# Econ 144 Project 2

Theo Teske

2023-02-22

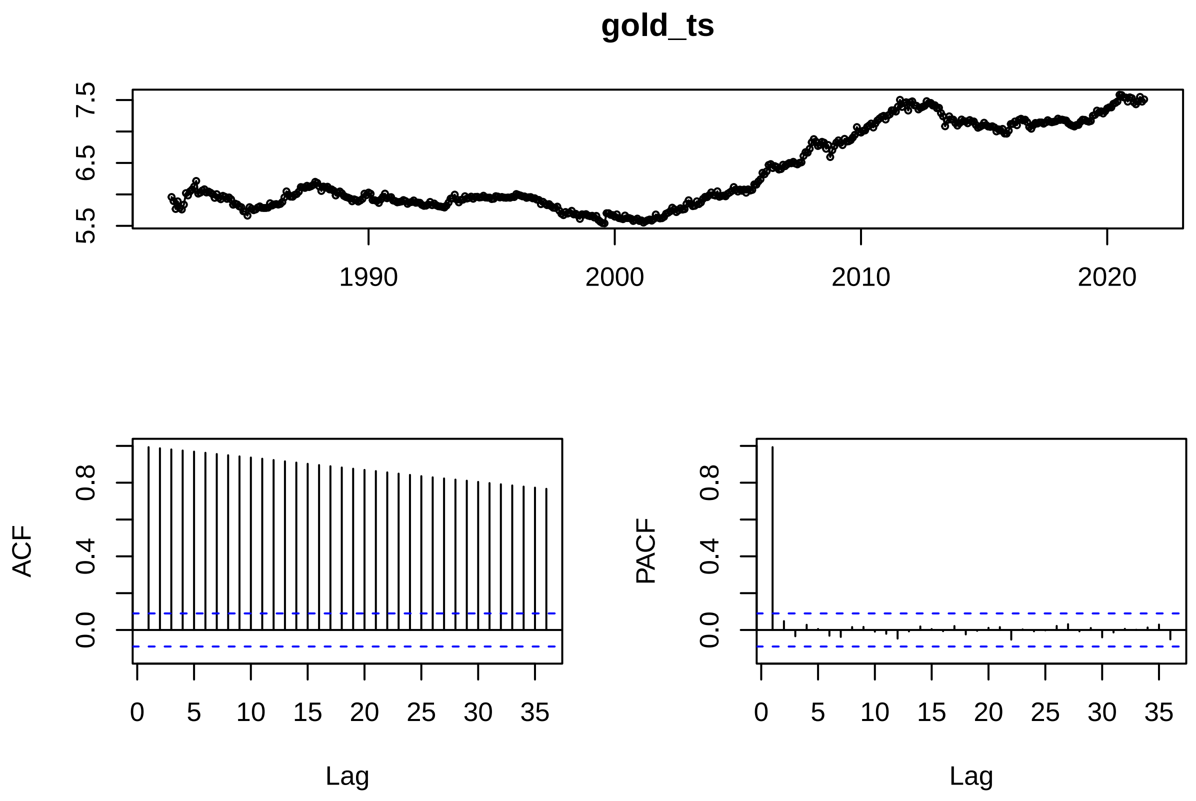
## I. Introduction

Purchasing gold has long been seen in the financial world as a hedge against inflation, as gold is likely to retain its purchasing power for long periods of time. Thus, when other assets may experience price fluctuations, the price of gold is thought to remain more stable. In order to explore this notion, the 10-Year Expected Inflation from the Federal Reserve Bank of Cleveland, in percent, is compared against the log of the price of gold in USD from the World Gold Council. The Federal Reserve Bank of Cleveland estimates the expected rate of inflation over the next 30 years along with the inflation risk premium, the real risk premium, and the real interest rate. Their estimates are calculated with a model that uses Treasury yields, inflation data, inflation swaps, and survey-based measures of inflation expectations. Both datasets are monthly, and we consider data from both from January 1982 to July 2021.

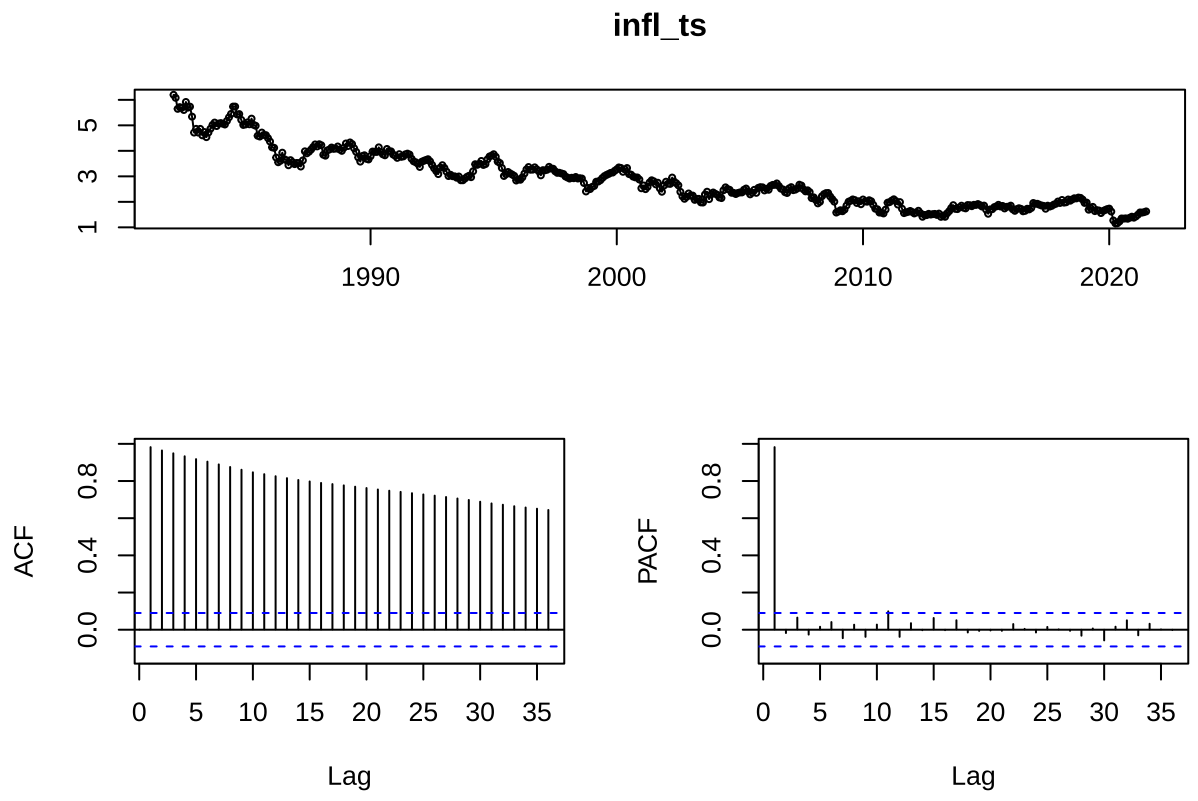
## II. Results

```r
gold <- read.table("C:/Users/Theo/Downloads/1979-2021.csv", header=TRUE,
   sep=",")

infl <- read.table("C:/Users/Theo/Downloads/EXPINF10YR.csv", header=TRUE,
   sep=",")

gold_ts_nolog <- ts(gold$United.States.USD.[37:511], start=1982, freq=12)
gold_ts <- log(gold_ts_nolog)
infl_ts <- ts(infl$EXPINF10YR[1:475], start=1982, freq=12)
```

```r
forecast::tsdisplay(gold_ts, plot.type="partial")
```

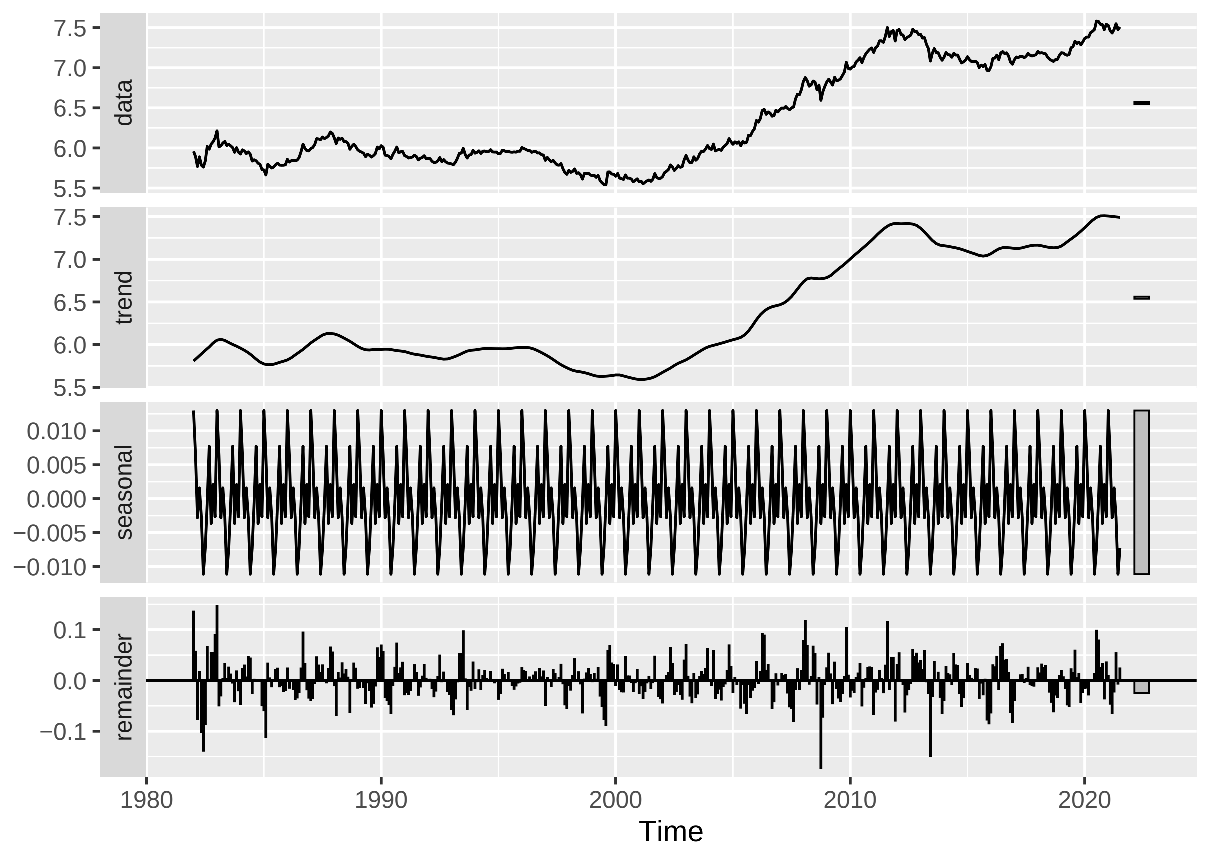


```
forecast::tsdisplay(infl_ts, plot.type="partial")
```

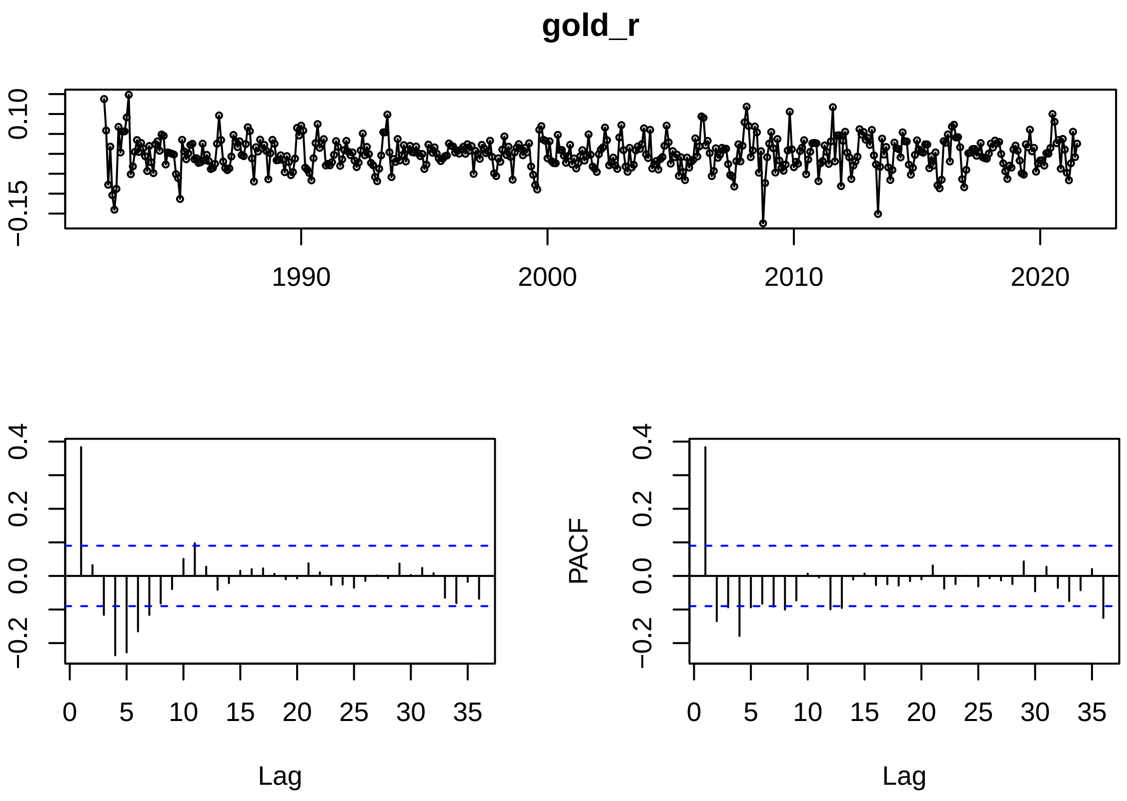


Clearly neither time series is covariance stationary initially, but we can perform an STL decomposition to remove the trend and seasonality influencing the ACF and PACF plots. An additive decomposition makes sense because neither time series looks like its volatility changes over time.

