean(resid(ardl_pi_vm1_tr)^2)))
```

```
## Time Series:
## Start = 2
## End = 46
## Frequency = 1
##            2            3            4            5            6            7
##  0.034192637 -0.074389198  0.026317627 -0.107536548  0.002350429  0.160583322
##            8            9           10           11           12           13
##  0.029992071 -0.159906805  0.127699054  0.007530078 -0.051204338  0.049808605
##           14           15           16           17           18           19
##  0.068952644 -0.240070610 -0.041874380  0.144180560 -0.056820540  0.305332243
##           20           21           22           23           24           25
## -0.183479837 -0.065068279  0.087974228  0.157824970  0.190828111  0.028372339
##           26           27           28           29           30           31
## -0.010818698  0.157714684  0.021264449  0.016853985  0.173571589 -0.154013677
##           32           33           34           35           36           37
## -0.115271268 -0.151909311  0.013389807  0.020077038 -0.023976357  0.046050159
##           38           39           40           41           42           43
```

```
##  0.020069901 -0.115929608 -0.019230167 -0.059813324  0.007282279  0.129417958
##           44           45           46
## -0.077468007 -0.097413180 -0.221436635
```

```
## [1] 0.1154869
```

```r
# testing RMSE
ardl_pi_vm1_ts <- forecast(model = ardl_pi_vm1_tr, x = data_ts$Implicit.Price.Index)
(ardl_pi_vm1_ts_rmse <- sqrt(mean((data_ts$Velocity.of.Money - ardl_pi_vm1_ts$forecasts)^2)))
```

```
## [1] 0.1932351
```

- Bacause of the perfect fit of the models, both the training and testing RMSE are very small. The training RMSE of ARDL(1, 1) is 0.1154869, and the testing RMSE is 0.1932351.

```r
# ARDL(2, 2) from training
ardl_pi_vm2_tr <- ardlDlm(Velocity.of.Money ~ Implicit.Price.Index, data = data_tr, p = 2, q = 2)
# training RMSE
(ardl_pi_vm2_tr_rmse <- sqrt(mean(resid(ardl_pi_vm2_tr)^2)))
```

```
## Time Series:
## Start = 3
## End = 46
## Frequency = 1
##            3            4            5            6            7            8
## -0.074413732  0.022401202 -0.103800863 -0.003350045  0.165135137  0.044787378
##            9           10           11           12           13           14
## -0.153776051  0.115527552  0.021578726 -0.046483733  0.045582790  0.078760600
##           15           16           17           18           19           20
## -0.234046154 -0.058003502  0.143244765 -0.049486482  0.307277163 -0.169634449
##           21           22           23           24           25           26
## -0.092696357  0.103809966  0.135704768  0.185962670  0.040786904 -0.016800194
##           27           28           29           30           31           32
##  0.154492295  0.028335672  0.009963943  0.173366206 -0.142836336 -0.125409928
##           33           34           35           36           37           38
## -0.159869552  0.002449757  0.026296075 -0.006792103  0.052117237  0.027215762
##           39           40           41           42           43           44
## -0.105010612 -0.023882968 -0.057844348  0.004666723  0.131952714 -0.061716701
##           45           46
## -0.102247077 -0.233314820
```

```
## [1] 0.1162296
```

```r
# testing RMSE
ardl_pi_vm2_ts <- forecast(model = ardl_pi_vm2_tr, x = data_ts$Implicit.Price.Index)
(ardl_pi_vm2_ts_rmse <- sqrt(mean((data_ts$Velocity.of.Money - ardl_pi_vm2_ts$forecasts)^2)))
```

```
## [1] 0.1822474
```

- Bacause of the perfect fit of the models, both the training and testing RMSE are very small. The training RMSE of ARDL(2, 2) is 0.1162296, and the testing RMSE is 0.1822474.

- ARDL(2, 2) has slightly smaller testing RMSE, so we should choose ARDL(2, 2).

```
# AIC&BIC of two models
AIC_BIC_3 <- data.frame("(AIC_of_ARDL(1, 1)" = AIC(ardl_pi_vm1_tr),
                        "AIC_of_ARDL(2, 2)" = AIC(ardl_pi_vm2_tr),
                        "BIC_of_ARDL(1, 1)" = BIC(ardl_pi_vm1_tr),
                        "BIC_of_ARDL(2, 2)" = BIC(ardl_pi_vm2_tr))
```

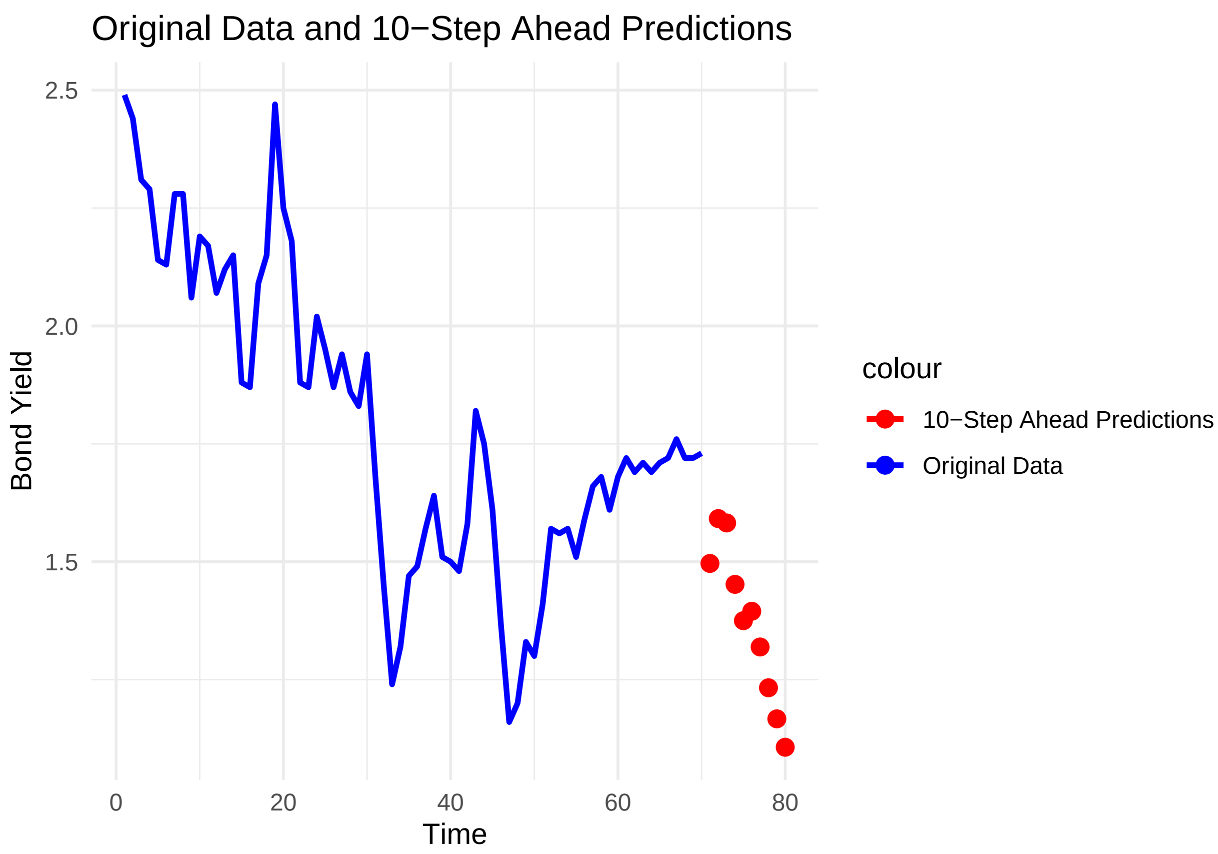
```
## [1] -56.56938
## [1] -50.52591
## [1] -47.53607
## [1] -38.03659
```

```
AIC_BIC_3
```

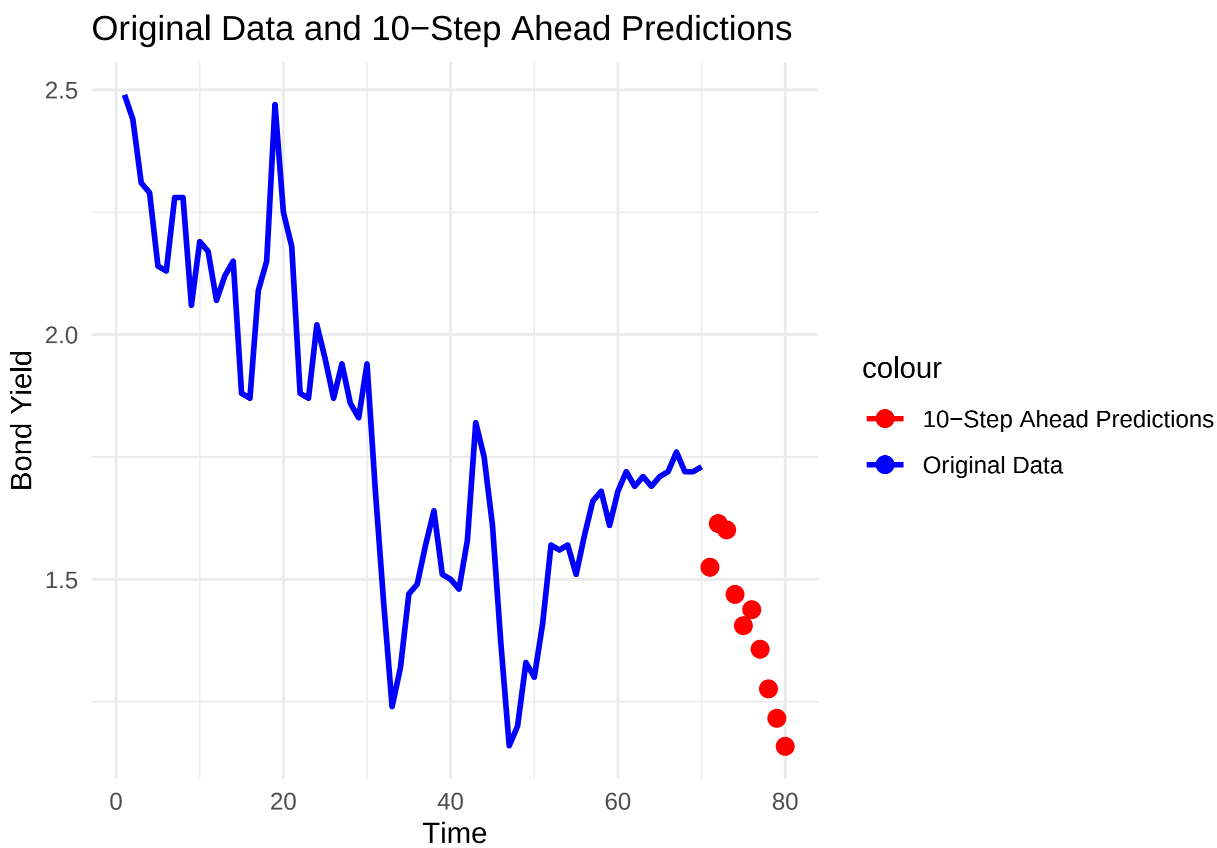
```
##   X.AIC_of_ARDL.1..1. AIC_of_ARDL.2..2. BIC_of_ARDL.1..1. BIC_of_ARDL.2..2.
## 1           -56.56938         -50.52591         -47.53607         -38.03659
```

- ARDL(1, 1) has smaller BIC and AIC, so we should keep ARDL(1, 1) according to the AIC and BIC.