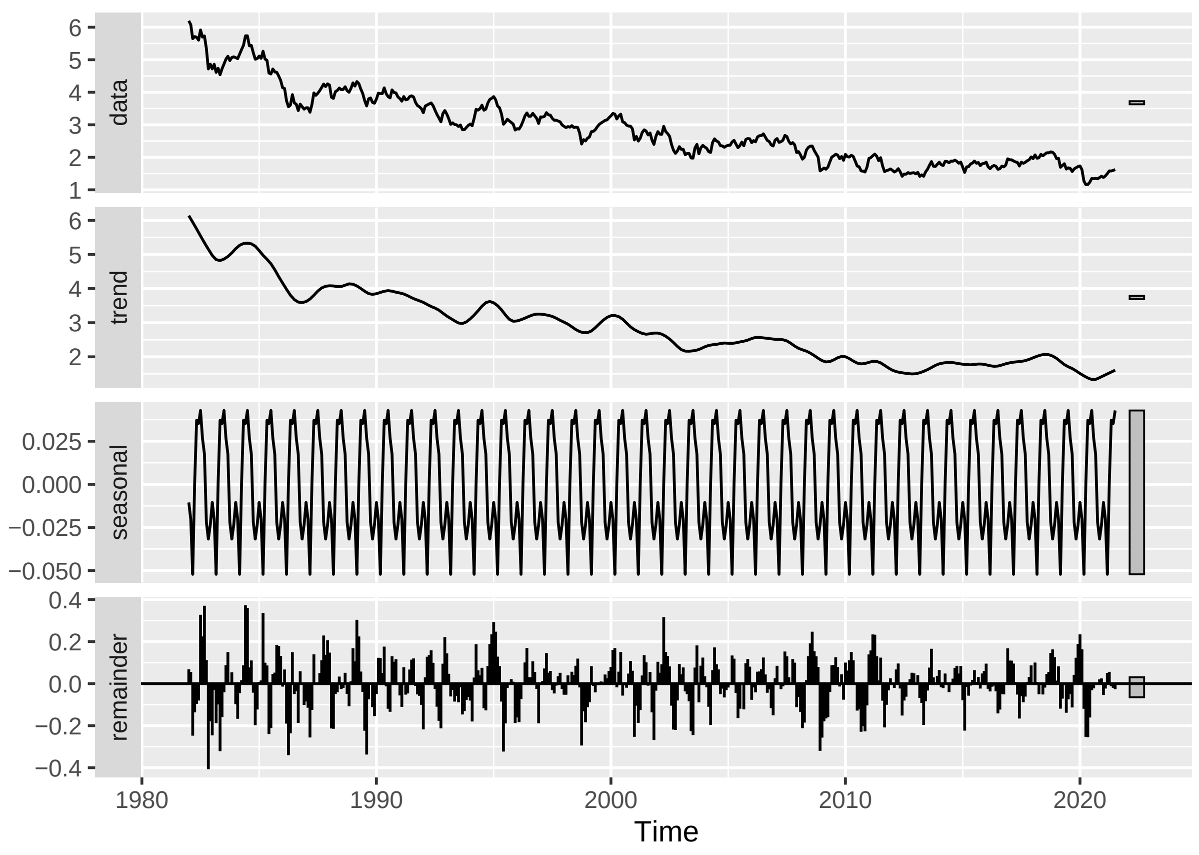
```
gold_stl <- stl(gold_ts, "periodic")
autoplot(gold_stl)
```



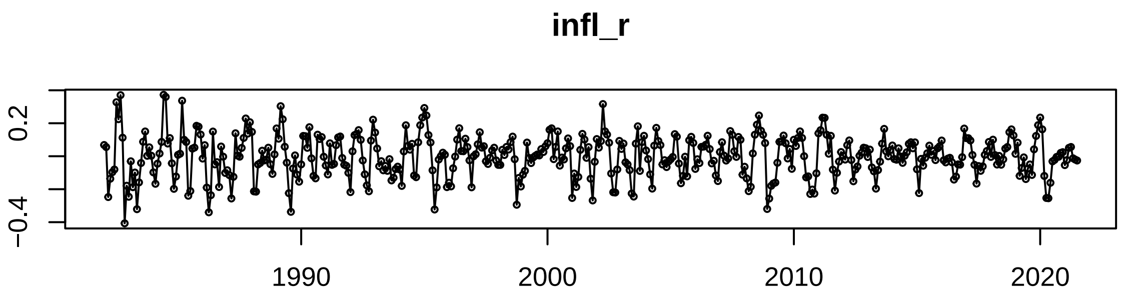
```
seas_gold = gold_stl$time.series[,1]
trend_gold = gold_stl$time.series[,2]
gold_r <- gold_stl$time.series[,3]

forecast::tsdisplay(gold_r, plot.type="partial")
```

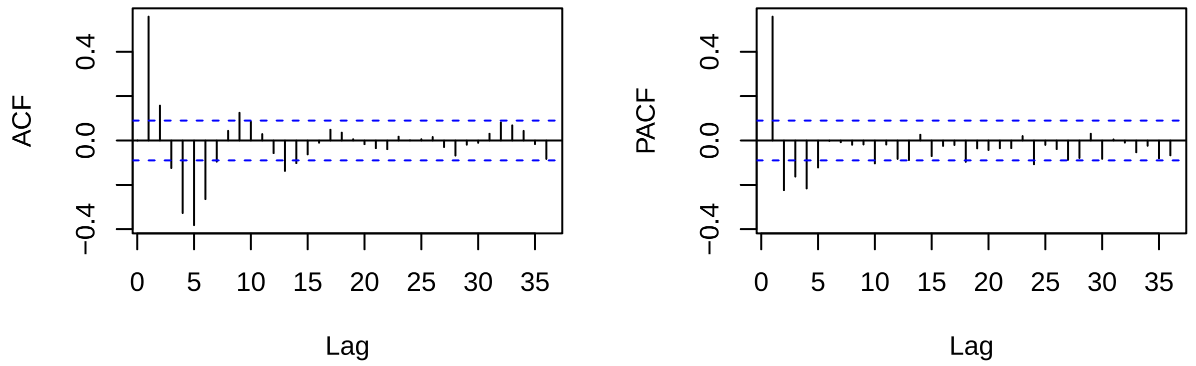


```r
infl_stl <- stl(infl_ts, "periodic")
autoplot(infl_stl)
```



```
seas_infl = (infl_stl$time.series[,1])
trend_infl = (infl_stl$time.series[,2])
#infl_r <- infl_ts-trend_infl-seas_infl
infl_r <- infl_stl$time.series[,3]

forecast::tsdisplay(infl_r, plot.type="partial")
```





Both time series look like they have trend, seasonal, and cyclic components. Based on the ACF and PACF of the gold time series' residuals, it looks like the cyclic component could be modelled by an ARMA(2,1) model, likely after differencing to remove the remaining seasonality visible in the ACF. For the inflation time series, again it looks like there's some seasonality left over after the STL decomposition judging by the ACF of the residuals, but after differencing an AR(2) model or an ARMA(2,1) model could be appropriate.

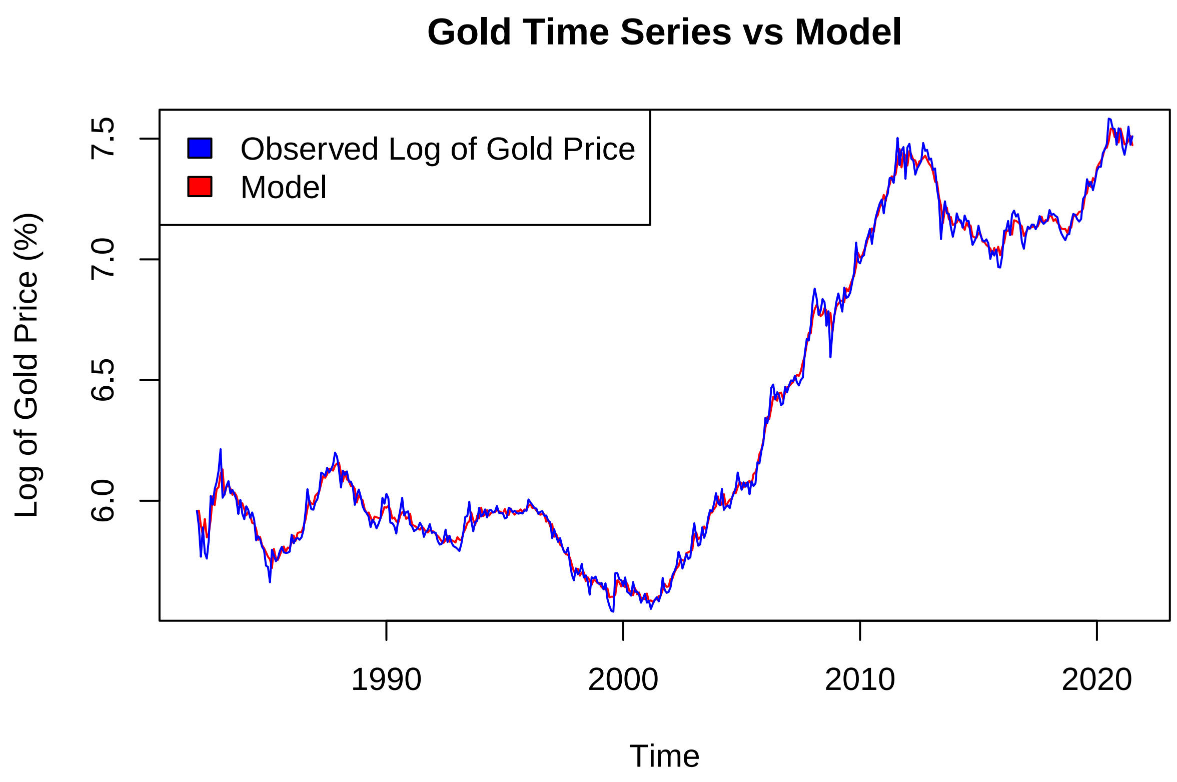
```
gold_ar <- Arima(gold_r, order=c(2,1,1), include.drift=TRUE)
summary(gold_ar)
```

```
## Series: gold_r
## ARIMA(2,1,1) with drift
##
## Coefficients:
##          ar1      ar2      ma1  drift
##       0.4477  -0.1362  -1.0000      0
## s.e.  0.0460   0.0461   0.0054      0
##
## sigma^2 = 0.00136:  log likelihood = 890.88
## AIC=-1771.77   AICc=-1771.64   BIC=-1750.96
##
## Training set error measures:
##                        ME       RMSE        MAE      MPE     MAPE      MASE
## Training set -0.001077867 0.03668351 0.02757147 114.2431 223.6189 0.6532088
##                      ACF1
## Training set -0.004263144
```

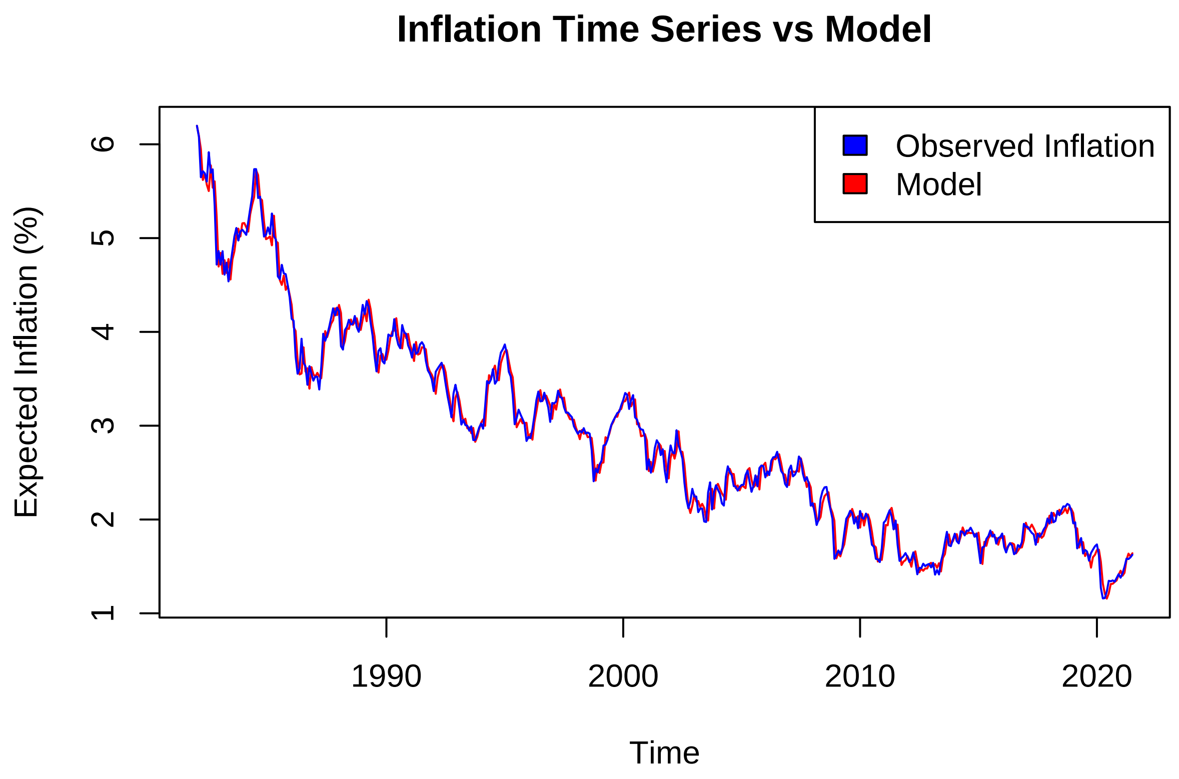
```r
infl_ar <- Arima(infl_r, order=c(2,1,0), include.drift=TRUE)
summary(infl_ar)
```

```
## Series: infl_r
## ARIMA(2,1,0) with drift
##
## Coefficients:
##           ar1      ar2    drift
##       -0.0519  -0.1358  -0.0001
## s.e.   0.0455   0.0457   0.0045
##
## sigma^2 = 0.01356:  log likelihood = 348.22
## AIC=-688.45   AICc=-688.36   BIC=-671.8
##
## Training set error measures:
##                          ME    RMSE       MAE       MPE    MAPE      MASE
## Training set -8.387168e-05 0.11594 0.0876881 -235.9467 675.138 0.6249912
##                     ACF1
## Training set -0.01418671
```

```r
mod1 <- trend_gold+seas_gold+gold_ar$fitted
plot(mod1, col="red", lwd = 1.0, ylab="Log of Gold Price (%)", main="Gold Time Series vs Model")
lines(gold_ts, col="blue", lwd=1.0)
legend(x = "topleft", legend=c("Observed Log of Gold Price", "Model"), fill = c("blue","red"))
```

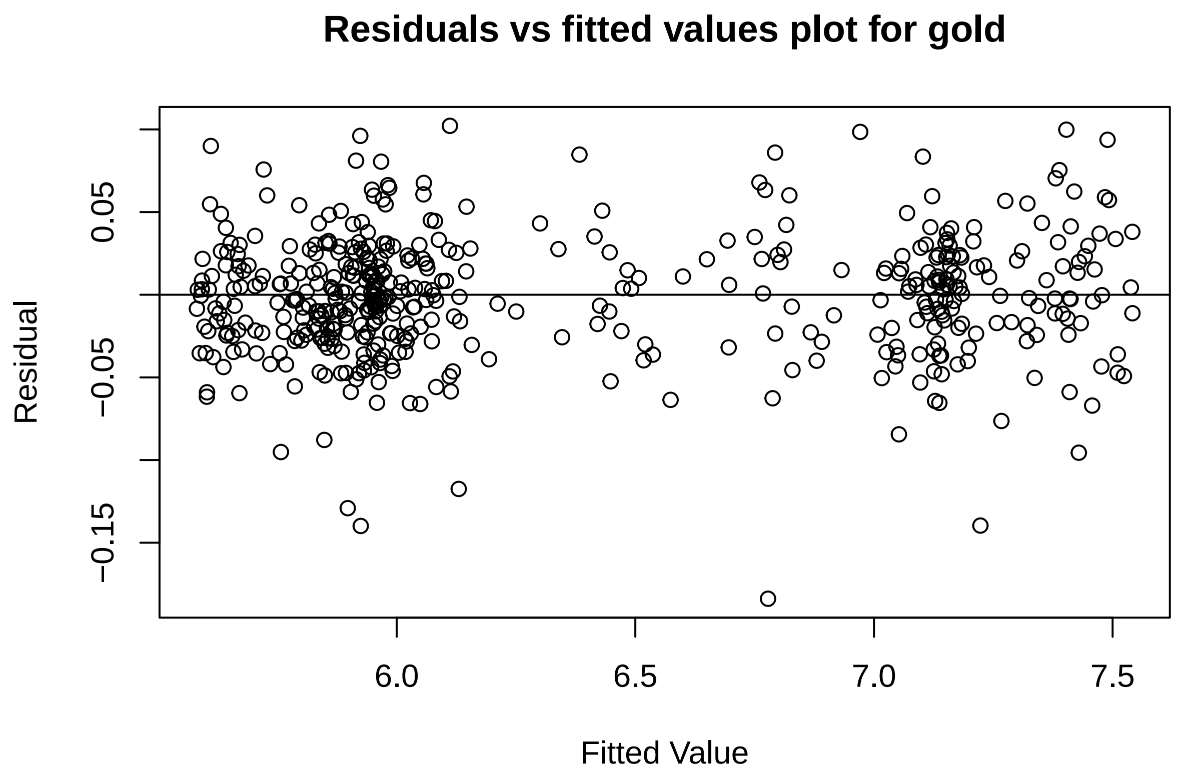


```r
mod2 <- trend_infl+seas_infl+infl_ar$fitted
plot(mod2, col="red", lwd = 1.0, ylab="Expected Inflation (%)", main="Inflation Time Series vs Model")
lines(infl_ts, col="blue", lwd=1.0)
legend(x = "topright", legend=c("Observed Inflation", "Model"), fill = c("blue","red"))
```

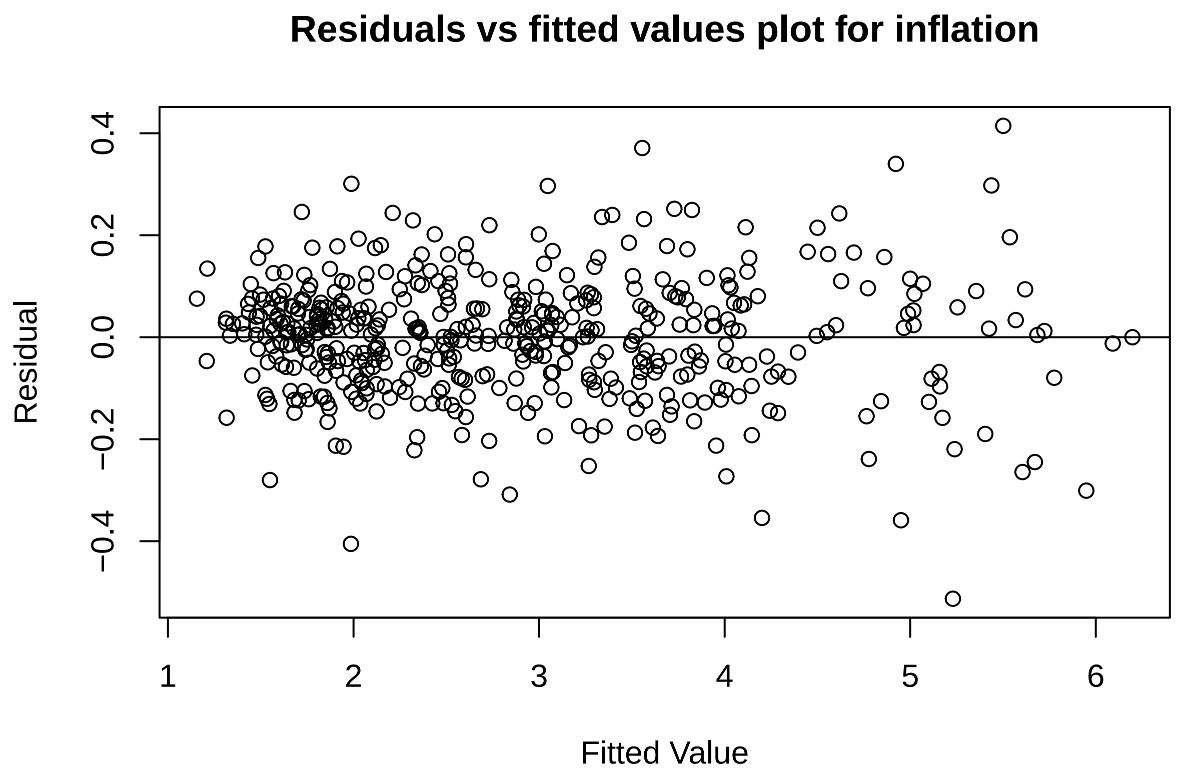


The models fitted above each have a trend component and seasonal component determined by an STL decomposition. The trend component fits a curve to the overall long-term shape of the time series, whereas the seasonal component approximates a pattern that repeats every year. On top of these two components, the remaining cyclic component is estimated by an ARIMA model; for the gold time series, an ARIMA(2,1,1) model was used, and for the inflation time series, an ARIMA(2,1,0) model was used.

```r
resg <- gold_ts - mod1
plot(mod1, resg, main="Residuals vs fitted values plot for gold", xlab="Fitted Value", ylab = "Residual
#add a horizontal line at 0
abline(0,0)
```

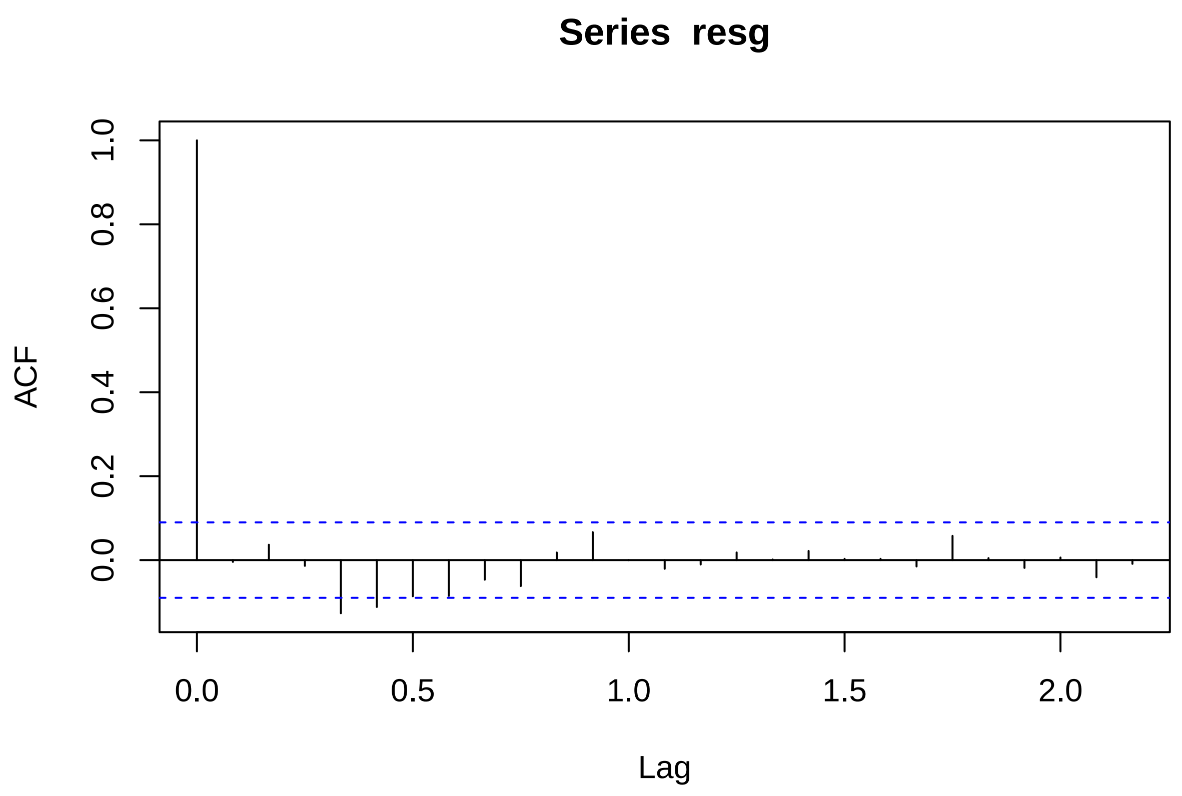


```
resi <- infl_ts - mod2
plot(mod2, resi, main="Residuals vs fitted values plot for inflation", xlab="Fitted Value", ylab = "Resi
#add a horizontal line at 0
abline(0,0)
```