```
# 10-step ahead for ARDL(1, 1)
ardl_pi_vm1_10 <- forecast(model = ardl_pi_vm1_tr, x = data_ts[1:10, ]$Implicit.Price.Index, h = 10)

# Combine the original data and the predictions into a new data frame
prediction_data <- data.frame(
  Time = (length(data$Velocity.of.Money) + 1):(length(data$Velocity.of.Money) + length(ardl_pi_vm1_10$fo
  Prediction = ardl_pi_vm1_10$forecasts)

# Create the ggplot
ggplot() +
  geom_line(data = data, aes(x = 1:length(Velocity.of.Money), y = Velocity.of.Money, color = "Original D
  geom_point(data = prediction_data, aes(x = Time, y = Prediction, color = "10-Step Ahead Predictions")
  scale_color_manual(values = c("Original Data" = "blue", "10-Step Ahead Predictions" = "red")) +
  labs(title = "Original Data and 10-Step Ahead Predictions",
       x = "Time",
       y = "Bond Yield") +
  theme_minimal()
```




```r
# 10-step ahead for ARDL(2, 2)
ardl_pi_vm2_10 <- forecast(model = ardl_pi_vm2_tr, x = data_ts[1:10, ]$Implicit.Price.Index, h = 10)

# Combine the original data and the predictions into a new data frame
prediction_data <- data.frame(
  Time = (length(data$Velocity.of.Money) + 1):(length(data$Velocity.of.Money) + length(ardl_pi_vm2_10$fo
  Prediction = ardl_pi_vm2_10$forecasts)

# Create the ggplot
ggplot() +
  geom_line(data = data, aes(x = 1:length(Velocity.of.Money), y = Velocity.of.Money, color = "Original D
  geom_point(data = prediction_data, aes(x = Time, y = Prediction, color = "10-Step Ahead Predictions"))
  scale_color_manual(values = c("Original Data" = "blue", "10-Step Ahead Predictions" = "red")) +
  labs(title = "Original Data and 10-Step Ahead Predictions",
       x = "Time",
       y = "Bond Yield") +
  theme_minimal()
```




```r
# using implicit price index to predict the velocity of money
# ARDL(1,1)
ardl_vm_pi1 <- ardlDlm(Implicit.Price.Index ~ Velocity.of.Money, data = data, p = 1, q = 1)
summary(ardl_vm_pi1)
```

```
##
## Time series regression with "ts" data:
## Start = 2, End = 70
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.0173  -1.4776  -0.3257   0.9866   7.8038
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -1.19718    2.72356  -0.440  0.66171
## Velocity.of.Money.t     9.74539    2.87015   3.395  0.00117 **
## Velocity.of.Money.1    -8.97671    2.79773  -3.209  0.00207 **
## Implicit.Price.Index.1  1.02395    0.01357  75.452  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 2.846 on 65 degrees of freedom
## Multiple R-squared:  0.9912, Adjusted R-squared:  0.9908
## F-statistic:  2435 on 3 and 65 DF,  p-value: < 2.2e-16
```

```r
# ARDL(2,2)
ardl_vm_pi2 <- ardlDlm(Implicit.Price.Index ~ Velocity.of.Money, data = data, p = 2, q = 2)
summary(ardl_vm_pi2)
```

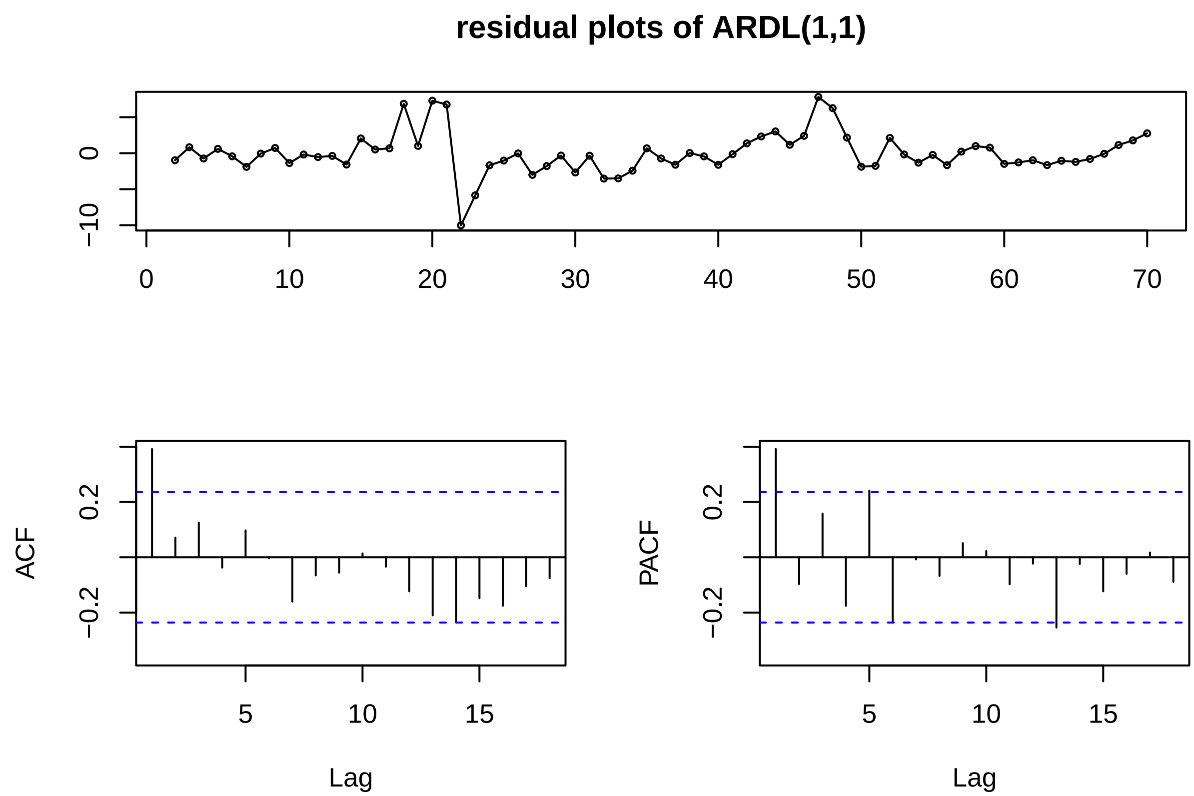
```
##
## Time series regression with "ts" data:
## Start = 3, End = 70
##
## Call:
## dynlm(formula = as.formula(model.text), data = data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.4542  -1.1350  -0.1409   0.7673   7.5863
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)              0.9318     2.6042   0.358 0.721714
## Velocity.of.Money.t      8.8197     2.6474   3.331 0.001459 **
## Velocity.of.Money.1     -7.2344     3.9198  -1.846 0.069726 .
## Velocity.of.Money.2     -1.7157     2.7753  -0.618 0.538709
## Implicit.Price.Index.1   1.4192     0.1136  12.488  < 2e-16 ***
## Implicit.Price.Index.2  -0.4134     0.1166  -3.545 0.000754 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.587 on 62 degrees of freedom
## Multiple R-squared:  0.9929, Adjusted R-squared:  0.9923
## F-statistic:  1731 on 5 and 62 DF,  p-value: < 2.2e-16
```

- According to the summaries of the two ARDL models, we observe that Velocity.of.Money, Implicit.Price.Index Lag 1 and 2 are significant, and all the other coefficients are not significant.

```r
# residual plots
tsdisplay(resid(ardl_vm_pi1), main = "residual plots of ARDL(1,1)")
```