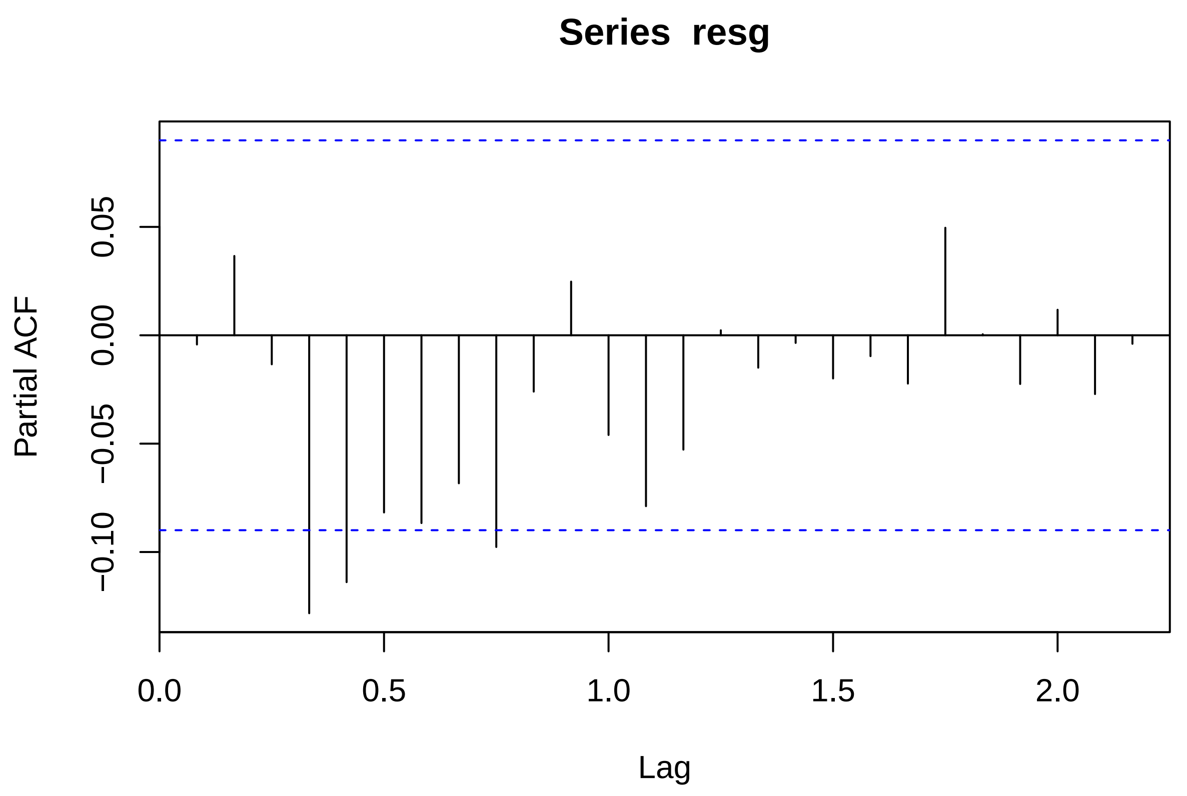


The residuals appear to be random and normally distributed for each time series; they resemble white noise, as desired.

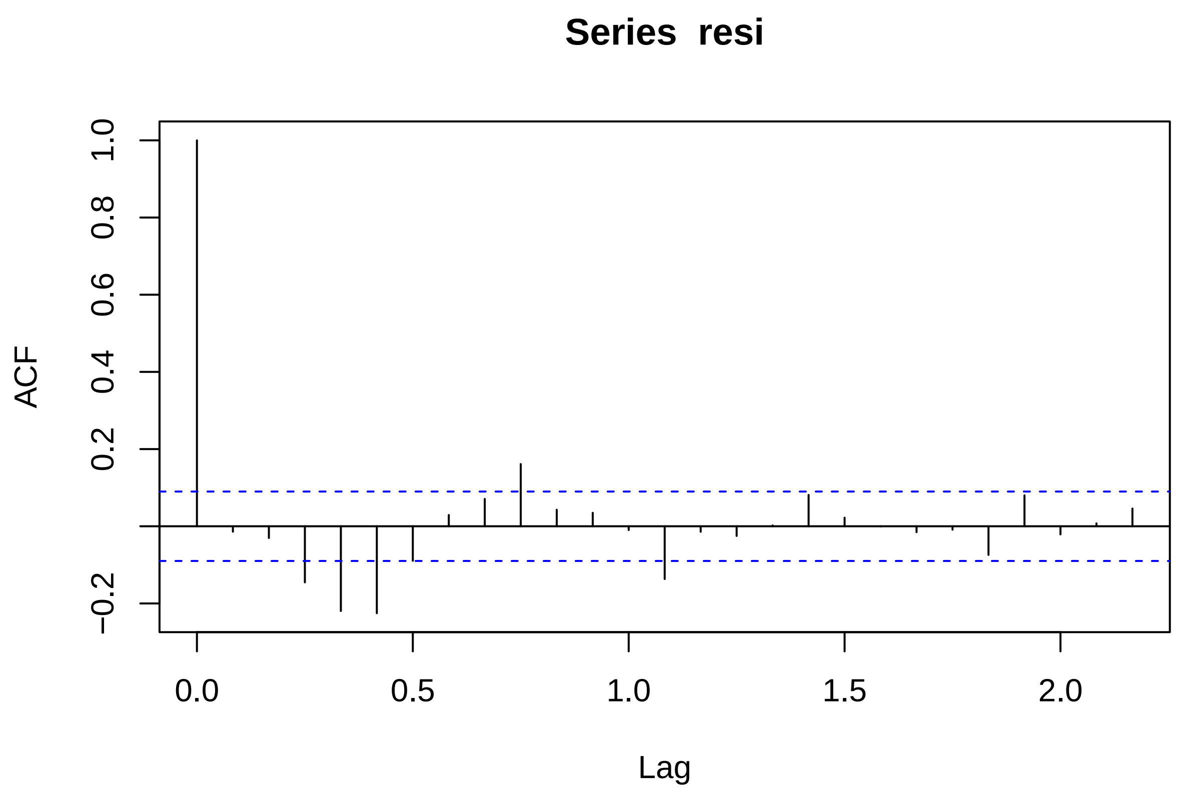
```
acf(resg)
```



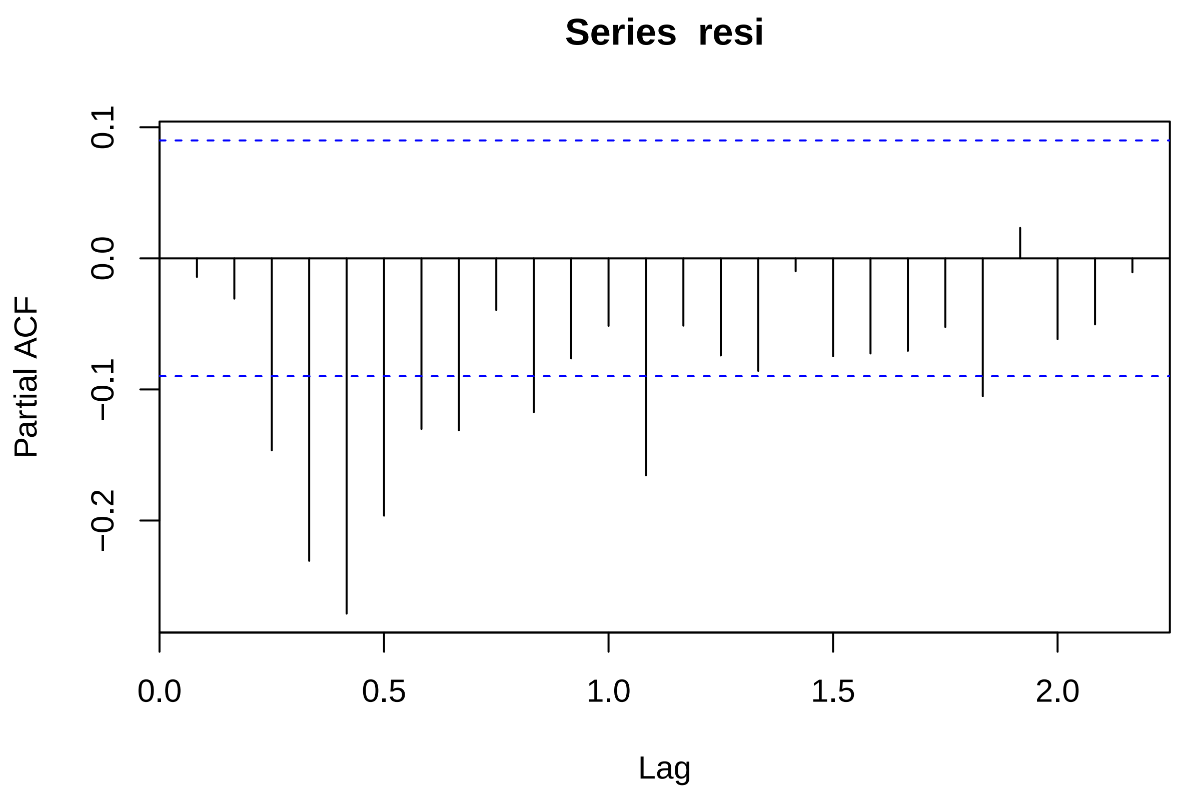
```
pacf(resg)
```



```
acf(resi)
```

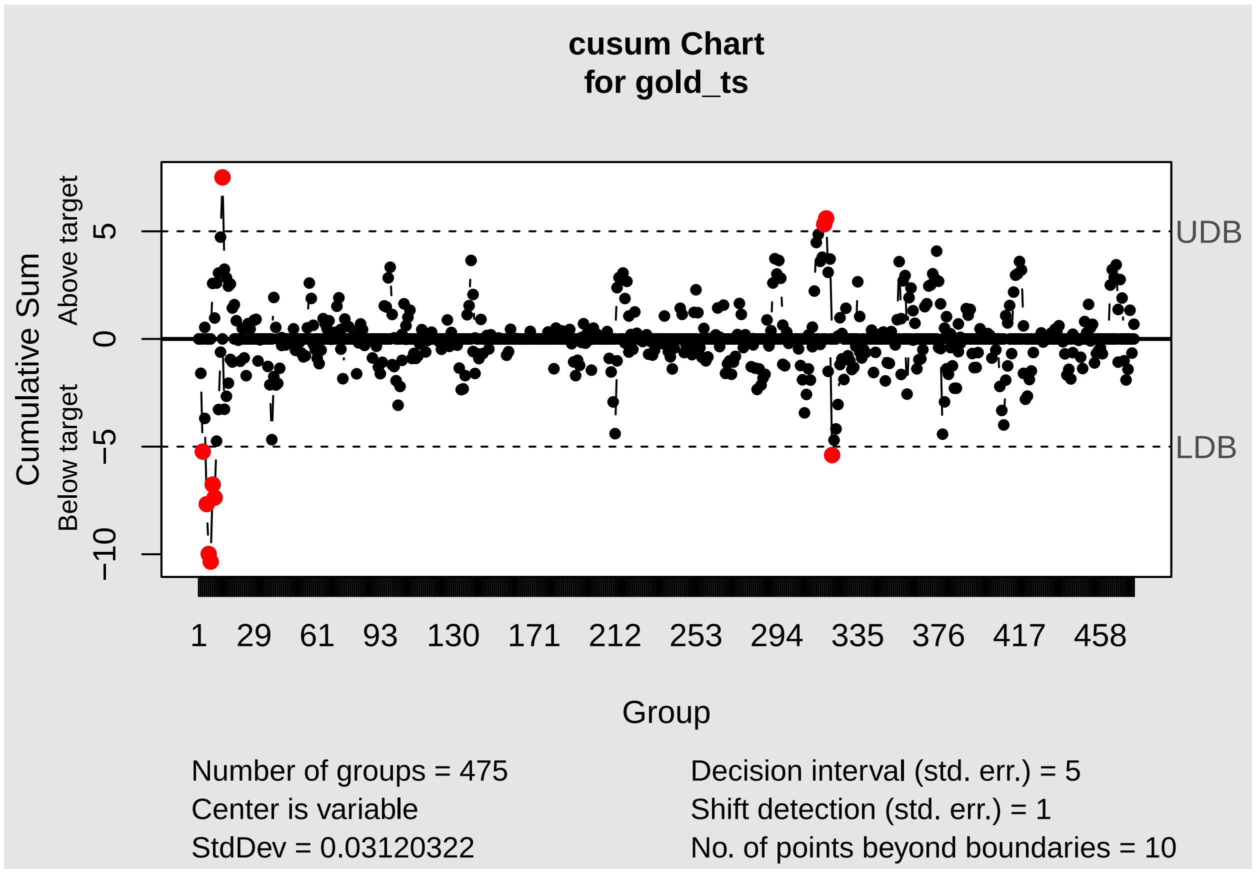


```
pacf(resi)
```



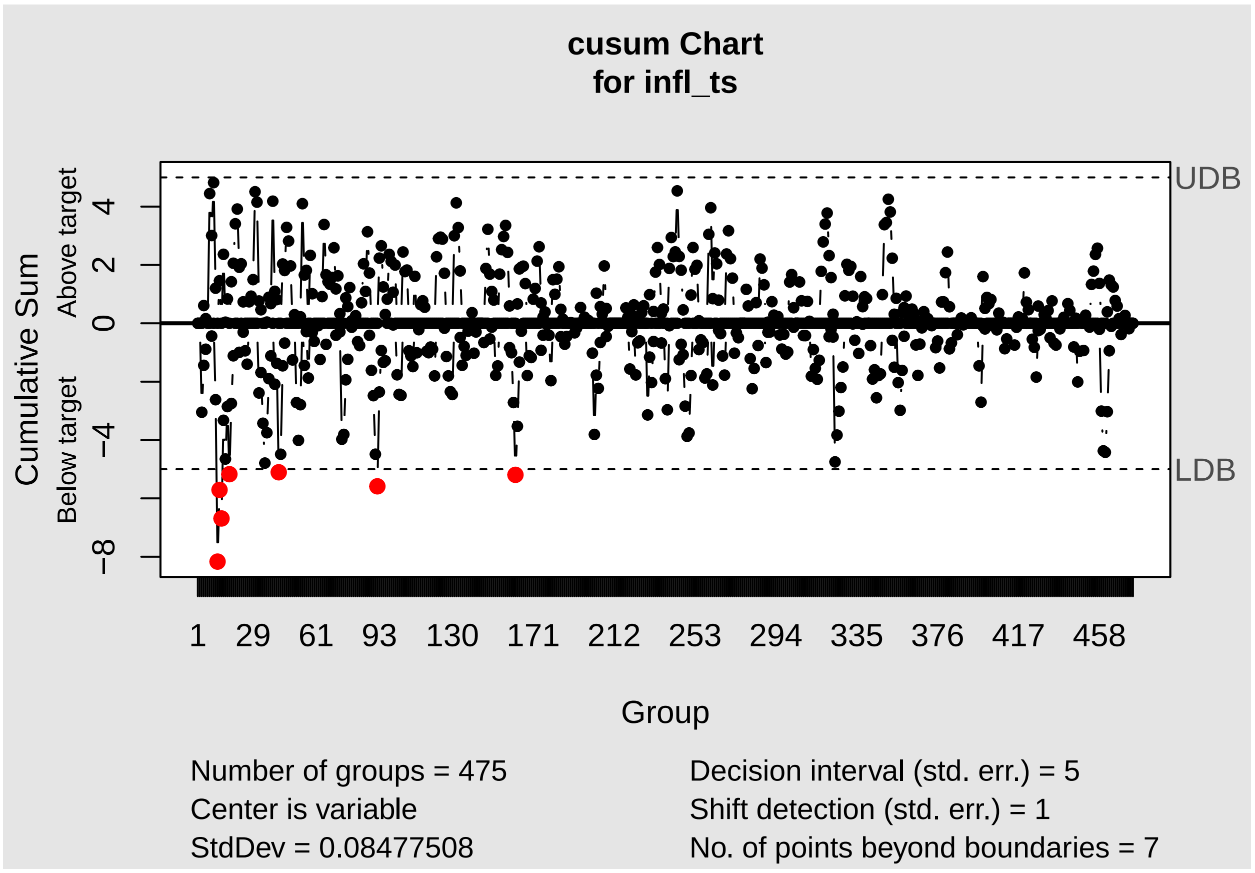
The ACF and PACF for each model's residuals appear to show stationarity for the most part except for the PACF of the inflation model's residuals, where a few serial correlations are visible.

```
qcc::cusum(gold_ts, center = mod1)
```



```
## List of 14
##  $ call              : language qcc::cusum(data = gold_ts, center = mod1)
##  $ type              : chr "cusum"
##  $ data.name         : chr "gold_ts"
##  $ data              : num [1:475, 1] 5.96 5.89 5.77 5.89 5.78 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ statistics        : Named num [1:475] 5.96 5.89 5.77 5.89 5.78 ...
##   ..- attr(*, "names")= chr [1:475] "1" "2" "3" "4" ...
##  $ sizes             : int [1:475] 1 1 1 1 1 1 1 1 1 1 ...
##  $ center            : Time-Series [1:475] from 1982 to 2022: 5.96 5.96 5.9 5.86 5.92 ...
##  $ std.dev           : num 0.0312
##  $ pos               : num [1:475] 0 0 0 0.541 0 ...
##  $ neg               : num [1:475] 0 -1.59 -5.23 -3.69 -7.67 ...
##  $ head.start        : num 0
##  $ decision.interval: num 5
##  $ se.shift          : num 1
##  $ violations        :List of 2
##  - attr(*, "class")= chr "cusum.qcc"
```

```
qcc::cusum(infl_ts, center = mod2)
```



```
## List of 14
##  $ call             : language qcc::cusum(data = infl_ts, center = mod2)
##  $ type             : chr "cusum"
##  $ data.name        : chr "infl_ts"
##  $ data             : num [1:475, 1] 6.2 6.08 5.65 5.71 5.69 ...
##   ..- attr(*, "dimnames")=List of 2
##  $ statistics       : Named num [1:475] 6.2 6.08 5.65 5.71 5.69 ...
##   ..- attr(*, "names")= chr [1:475] "1" "2" "3" "4" ...
##  $ sizes            : int [1:475] 1 1 1 1 1 1 1 1 1 1 ...
##  $ center           : Time-Series [1:475] from 1982 to 2022: 6.2 6.09 5.95 5.62 5.69 ...
##  $ std.dev          : num 0.0848
##  $ pos              : num [1:475] 0 0 0 0.608 0.159 ...
##  $ neg              : num [1:475] 0 0 -3.049 -1.441 -0.889 ...
##  $ head.start       : num 0
##  $ decision.interval: num 5
##  $ se.shift         : num 1
##  $ violations       :List of 2
##  - attr(*, "class")= chr "cusum.qcc"
```

The CUSUM plot for the gold model has 10 points beyond boundaries out of 475 and returns back to being within the boundaries quickly after each excursion. The CUSUM plot for the inflation model only has 7 points beyond boundaries, and again shows no pattern of straying beyond the boundaries.

```
rmse_g <- sqrt(mean((gold_ts - mod1) ^ 2))
rmse_i <- sqrt(mean((infl_ts - mod2) ^ 2))
print(paste("RMSE Gold: ", rmse_g))
```

```
## [1] "RMSE Gold:  0.0366835129296428"
```

```
print(paste("RMSE Inflation: ", rmse_i))
```

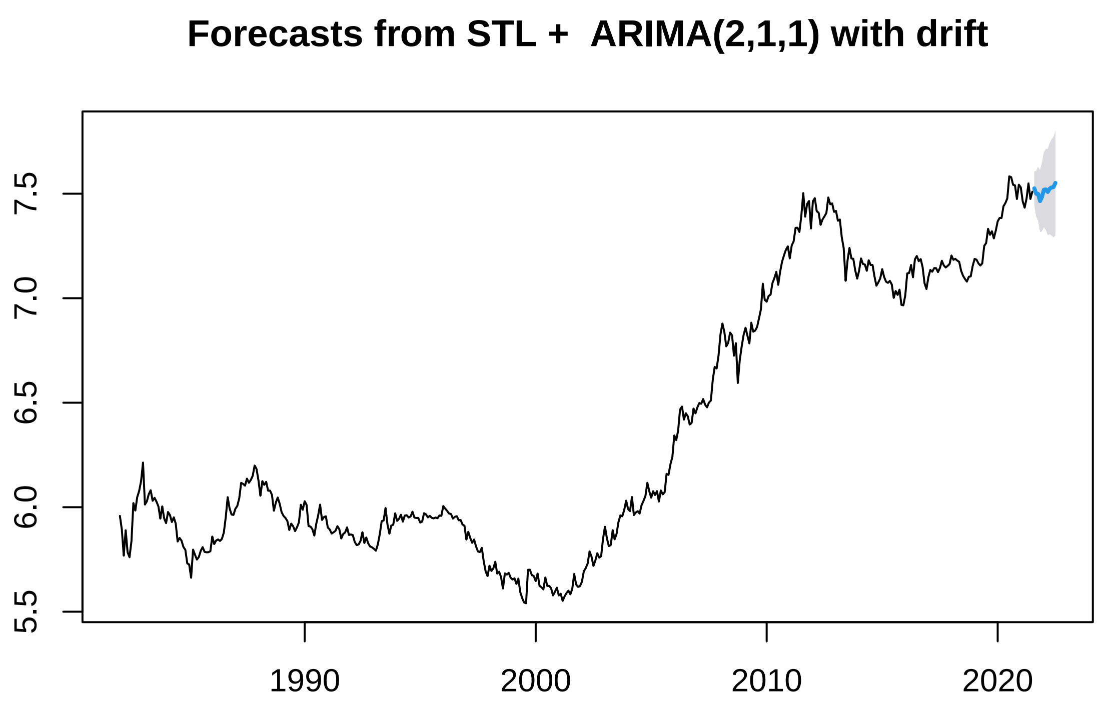
```
## [1] "RMSE Inflation:  0.115940040997104"
```

The RMSE for each model is very small. The RMSE for log(gold) is 0.037, which suggests that on average, the model's prediction is only off by 0.037 of the actual value for log(gold). The RMSE for inflation is similarly small, only 0.116.

```
castg <- stlf(gold_ts, method="arima", level=0.95, h=12)
castg
```

```
##          Point Forecast    Lo 95    Hi 95
## Aug 2021       7.525693 7.444492 7.606894
## Sep 2021       7.499695 7.389947 7.609443
## Oct 2021       7.499080 7.369104 7.629056
## Nov 2021       7.464878 7.316025 7.613731
## Dec 2021       7.484824 7.319711 7.649936
## Jan 2022       7.518830 7.338809 7.698852
## Feb 2022       7.520562 7.326790 7.714334
## Mar 2022       7.508241 7.301633 7.714848
## Apr 2022       7.524399 7.305707 7.743092
## May 2022       7.529740 7.299597 7.759884
## Jun 2022       7.531781 7.290730 7.772832
## Jul 2022       7.551680 7.300194 7.803166
```