```
## Time Series:
## Start = 2
## End = 70
## Frequency = 1
##            2            3            4            5            6            7
##  -0.97497878   0.82699064  -0.73264132   0.60466956  -0.42780756  -1.89217302
##            8            9           10           11           12           13
##  -0.05385134   0.72780721  -1.36432327  -0.18022028  -0.53427995  -0.37422094
##           14           15           16           17           18           19
##  -1.57355980   2.04896710   0.51045899   0.68124676   6.85554873   1.00999235
##           20           21           22           23           24           25
##   7.28510878   6.75560648 -10.01727133  -5.84684055  -1.67322542  -1.01995297
```

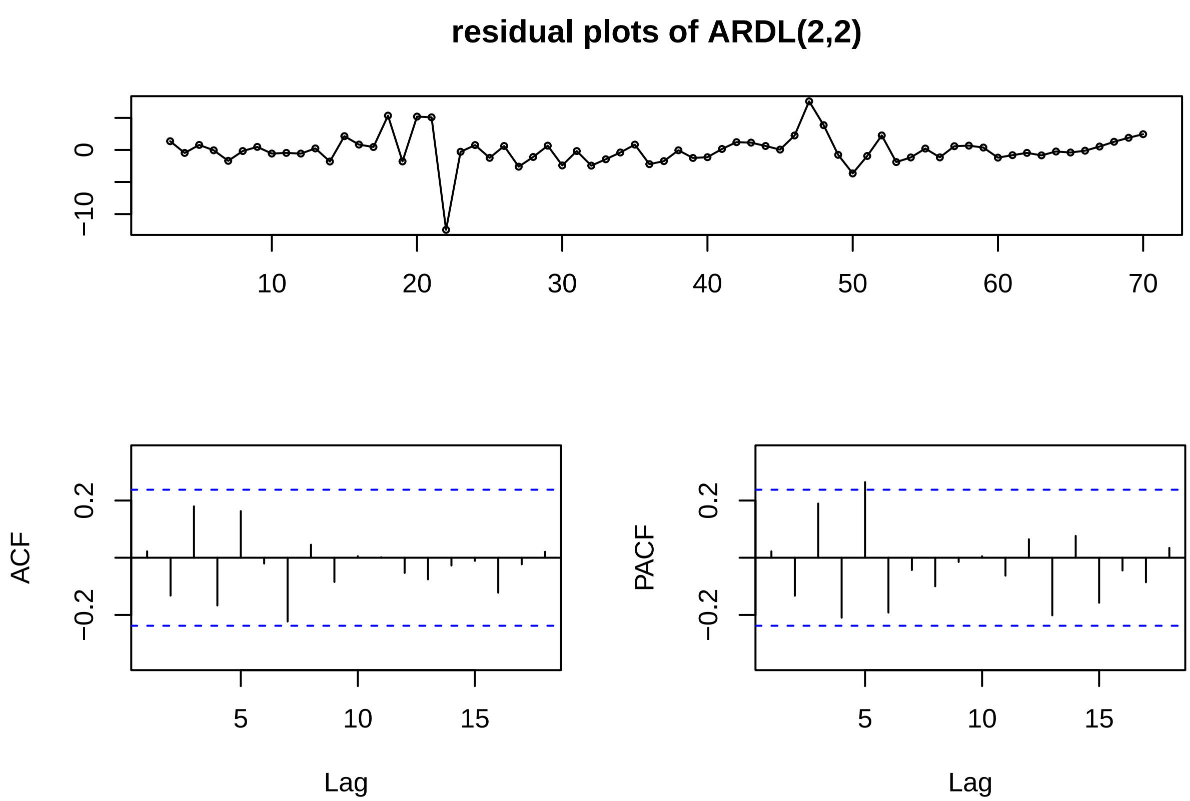
```
##          26          27          28          29          30          31
## -0.03086509 -3.00735180 -1.79018444 -0.32573944 -2.67148168 -0.34809839
##          32          33          34          35          36          37
## -3.52367058 -3.49421930 -2.42796039  0.67761766 -0.73319263 -1.59798977
##          38          39          40          41          42          43
##  0.02637694 -0.44107443 -1.60772529 -0.13305864  1.36498877  2.32198625
##          44          45          46          47          48          49
##  3.02644486  1.16803463  2.41174226  7.80379478  6.25621018  2.16325785
##          50          51          52          53          54          55
## -1.87679598 -1.75359998  2.13124154 -0.17268749 -1.31580400 -0.23334202
##          56          57          58          59          60          61
## -1.65287932  0.21845310  0.98662018  0.77224126 -1.47763732 -1.27836232
##          62          63          64          65          66          67
## -0.97621117 -1.65227456 -1.04537572 -1.20358806 -0.79126578 -0.07993405
##          68          69          70
##  1.13494266  1.77893367  2.75843298
```




```r
tsdisplay(resid(ardl_vm_pi2), main = "residual plots of ARDL(2,2)")
```

```
## Time Series:
## Start = 3
## End = 70
## Frequency = 1
##           3           4           5           6           7           8
##  1.36380757 -0.46388657  0.78571300 -0.04856210 -1.69689583 -0.16689299
```

```
##           9          10          11          12          13          14
##  0.47962963 -0.56049124 -0.45490422 -0.56509160  0.24223417 -1.80153151
##          15          16          17          18          19          20
##  2.15089419  0.83862469  0.46111024  5.35236760 -1.78134698  5.20208040
##          21          22          23          24          25          26
##  5.09202642 -12.45420652 -0.28495348  0.76122181 -1.22975777  0.61887843
##          27          28          29          30          31          32
## -2.60566662 -1.09260574  0.67163219 -2.41348407 -0.16986151 -2.44470908
##          33          34          35          36          37          38
## -1.44853179 -0.39489830  0.83563692 -2.21011291 -1.74038841 -0.04815391
##          39          40          41          42          43          44
## -1.24709638 -1.12744049  0.15643258  1.21747323  1.14572801  0.62004698
##          45          46          47          48          49          50
##  0.05906783  2.26304495  7.58627142  3.86935307 -0.75929201 -3.65698301
##          51          52          53          54          55          56
## -0.94013899  2.26708437 -1.88860138 -1.16591328  0.21982355 -1.15753777
##          57          58          59          60          61          62
##  0.58769635  0.67510178  0.37595034 -1.19695341 -0.80951571 -0.45490450
##          63          64          65          66          67          68
## -0.83921561 -0.24182358 -0.39420123 -0.11494281  0.53426889  1.27757820
##          69          70
##  1.90267400  2.45804050
```



- The residual plots of ARDL(1, 1) seem to contain some serial correlation, and ARDL(2, 2) has residual plots more similar to white noise.

```r
# using the velocity of money to predict implicit price index
# ARDL(1, 1) from training
ardl_vm_pi1_tr <- ardlDlm(Implicit.Price.Index ~ Velocity.of.Money, data = data_tr, p = 1, q = 1)
# training RMSE
(ardl_vm_pi1_tr_rmse <- sqrt(mean(resid(ardl_vm_pi1_tr)^2)))
```

```
## Time Series:
## Start = 2
## End = 46
## Frequency = 1
##           2           3           4           5           6           7
## -1.82699076  0.22282246 -1.31057922  0.31993144 -0.71119359 -2.46873236
##           8           9          10          11          12          13
## -0.60730120  0.61223317 -1.76533154 -0.51987188 -0.67612935 -0.62921596
##          14          15          16          17          18          19
## -1.87470041  2.28140550  0.72980942  0.47436389  6.57957196  0.15803928
##          20          21          22          23          24          25
##  6.93647927  6.55131298 -9.60091214 -5.50625855 -1.65098474 -0.83409869
##          26          27          28          29          30          31
##  0.30222123 -2.81866887 -1.43942300  0.06795993 -2.49363076  0.35404017
##          32          33          34          35          36          37
## -2.41131429 -2.02286849 -1.16478769  1.65206442  0.23665396 -0.78053229
##          38          39          40          41          42          43
##  0.71772903  0.52375071 -0.64344080  0.86565273  2.16805356  2.68399357
##          44          45          46
##  3.58739326  2.01414552  3.71733911
```

```
## [1] 2.837261
```

```r
# testing RMSE
ardl_vm_pi1_ts <- forecast(model = ardl_vm_pi1_tr, x = data_ts$Velocity.of.Money)
(ardl_vm_pi1_ts_rmse <- sqrt(mean((data_ts$Implicit.Price.Index - ardl_pi_vm1_ts$forecasts)^2)))
```

```
## [1] 97.13756
```

- Bacause of the perfect fit of the models, both the training and testing RMSE are very small. The training RMSE of ARDL(1, 1) is 2.837261, and the testing RMSE is 97.13756.

```r
# ARDL(2, 2) from training
ardl_vm_pi2_tr <- ardlDlm(Implicit.Price.Index ~ Velocity.of.Money, data = data_tr, p = 2, q = 2)
# training RMSE
(ardl_vm_pi2_tr_rmse <- sqrt(mean(resid(ardl_vm_pi2_tr)^2)))
```

```
## Time Series:
## Start = 3
## End = 46
## Frequency = 1
##             3             4             5             6             7             8
##   0.71915717  -0.61684053   0.23559243  -0.03976108  -2.37275165  -1.47117327
##             9            10            11            12            13            14
##   0.10752053  -0.32190017  -1.48875513  -0.87289742   0.06127049  -2.40571701
```

```
##           15           16           17           18           19           20 
##   1.70516628   1.75015733   0.21662127   4.53093332  -1.32816915   4.56350369 
##           21           22           23           24           25           26 
##   7.16101177 -10.40236022  -0.02920777   0.36098865  -1.46834491   1.19886625 
##           27           28           29           30           31           32 
##  -1.92750897  -1.19337167   1.16549891  -1.97070017  -0.20281377  -0.92836629 
##           33           34           35           36           37           38 
##  -0.42619542   0.30608021   0.62154311  -2.14507573  -1.50436168  -0.14871177 
##           39           40           41           42           43           44 
##  -0.92423556  -0.27947894   0.47951570   1.68322888   1.33797869   0.79884246 
##           45           46 
##   1.52331149   3.94190963
```

```
## [1] 2.491347
```

```r
# testing RMSE
ardl_vm_pi2_ts <- forecast(model = ardl_vm_pi2_tr, x = data_ts$Velocity.of.Money)
(ardl_vm_pi2_ts_rmse <- sqrt(mean((data_ts$Implicit.Price.Index - ardl_pi_vm2_ts$forecasts)^2)))
```

```
## [1] 97.1096
```

- Bacause of the perfect fit of the models, both the training and testing RMSE are very small. The training RMSE of ARDL(2, 2) is 2.491347, and the testing RMSE is 97.1096.
- ARDL(2, 2) has slightly smaller testing RMSE, so we should choose ARDL(2, 2).

```r
# AIC&BIC of two models
AIC_BIC_4 <- data.frame("(AIC_of_ARDL(1, 1)" = AIC(ardl_vm_pi1_tr),
                        "AIC_of_ARDL(2, 2)" = AIC(ardl_vm_pi2_tr),
                        "BIC_of_ARDL(1, 1)" = BIC(ardl_vm_pi1_tr),
                        "BIC_of_ARDL(2, 2)" = BIC(ardl_vm_pi2_tr))
```

```
## [1] 231.56
## [1] 219.1951
## [1] 240.5933
## [1] 231.6844
```

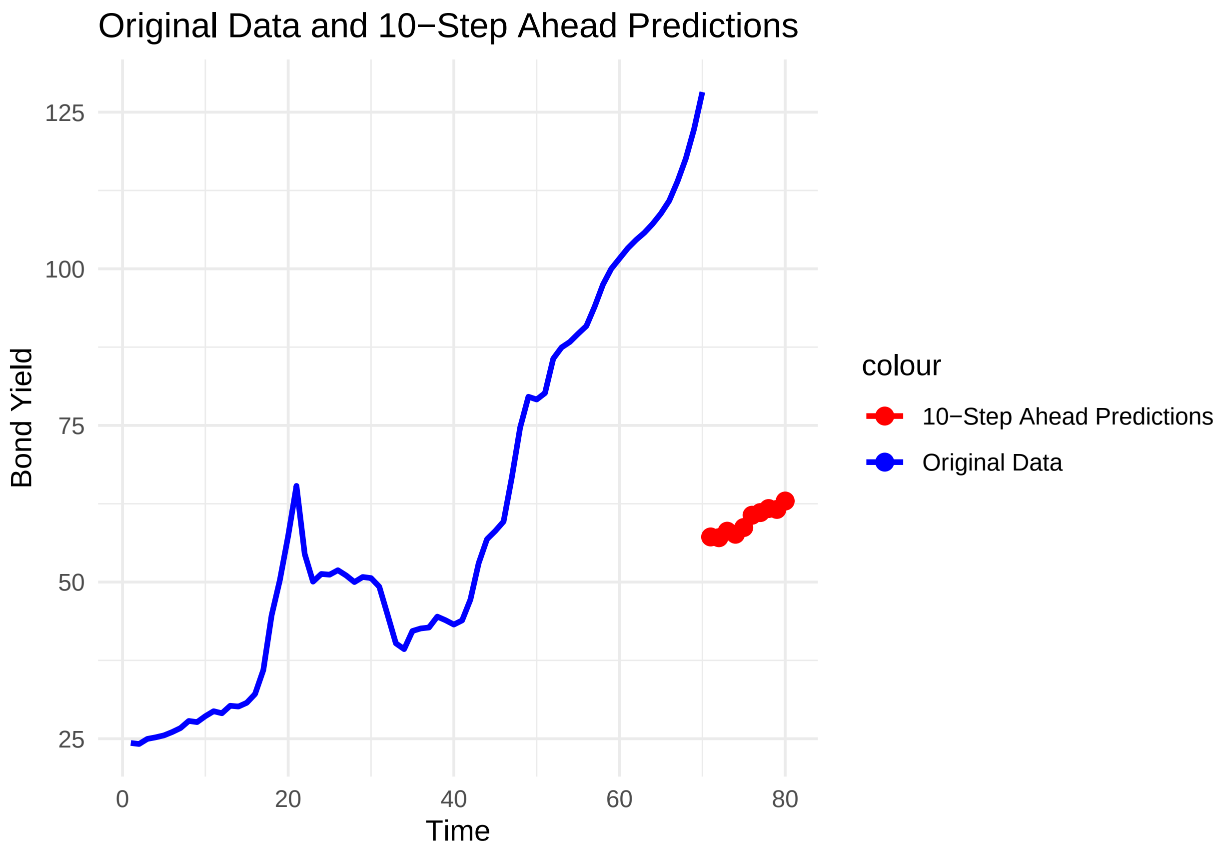
```r
AIC_BIC_4
```

```
##   X.AIC_of_ARDL.1..1. AIC_of_ARDL.2..2. BIC_of_ARDL.1..1. BIC_of_ARDL.2..2.
## 1              231.56          219.1951          240.5933          231.6844
```

- ARDL(2, 2) has smaller BIC and AIC, so we should keep ARDL(2, 2) according to the AIC and BIC.

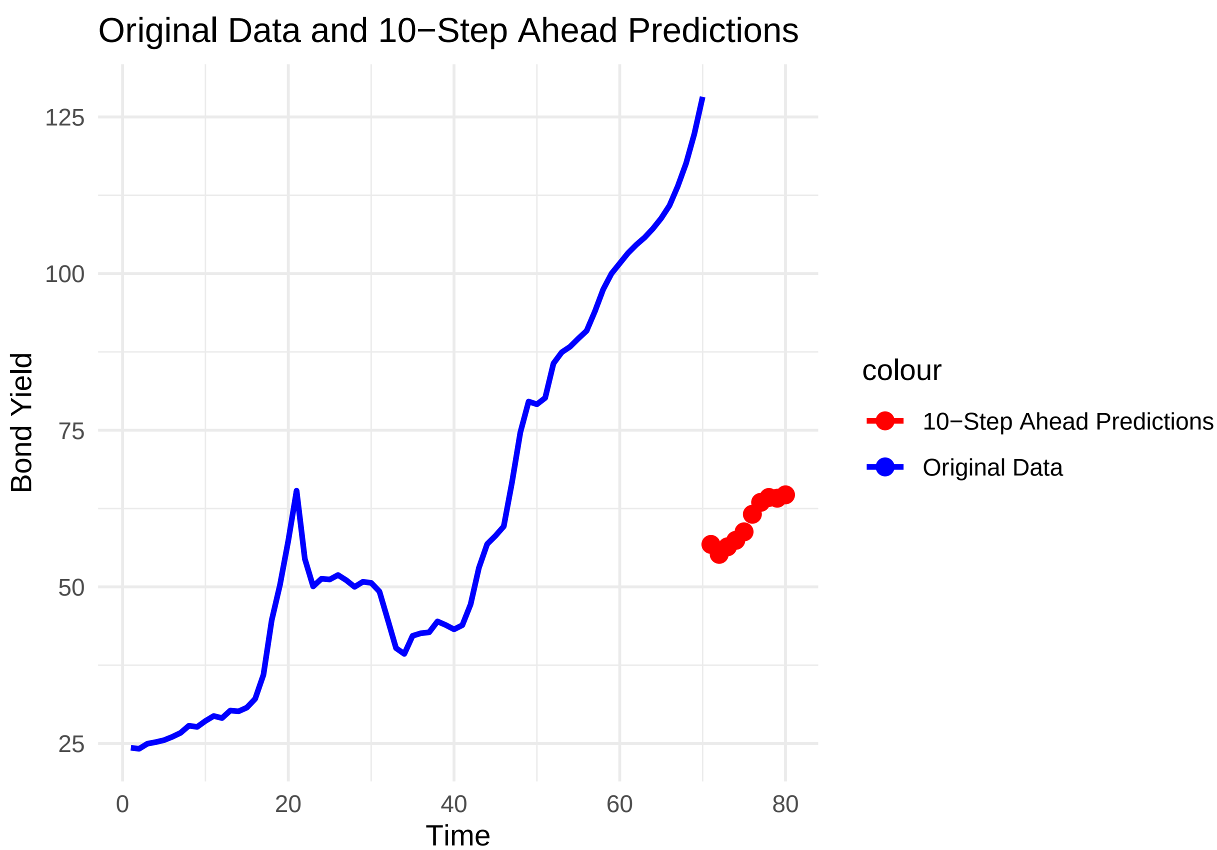
```r
# 10-step ahead for ARDL(1, 1)
ardl_vm_pi1_10 <- forecast(model = ardl_vm_pi1_tr, x = data_ts[1:10, ]$Velocity.of.Money, h = 10)

# Combine the original data and the predictions into a new data frame
prediction_data <- data.frame(
  Time = (length(data$Implicit.Price.Index) + 1):(length(data$Implicit.Price.Index) + length(ardl_vm_pi
  Prediction = ardl_vm_pi1_10$forecasts)
```

```r
# Create the ggplot
ggplot() +
  geom_line(data = data, aes(x = 1:length(Implicit.Price.Index), y = Implicit.Price.Index, color = "Orig
  geom_point(data = prediction_data, aes(x = Time, y = Prediction, color = "10-Step Ahead Predictions")
  scale_color_manual(values = c("Original Data" = "blue", "10-Step Ahead Predictions" = "red")) +
  labs(title = "Original Data and 10-Step Ahead Predictions",
       x = "Time",
       y = "Bond Yield") +
  theme_minimal()
```




```r
# 10-step ahead for ARDL(2, 2)
ardl_vm_pi2_10 <- forecast(model = ardl_vm_pi2_tr, x = data_ts[1:10, ]$Velocity.of.Money, h = 10)

# Combine the original data and the predictions into a new data frame
prediction_data <- data.frame(
  Time = (length(data$Implicit.Price.Index) + 1):(length(data$Implicit.Price.Index) + length(ardl_vm_pi2
  Prediction = ardl_vm_pi2_10$forecasts)

# Create the ggplot
ggplot() +
  geom_line(data = data, aes(x = 1:length(Implicit.Price.Index), y = Implicit.Price.Index, color = "Orig
  geom_point(data = prediction_data, aes(x = Time, y = Prediction, color = "10-Step Ahead Predictions")
  scale_color_manual(values = c("Original Data" = "blue", "10-Step Ahead Predictions" = "red")) +
  labs(title = "Original Data and 10-Step Ahead Predictions",
       x = "Time",
```

```
      y = "Bond Yield") +
  theme_minimal()
```



Part 5. Fit a VAR(p) model to the data. Evaluate model performance.