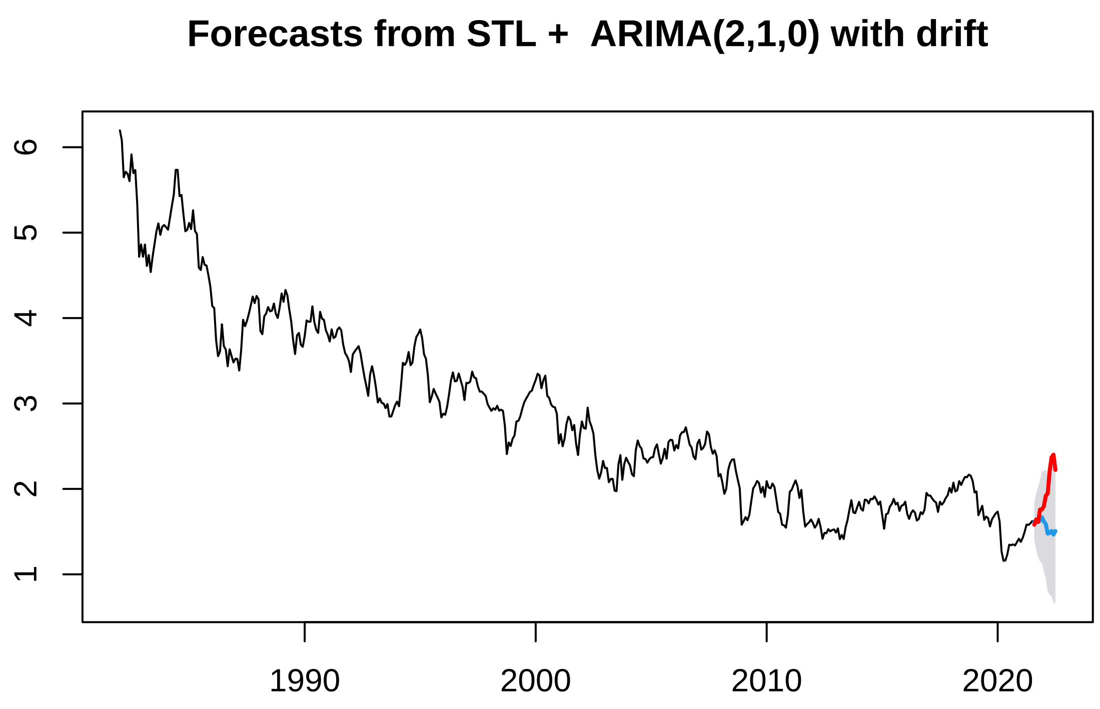
```
plot(castg)
```



```
casti <- stlf(infl_ts, method="arima", level=0.95, h=12)
casti
```

```
##          Point Forecast     Lo 95    Hi 95
## Aug 2021       1.617932 1.3943430 1.841520
## Sep 2021       1.620768 1.2914228 1.950112
## Oct 2021       1.613066 1.2030435 2.023088
## Nov 2021       1.628823 1.1514322 2.106213
## Dec 2021       1.668266 1.1318879 2.204643
## Jan 2022       1.618525 1.0290321 2.208019
## Feb 2022       1.588425 0.9502212 2.226628
## Mar 2022       1.476555 0.7931038 2.160005
## Apr 2022       1.486202 0.7603191 2.212085
## May 2022       1.507498 0.7415292 2.273466
## Jun 2022       1.465590 0.6615319 2.269648
## Jul 2022       1.507027 0.6666041 2.347450
```

```
plot(casti)

#just for fun
truinfl <- ts(infl$EXPINF10YR[476:487], start=2021.5833333, freq=12)
lines(truinfl, col="red", lwd=2.0)
```



```r
#arima
alt_arg <- auto.arima(gold_ts)
summary(alt_arg)
```

```
## Series: gold_ts
## ARIMA(1,1,0)(1,0,0)[12] with drift
##
## Coefficients:
##           ar1    sar1   drift
##       -0.1201  0.0397  0.0033
## s.e.   0.0458  0.0484  0.0020
##
## sigma^2 = 0.002162:  log likelihood = 783.32
## AIC=-1558.64   AICc=-1558.55   BIC=-1541.99
##
## Training set error measures:
##                         ME       RMSE       MAE          MPE      MAPE      MASE
## Training set -2.3022e-05 0.04630156 0.0348816 -0.006503268 0.5480459 0.2911952
##                      ACF1
## Training set -0.008723968
```

```r
forecast::forecast(alt_arg, h=12)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Aug 2021       7.509474 7.449885 7.569064 7.418340 7.600608
## Sep 2021       7.511576 7.432201 7.590950 7.390183 7.632969
## Oct 2021       7.514582 7.418976 7.610188 7.368365 7.660799
## Nov 2021       7.515150 7.405743 7.624556 7.347826 7.682473
## Dec 2021       7.521025 7.399368 7.642683 7.334967 7.707084
## Jan 2022       7.523681 7.390900 7.656463 7.320610 7.726753
## Feb 2022       7.524181 7.381137 7.667224 7.305415 7.742947
## Mar 2022       7.526143 7.373527 7.678760 7.292736 7.759550
## Apr 2022       7.531059 7.369436 7.692683 7.283877 7.778242
## May 2022       7.537081 7.366926 7.707236 7.276851 7.797311
## Jun 2022       7.537277 7.358998 7.715555 7.264623 7.809930
## Jul 2022       7.541820 7.355772 7.727867 7.257284 7.826355
```

```
alt_ari <- auto.arima(infl_ts)
forecast::forecast(alt_ari, h=12)
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Aug 2021       1.601601 1.439575 1.763627 1.3538030 1.849399
## Sep 2021       1.618055 1.386807 1.849304 1.2643910 1.971719
## Oct 2021       1.622499 1.342684 1.902313 1.1945595 2.050438
## Nov 2021       1.601367 1.279791 1.922942 1.1095595 2.093174
## Dec 2021       1.620901 1.261697 1.980104 1.0715467 2.170255
## Jan 2022       1.618543 1.228146 2.008939 1.0214830 2.215602
## Feb 2022       1.602010 1.181788 2.022233 0.9593357 2.244685
## Mar 2022       1.622417 1.174359 2.070475 0.9371714 2.307663
## Apr 2022       1.611190 1.138952 2.083428 0.8889642 2.333416
## May 2022       1.601692 1.105320 2.098063 0.8425563 2.360827
## Jun 2022       1.622348 1.103445 2.141252 0.8287536 2.415943
## Jul 2022       1.607215 1.068098 2.146331 0.7827072 2.431722
```

```
#mape
mape_arg <- mean(abs((gold_ts-alt_arg$fitted)/gold_ts))
print(paste("MAPE ARIMA gold: ", mape_arg))
```

```
## [1] "MAPE ARIMA gold:  0.0054804590207938"
```

```
mape_ari <- mean(abs((infl_ts-alt_ari$fitted)/infl_ts))
print(paste("MAPE ARIMA inflation: ", mape_ari))
```

```
## [1] "MAPE ARIMA inflation:  0.035407367555079"
```

```
mape_mod1 <- mean(abs((gold_ts-mod1)/gold_ts))
print(paste("MAPE Model gold: ", mape_mod1))
```

```
## [1] "MAPE Model gold:  0.00434945093125225"
```

```
mape_mod2 <- mean(abs((infl_ts-mod2)/infl_ts))
print(paste("MAPE Model inflation: ", mape_mod2))
```

```
## [1] "MAPE Model inflation:  0.0329376205216756"
```

The model which includes trend, seasonality, and cyclic components performs better in terms of MAPE than the pure ARIMA model in predicting both log(gold) and inflation. We combine our model with the ARIMA model using a linear regression for both log(gold) and for inflation.

```
comb.reg_g <- lm(gold_ts~mod1+alt_arg$fitted)

mape_combg <- mean(abs((gold_ts-comb.reg_g$fitted.values)/gold_ts))
print(paste("MAPE Combined gold: ", mape_combg))
```

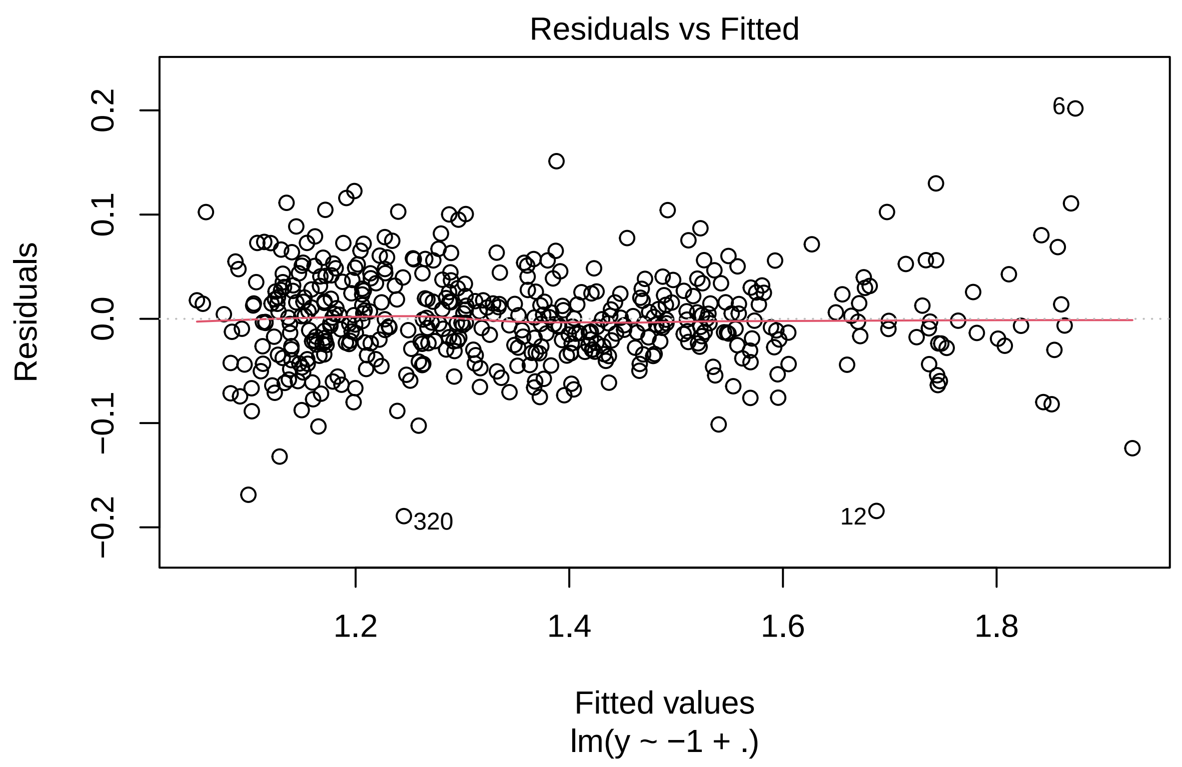
```
## [1] "MAPE Combined gold:  0.00433531336346721"
```

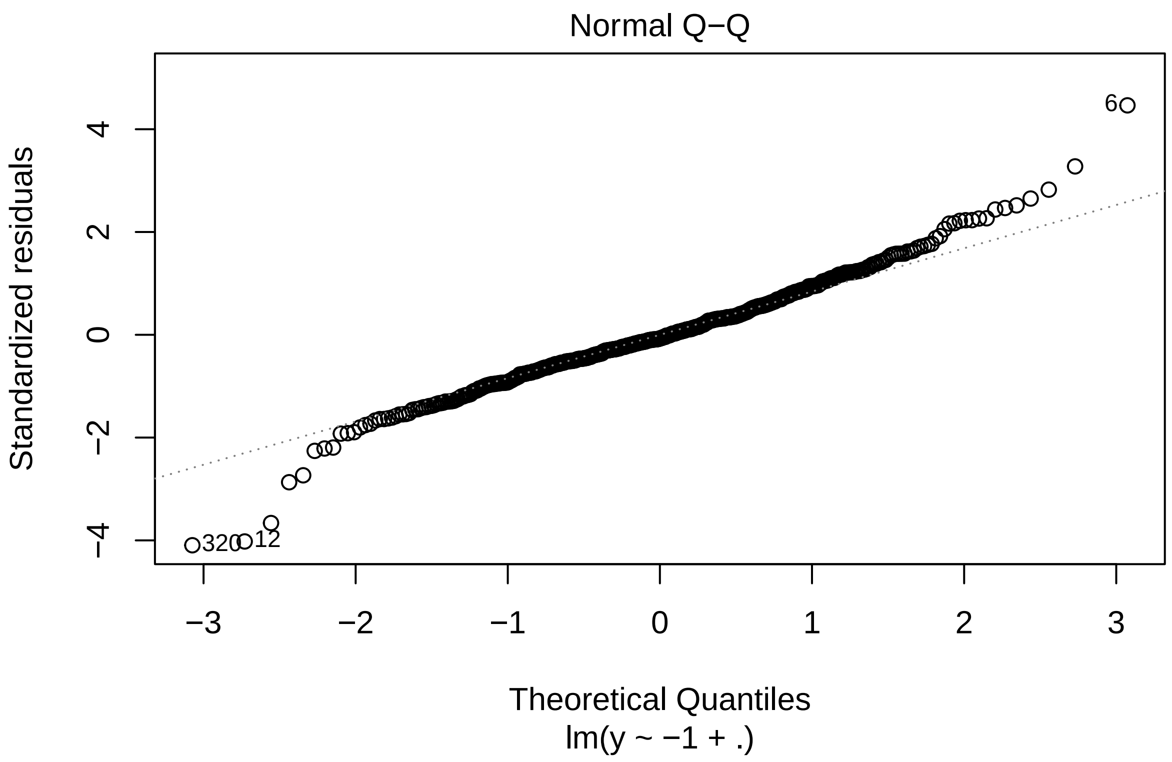
```
comb.reg_i <- lm(infl_ts~mod2+alt_ari$fitted)

mape_combi <- mean(abs((infl_ts-comb.reg_i$fitted.values)/infl_ts))
print(paste("MAPE Combined gold: ", mape_combi))
```