```
# Fit an appropriate VAR model
var_data <- data.frame(cbind(stock, price_index))
VARselect(var_data, lag.max = 10)
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
##
## $criteria
##                 1          2          3          4          5          6
## AIC(n)   5.360599   5.216286   5.280416   5.364987   5.423548   5.424285
## HQ(n)    5.442520   5.352821   5.471566   5.610751   5.723927   5.779278
## SC(n)    5.570033   5.565343   5.769097   5.993290   6.191475   6.331835
## FPE(n) 212.887857 184.391356 196.871485 214.765919 228.590952 230.037474
##                 7          8          9         10
## AIC(n)   5.538910   5.636210   5.678751   5.770306
## HQ(n)    5.948517   6.100431   6.197586   6.343756
## SC(n)    6.586082   6.823005   7.005169   7.236348
## FPE(n) 259.968718 289.480285 306.076723 341.067849
```

- We can see that all algorithms are suggest p = 2. Thus we consider a VAR(2) model:

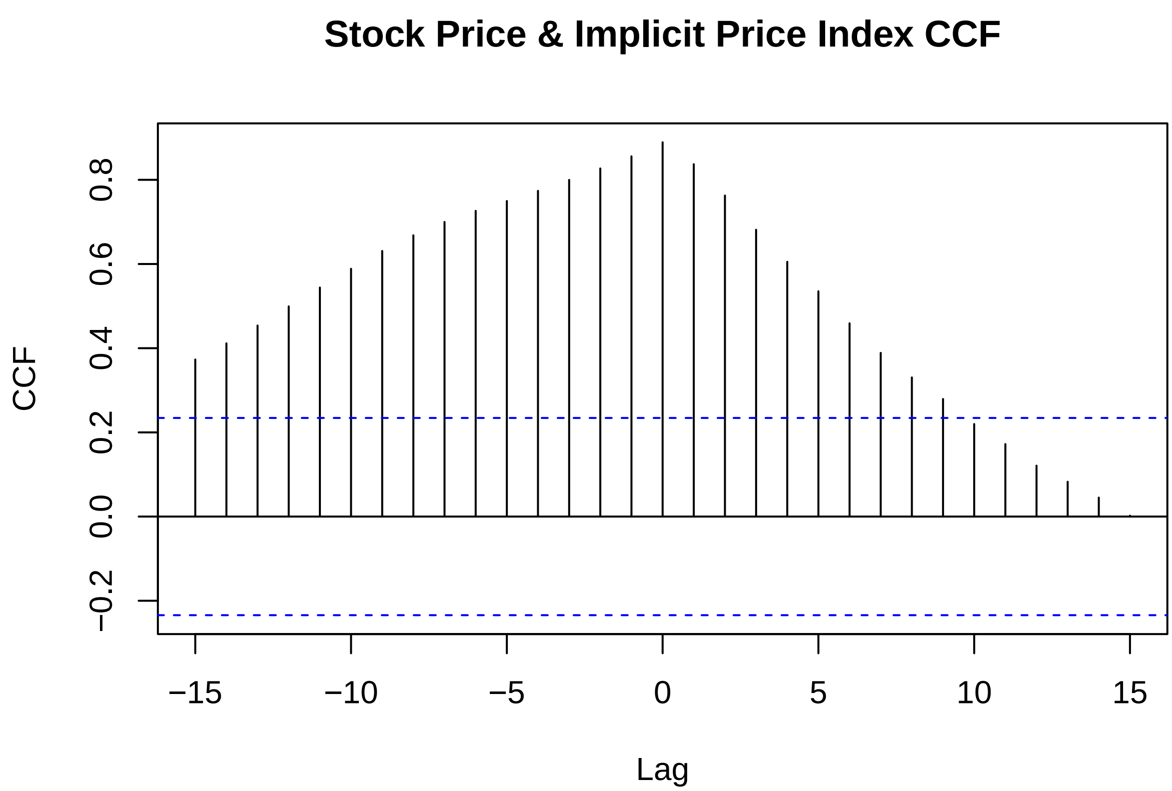
```r
var_model <- VAR(var_data, p = 2)
summary(var_model)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: stock, price_index
## Deterministic variables: const
## Sample size: 68
## Log Likelihood: -351.956
## Roots of the characteristic polynomial:
##  1.02 0.826 0.4388 0.2403
## Call:
## VAR(y = var_data, p = 2)
##
##
## Estimation results for equation stock:
## ======================================
## stock = stock.l1 + price_index.l1 + stock.l2 + price_index.l2 + const
##
##                Estimate Std. Error t value Pr(>|t|)
## stock.l1        1.06746    0.14526   7.349 4.92e-10 ***
## price_index.l1 -0.03871    0.16365  -0.237   0.8138
## stock.l2       -0.16020    0.14668  -1.092   0.2789
## price_index.l2  0.14710    0.17083   0.861   0.3925
## const          -3.11368    1.49894  -2.077   0.0419 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 4.036 on 63 degrees of freedom
## Multiple R-Squared: 0.9785,  Adjusted R-squared: 0.9772
## F-statistic: 717.7 on 4 and 63 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation price_index:
## ============================================
## price_index = stock.l1 + price_index.l1 + stock.l2 + price_index.l2 + const
##
##                Estimate Std. Error t value Pr(>|t|)
## stock.l1        0.11383    0.09998   1.139 0.259190
## price_index.l1  1.45803    0.11264  12.945  < 2e-16 ***
## stock.l2       -0.08010    0.10096  -0.793 0.430523
## price_index.l2 -0.48117    0.11758  -4.092 0.000124 ***
## const           1.28627    1.03168   1.247 0.217097
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 2.778 on 63 degrees of freedom
## Multiple R-Squared: 0.9917,  Adjusted R-squared: 0.9911
## F-statistic:  1873 on 4 and 63 DF,  p-value: < 2.2e-16
##
```

```
##
##
## Covariance matrix of residuals:
##               stock price_index
## stock       16.2925     -0.8412
## price_index -0.8412      7.7181
##
## Correlation matrix of residuals:
##                stock price_index
## stock        1.00000    -0.07502
## price_index -0.07502     1.00000
```

- The model summary of both equations are shown above. Notice that for both equations, the terms of the other variable seems to be insignificant. Next, we plot the CCF plot:

```
ccf(price_index, stock, ylab = "CCF", main = "Stock Price & Implicit Price Index CCF")
```



- The CCF plot between stock price and implicit price index is shown above. We can see that the significant lags are from approximately -15 to 9. This suggests that the implicit price index leads the stock price at most 15 years earlier, then gradually the effect becomes more significant where the price index impose immediate effects on the stock price. Then we can see the implicit price index lags stock price up to 9 years. Next we perform a Granger Causality test:

```r
grangertest(stock~price_index, order = 2)
```

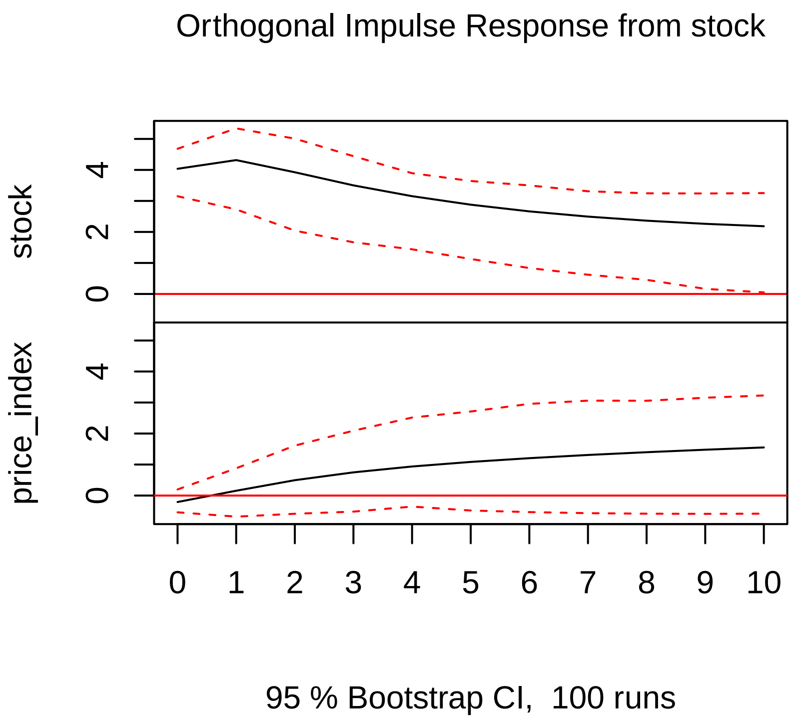
```
## Granger causality test
##
## Model 1: stock ~ Lags(stock, 1:2) + Lags(price_index, 1:2)
## Model 2: stock ~ Lags(stock, 1:2)
##   Res.Df Df      F  Pr(>F)
## 1     63
## 2     65 -2 4.0394 0.02236 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

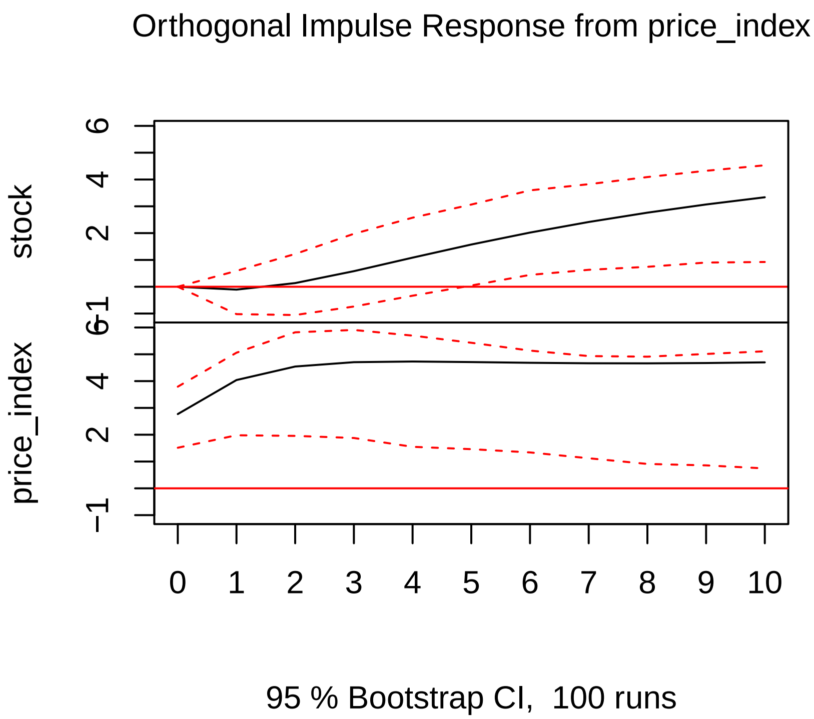
```r
grangertest(price_index~stock, order = 2)
```

```
## Granger causality test
##
## Model 1: price_index ~ Lags(price_index, 1:2) + Lags(stock, 1:2)
## Model 2: price_index ~ Lags(price_index, 1:2)
##   Res.Df Df      F Pr(>F)
## 1     63
## 2     65 -2 1.2058 0.3063
```

- The test result suggests that shocks and effects of price index is likely to have an impact on stock price, since the p-value is less than 0.05. However, there is no statistical evidence suggesting the opposite, since the p-value for that is very large. Next we plot the IRF plots:

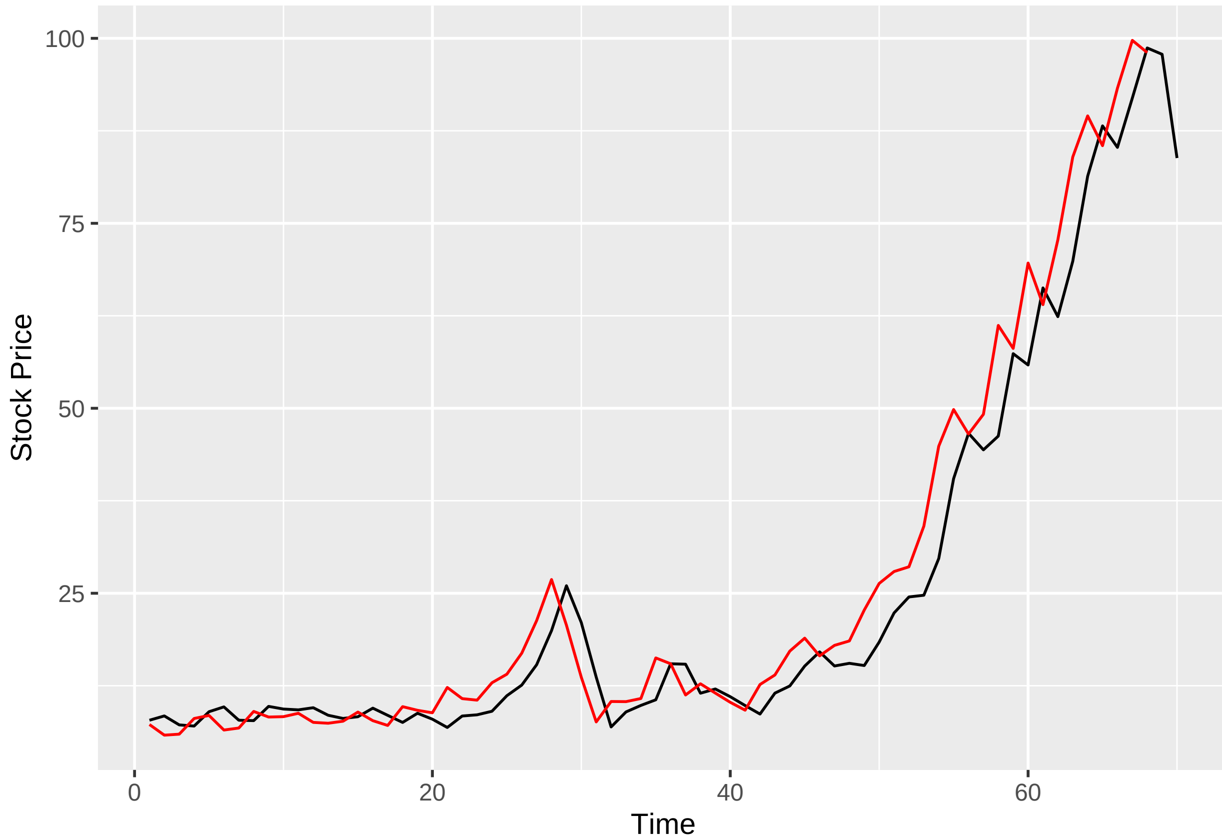
```r
plot(irf(var_model))
```


Orthogonal Impulse Response from stock
stock
price_index
0
2
4
0 1 2 3 4 5 6 7 8 9 10
95 % Bootstrap CI, 100 runs

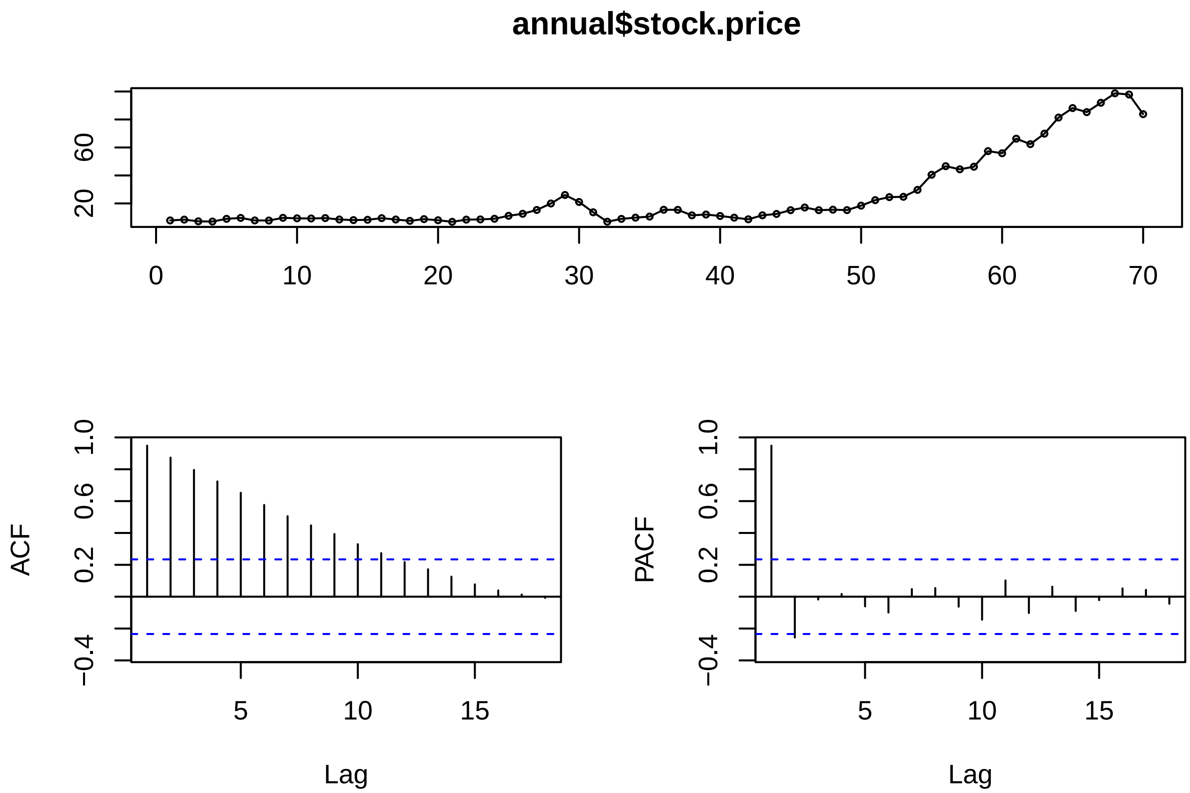


- The first IRF plot shows effects from stock price. We can see that the effect of a shock of stock price on itself reaches peak at around lag 1, then gradually decreases and becomes statistically insignificant at lag 7. Its effect on implicit index, though, seems to be insignificant at the beginning. Similarly, for price index, the effect of a shock on itself reaches the highest at around lag3, and remains at that level ever after. Its effect on stock price, seems to be insignificant at the very beginning.

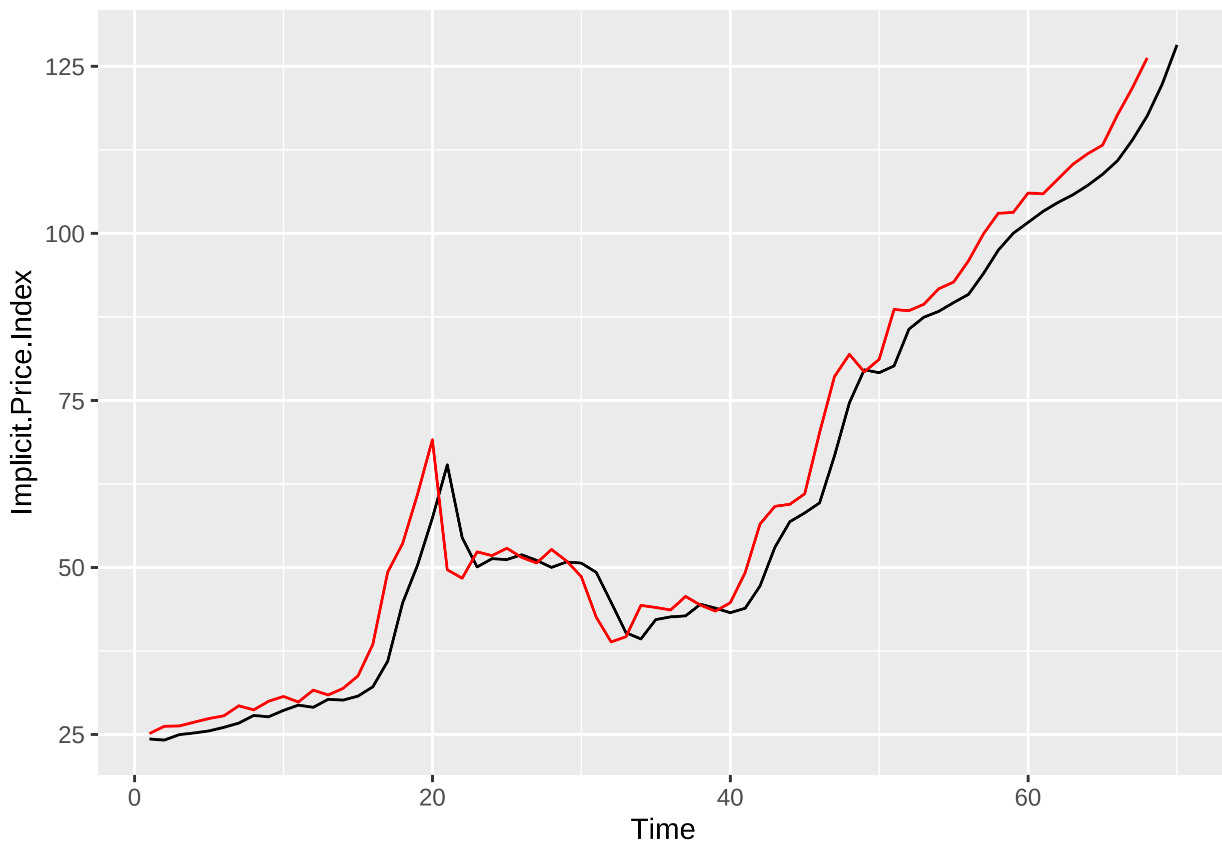
```r
fitted_var <- fitted(var_model)
# plot stock price
ggplot() +
  geom_line(aes(y = annual$stock.price, x = 1:70)) +
  geom_line(aes(y = fitted_var[, 1], x = 1:68), col = "red") +
  labs(x = "Time", y = "Stock Price")
```



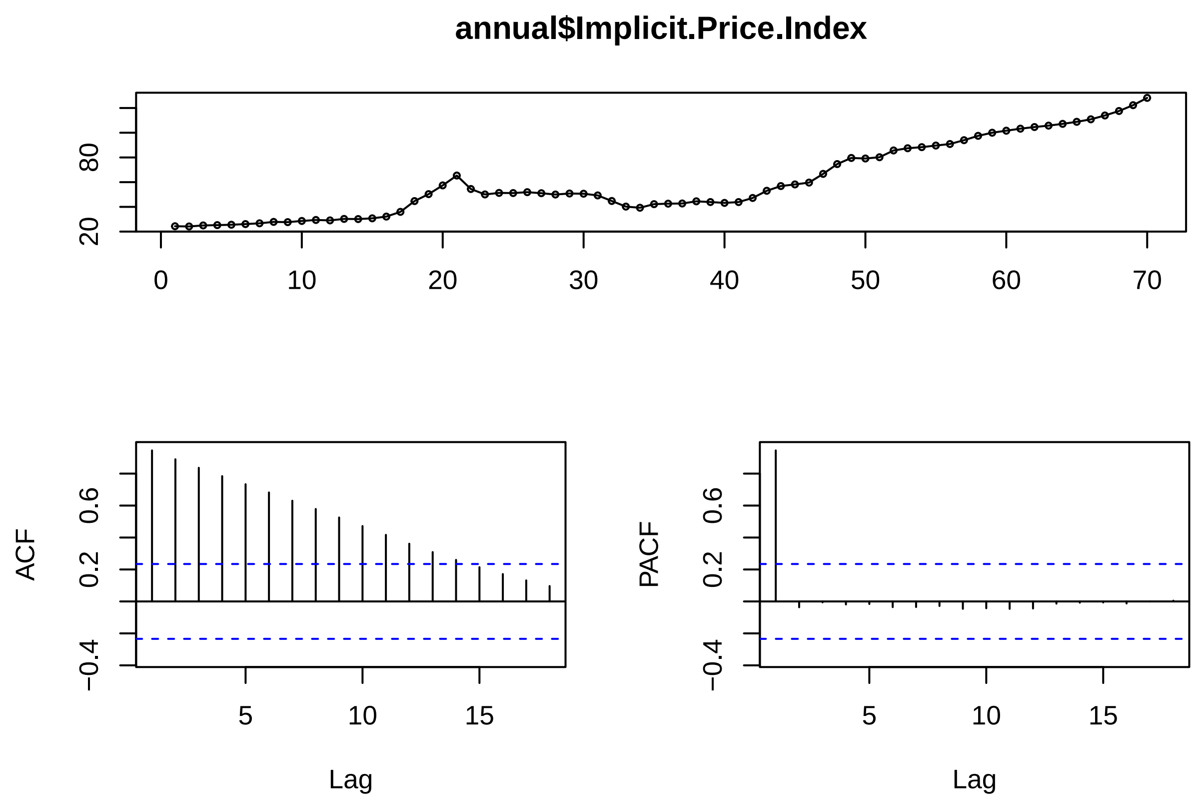

```
tsdisplay(annual$stock.price)
```




```
# plot implicit price index
ggplot() +
  geom_line(aes(y = annual$Implicit.Price.Index, x = 1:70)) +
  geom_line(aes(y = fitted_var[, 2], x = 1:68), col = "red") +
  labs(x = "Time", y = "Implicit.Price.Index")
```




```
tsdisplay(annual$Implicit.Price.Index)
```




```r
# splitting the data
var_tr <- var_data[1:46,]
var_ts <- var_data[47:70,]

# construct model with the training data
var_train <- VAR(var_tr, p = 2)
summary(var_train)
```

```
##
## VAR Estimation Results:
## =========================
## Endogenous variables: stock, price_index
## Deterministic variables: const
## Sample size: 44
## Log Likelihood: -201.708
## Roots of the characteristic polynomial:
## 0.8226 0.6735 0.6735 0.6112
## Call:
## VAR(y = var_tr, p = 2)
##
##
## Estimation results for equation stock:
## ======================================
## stock = stock.l1 + price_index.l1 + stock.l2 + price_index.l2 + const
##
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## stock.l1        1.15273    0.13759   8.378 2.99e-10 ***
## price_index.l1  0.04692    0.09982   0.470 0.640963
## stock.l2       -0.52142    0.13646  -3.821 0.000466 ***
## price_index.l2  0.01709    0.10128   0.169 0.866866
## const           1.43925    1.35660   1.061 0.295252
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 2.158 on 39 degrees of freedom
## Multiple R-Squared: 0.7468,  Adjusted R-squared: 0.7208
## F-statistic: 28.75 on 4 and 39 DF,  p-value: 3.636e-11
##
##
## Estimation results for equation price_index:
## ============================================
## price_index = stock.l1 + price_index.l1 + stock.l2 + price_index.l2 + const
##
##                 Estimate Std. Error t value Pr(>|t|)
## stock.l1         0.2507     0.1930   1.299  0.20148
## price_index.l1   1.3662     0.1400   9.759 5.08e-12 ***
## stock.l2        -0.2938     0.1914  -1.535  0.13279
## price_index.l2  -0.4277     0.1420  -3.011  0.00455 **
## const            3.4425     1.9025   1.809  0.07809 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 3.026 on 39 degrees of freedom
## Multiple R-Squared: 0.9345,  Adjusted R-squared: 0.9278
## F-statistic: 139.1 on 4 and 39 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##               stock price_index
## stock        4.6571     -0.8991
## price_index -0.8991      9.1596
##
## Correlation matrix of residuals:
##               stock price_index
## stock        1.0000     -0.1377
## price_index -0.1377      1.0000
```

```r
# training RMSE
((var_rmse_tr <- sqrt(mean(resid(var_train)^2))))
```

```
## [1] 2.474531
```

```r
# testing RMSE
var_fc <- predict(var_train, n.ahead = 24)
(rmse_stock <- sqrt(mean((var_fc$fcst$stock - var_ts$stock)^2)))
```

```
## [1] 50.96014
```

```
(rmse_price_index <- sqrt(mean((var_fc$fcst$price_index - var_ts$price_index)^2)))
```

```
## [1] 60.29957
```

- The testing RMSE of the equation that stock price is the y-variable is 50.96014, and the testing RMSE of the equation that the implicit price index is the y-variable is 60.29957. Hence, according to RMSE, we should choose the equation with the stock price to be the y-variable because it has a smaller RMSE.

```
# AIC and BIC of the model
AIC(var_train)
```

```
## [1] 423.4165
```

```
BIC(var_train)
```

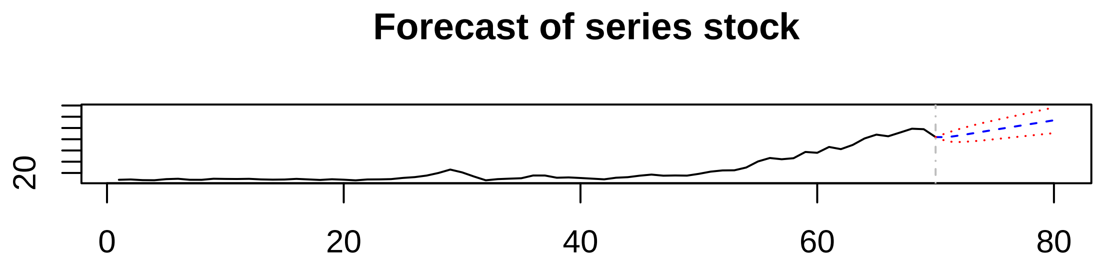
```
## [1] 441.2584
```

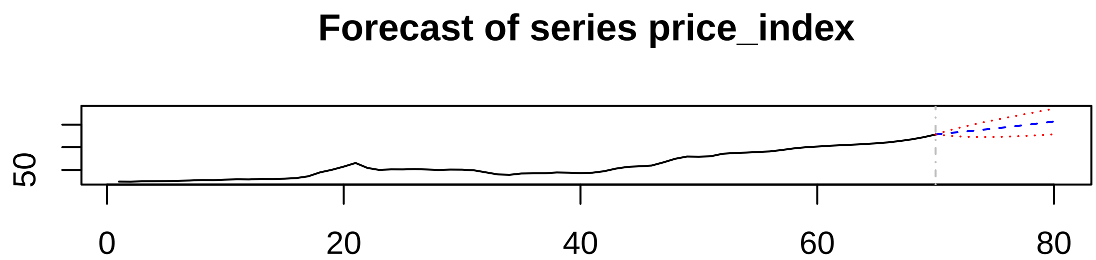
- Next we forecast 10 step ahead using both models:

```
var_forecast <- predict(var_model, n.ahead = 10)
var_forecast
```

```
## $stock
##            fcst    lower     upper        CI
##  [1,]  83.71398 75.80279  91.62517  7.911191
##  [2,]  86.60510 75.02001  98.19019 11.585091
##  [3,]  90.03528 76.12644 103.94413 13.908847
##  [4,]  93.49600 77.94431 109.04770 15.551696
##  [5,]  96.90925 80.04017 113.77833 16.869079
##  [6,] 100.28249 82.22940 118.33558 18.053090
##  [7,] 103.63478 84.42958 122.83997 19.205192
##  [8,] 106.98309 86.60752 127.35866 20.375567
##  [9,] 110.34107 88.75533 131.92682 21.585742
## [10,] 113.71969 90.87782 136.56157 22.841874
##
## $price_index
##           fcst    lower    upper        CI
##  [1,] 131.0776 125.6325 136.5227  5.445075
##  [2,] 133.5258 123.9125 143.1391  9.613345
##  [3,] 136.0532 122.9124 149.1940 13.140782
##  [4,] 138.7190 122.5974 154.8406 16.121596
##  [5,] 141.5090 122.8213 160.1967 18.687723
##  [6,] 144.4055 123.4545 165.3565 20.951005
##  [7,] 147.3968 124.4021 170.3916 22.994732
##  [8,] 150.4759 125.5974 175.3545 24.878525
##  [9,] 153.6387 126.9939 180.2835 26.644816
## [10,] 156.8825 128.5584 185.2066 28.324086
```