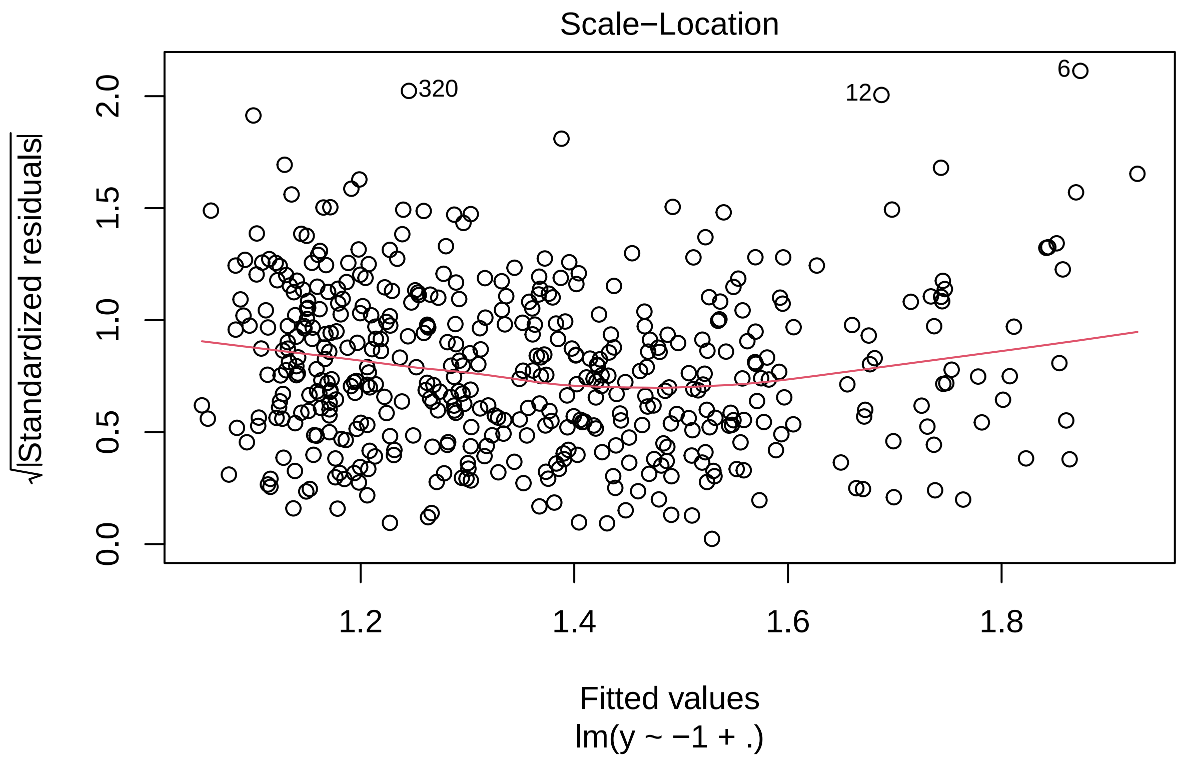
```
## [1] "MAPE Combined gold:  0.0327710519465302"
```

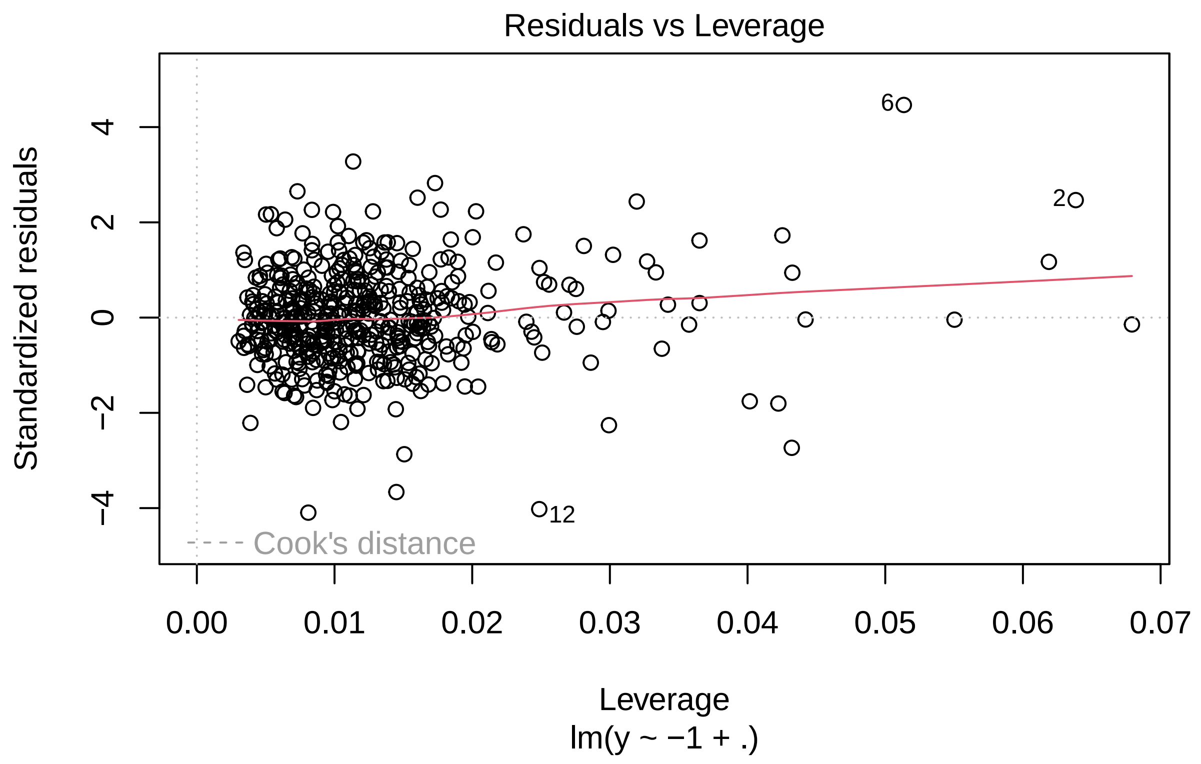
The MAPE from the forecast generated by combining our model and the ARIMA model is lower than the MAPE for either individual model when predicting both log(gold) and inflation.

```
x <- data.frame(gold_ts,infl_ts)
var_est <- VAR(x, p=2, type="both")
plot(var_est$varresult$gold_ts)
```

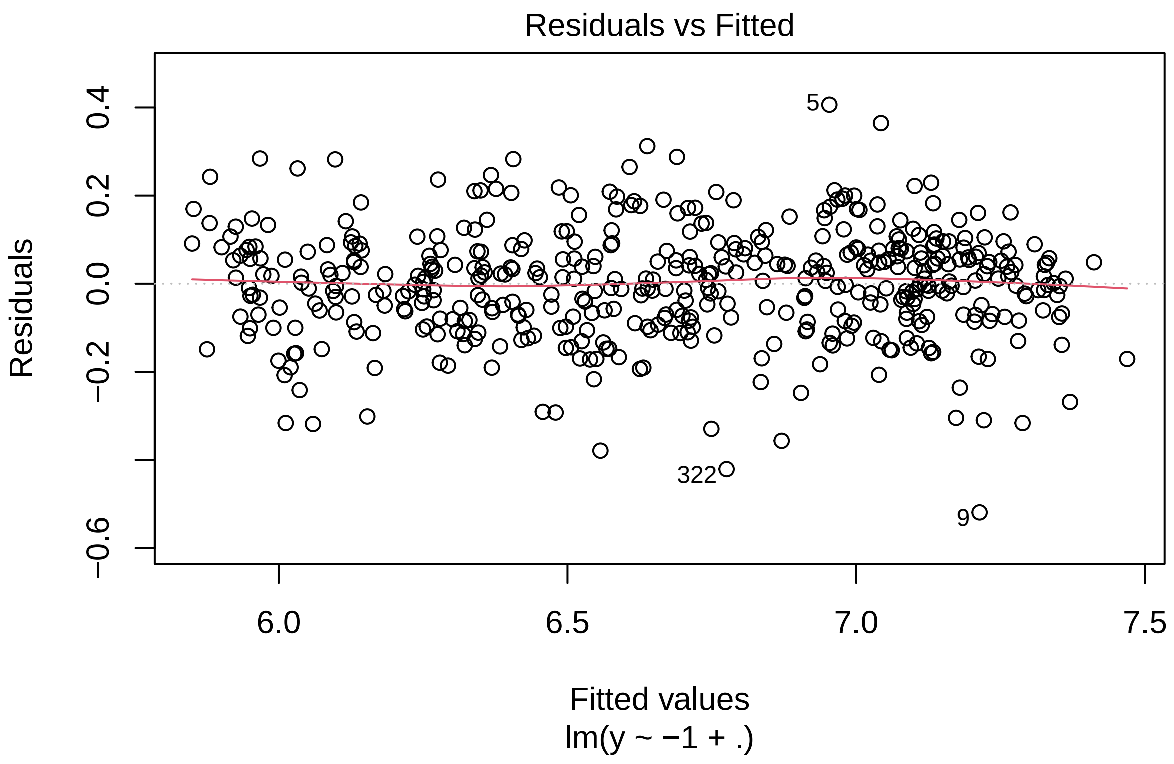



Normal Q−Q
Standardized residuals
4
2
0
−2
−4
6
320
12
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(y ~ −1 + .)