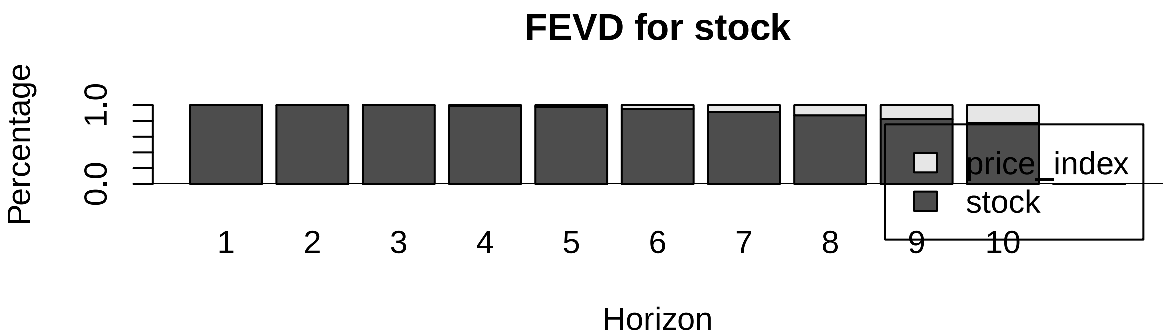
```r
plot(var_forecast)
```

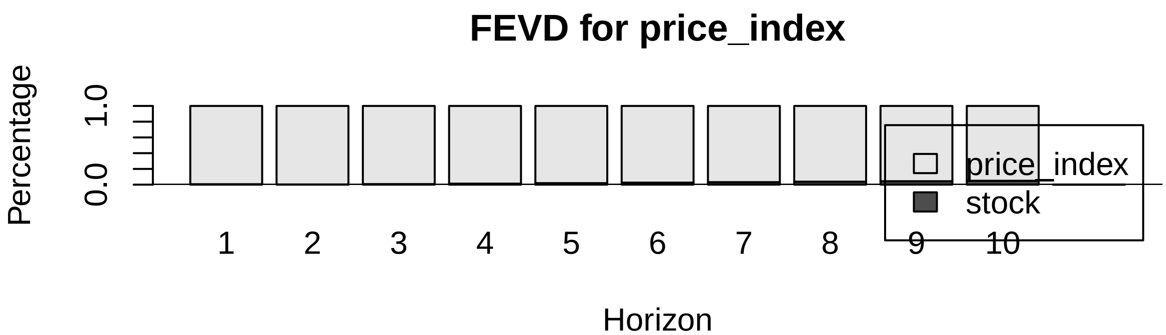




- The calculated forecast and plot are shown above.

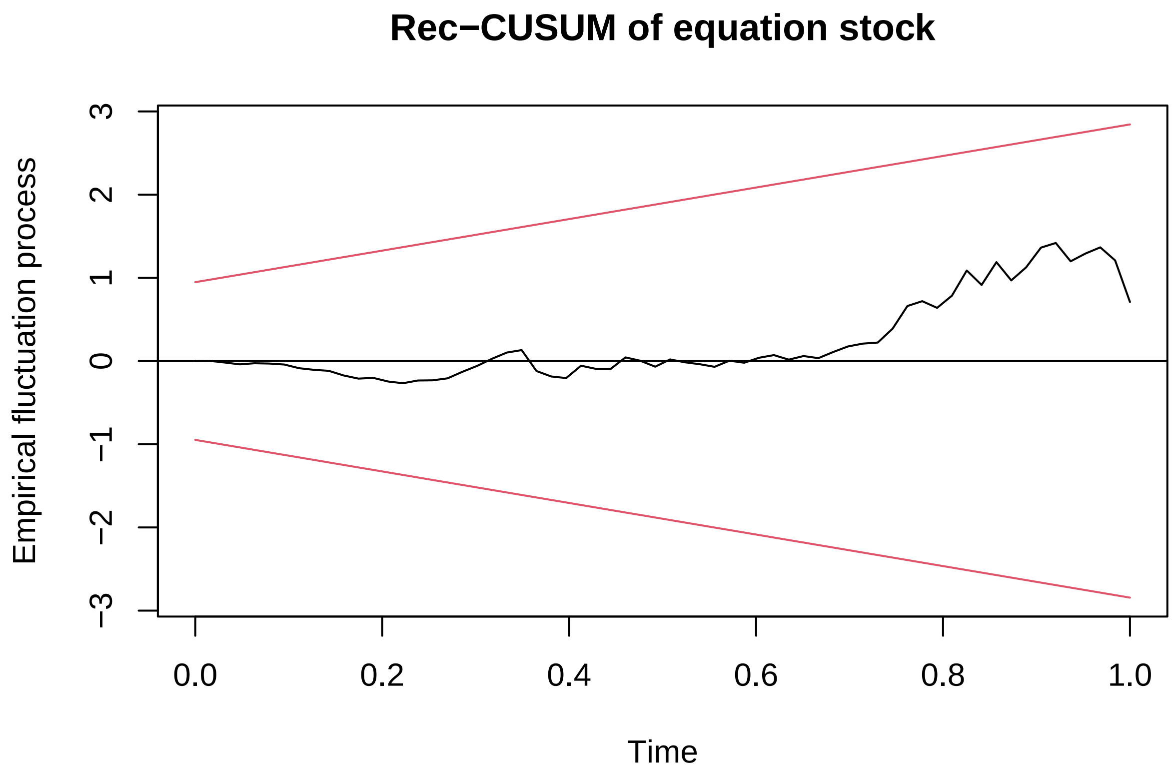
```r
plot(fevd(var_model))
```

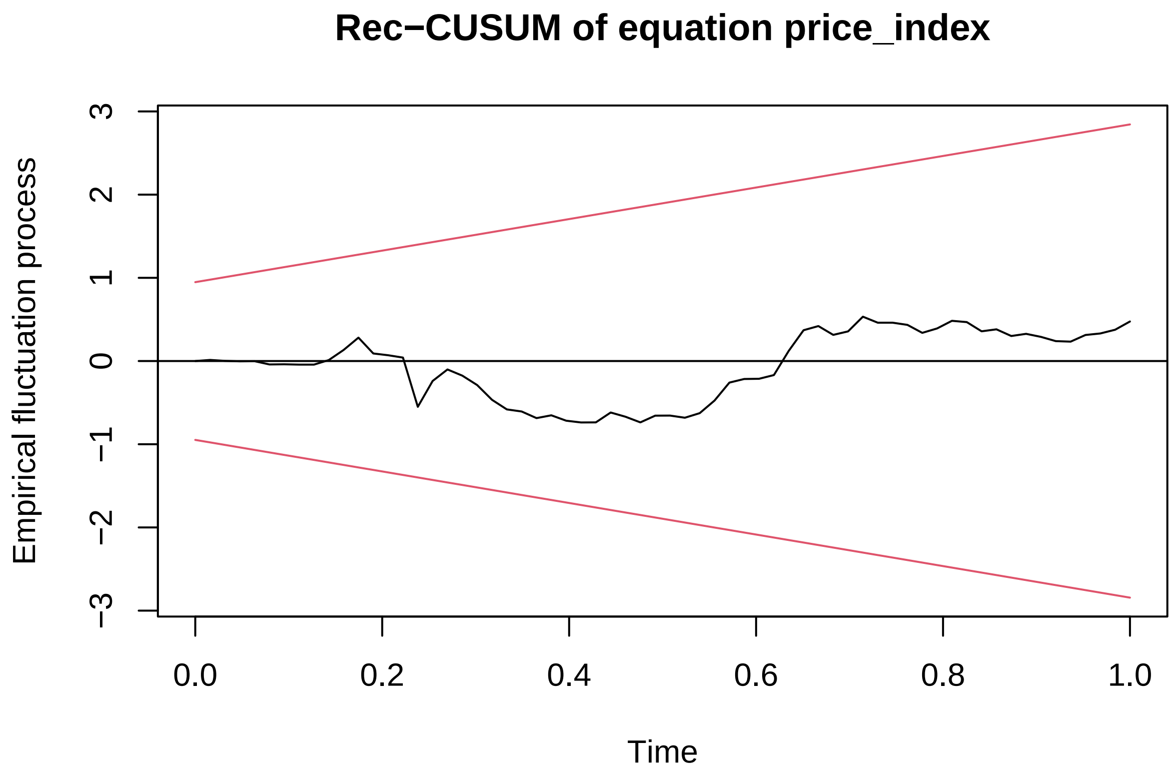




- We see that for stock price, at first 100% of variations come from itself's shocks. then at around horizon 5 or 6 we see gradually an increasing share of variation from implicit price index, which comes to at around 20% at horizon 10. For implicit price index, we see that stock price only takes a very little portion of variation starting horizon 5, and most of the variation comes from the shock from itself.

```
plot(stability(var_model, type = "Rec-CUSUM"), plot.type="single")
```


Rec−CUSUM of equation stock
Empirical fluctuation process
3
2
1
0
−1
−2
−3
0.0
0.2
0.4
0.6
0.8
1.0
Time



- We see that for both equations, the cumulative error fluctuates around 0 through time, therefore we conclude that neither model is broken across time.

Part 6. Summary.

- In this project, we focused on four variables: annual US stock price, annual US bond yield, annual US implicit price index, and annual US velocity of money, all of them last from 1990 to 1969. We first provide descriptive statistics on each of the variable's distribution and their correlation, concluding that there are considerable relationships among all variables. Then we suggest AR(1) and AR(2) models for each variable and picked the better model for each based on residual plots, testing RMSE, as well as AIC and BIC.
- We've also considered ARDL models for each pair of variables: bond yield vs. stock price, and velocity of money vs. implicit index. We've constructed ARDL(1,1) and ARDL(2,2) for each pair, then picked the best one for each based on testing RMSE, AIC, and BIC.
- Then we consider a VAR(2) model for stock price and implicit price index, where we believe they have some shared dynamics. We've implemented several diagnostics including the CCF plot, the IRF plot, and conducted the Granger Causality test. Our findings suggest potentially there's more dynamics from implicit price index to stock price than the opposite. We've produced 10-step forecast based on that, and then also shown the FEVD and CUSUM plot as a measurement of variation and error.
- After evaluating all the models above, we believe ARDL models are the best to consider for each variable. The ARDL models take into consideration of both variables, its testing RMSE is smaller than the AR model ones. Also compared to VAR models, the ARDL also considers predictor variables, where VAR models' diagnostics show mixed results and insignificant values, therefore we suggest the ARDL models for each variable.