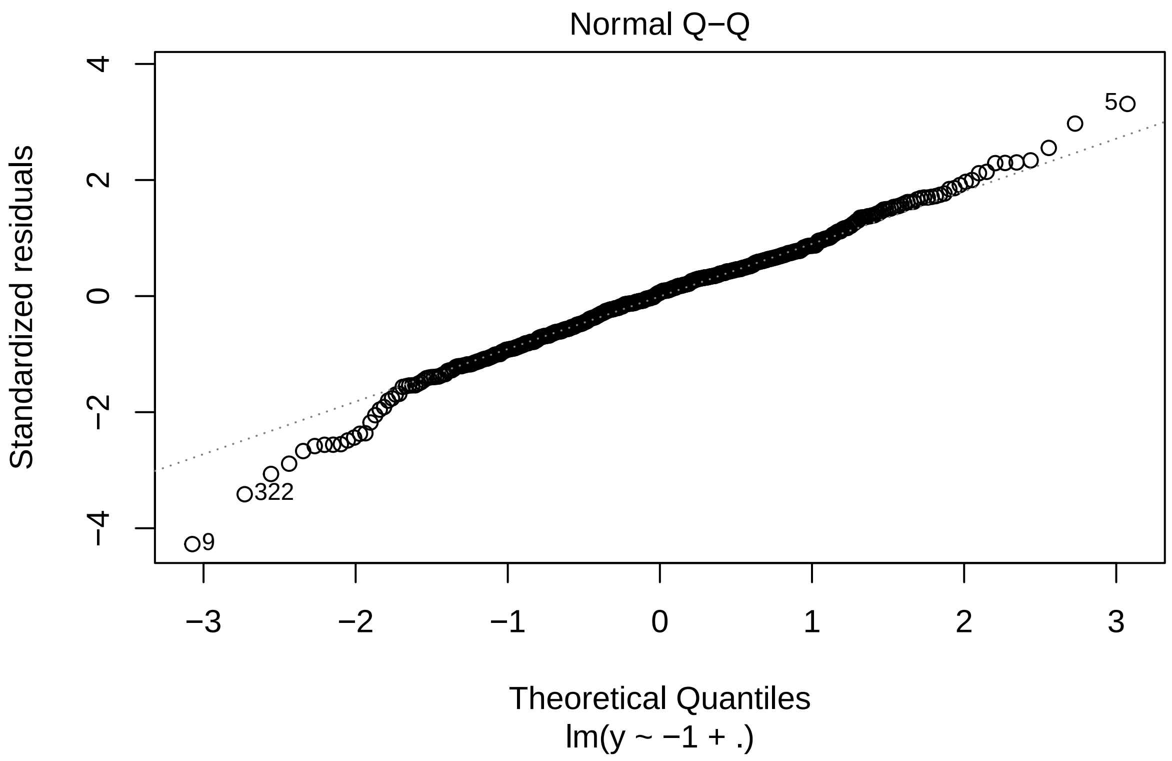

Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
1.2
1.4
1.6
1.8
Fitted values
lm(y ~ −1 + .)
320
12
6

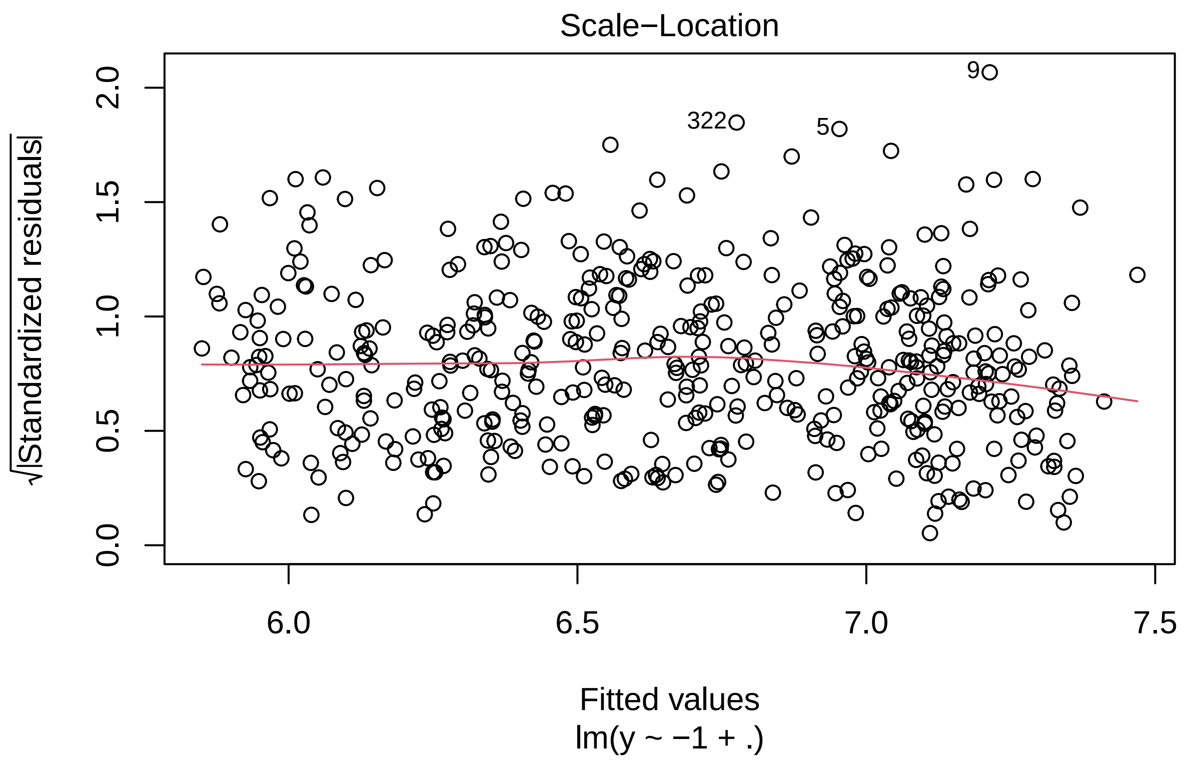


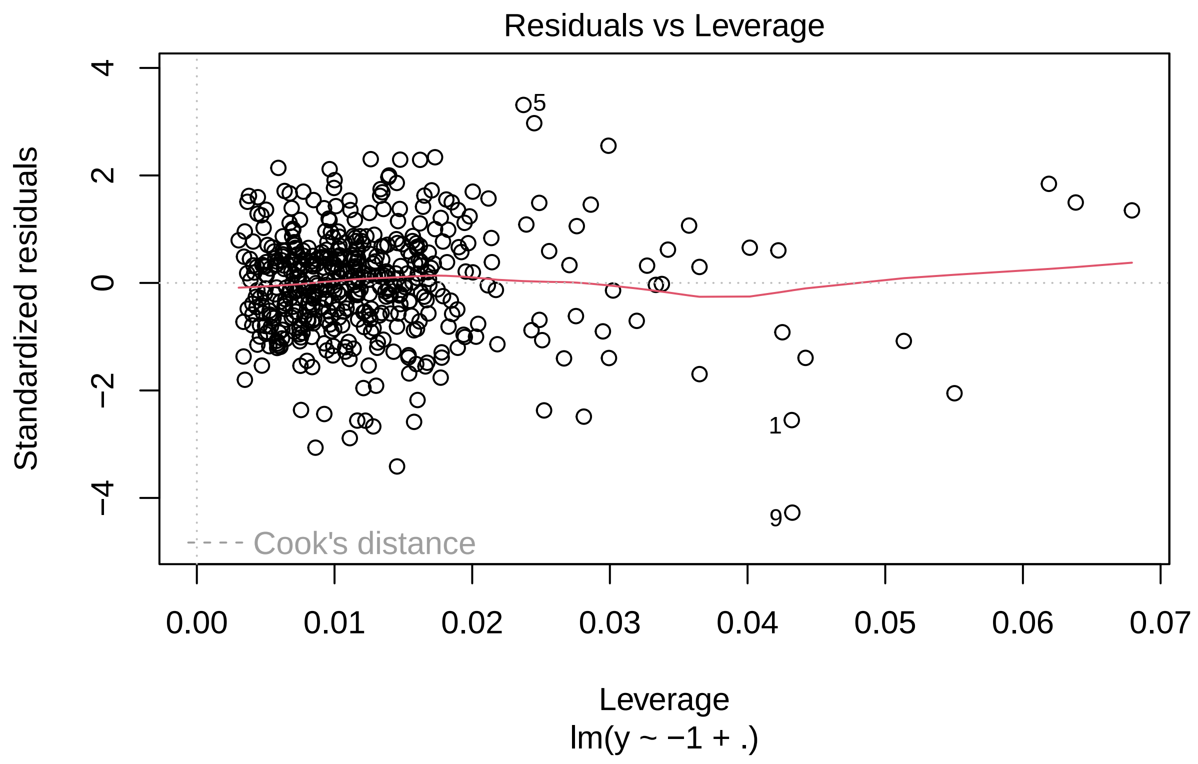
```
plot(var_est$varresult$infl_ts)
```



Residuals vs Fitted
Residuals
Fitted values
lm(y ~ −1 + .)
0.4
0.2
0.0
−0.2
−0.6
6.0
6.5
7.0
7.5
5
322
9

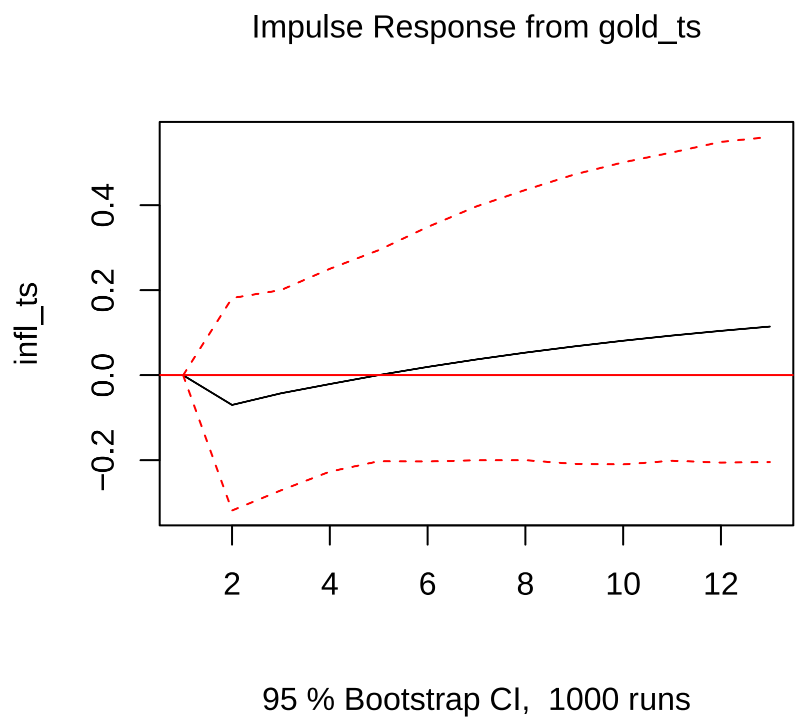

Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(y ~ −1 + .)
4
2
0
−2
−4
−3
−2
−1
0
1
2
3
5
322
9


Scale−Location
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
6.0
6.5
7.0
7.5
Fitted values
lm(y ~ −1 + .)
9
322
5

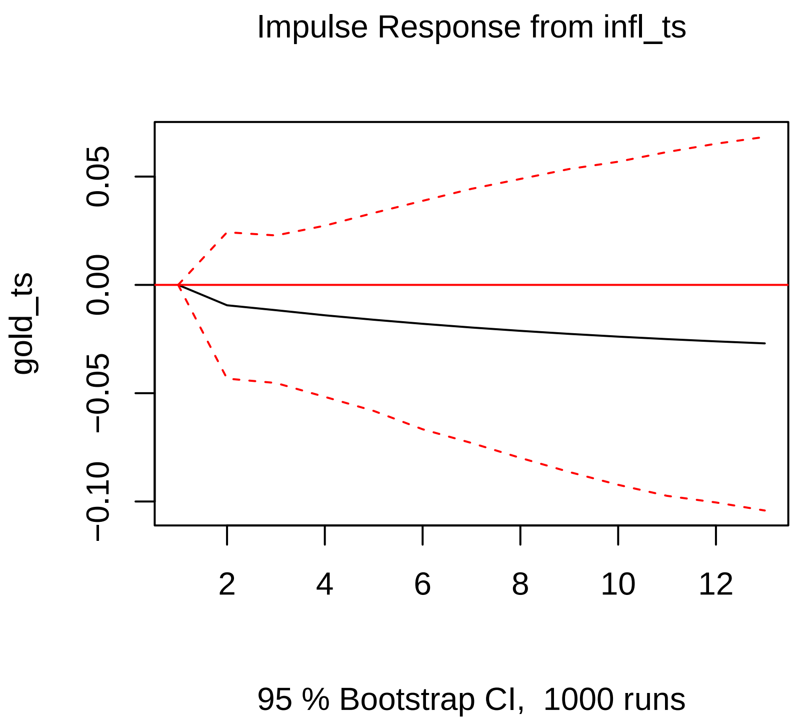


Make sure to show the relevant plots and discuss your results from the fit

```r
#plot irfs
plf <- irf(var_est, impulse = "gold_ts", response = "infl_ts",
             n.ahead = 12, ortho = FALSE, runs = 1000)
plot(plf)
```



```
pls <- irf(var_est, impulse = "infl_ts", response = "gold_ts",
             n.ahead = 12, ortho = FALSE, runs = 1000)
plot(pls)
```



The impact of an unanticipated one-unit upward shift in the log of gold price is visible in the first IRF plot, which shows a small negative shock before an upward swing. A similar shock in the inflation rate causes a slight negative shock in the log of gold price, which maintains its level.

```
#granger test
grangertest(gold_ts~infl_ts, order=1, data=x)
```

```
## Granger causality test
##
## Model 1: gold_ts ~ Lags(gold_ts, 1:1) + Lags(infl_ts, 1:1)
## Model 2: gold_ts ~ Lags(gold_ts, 1:1)
##   Res.Df Df      F  Pr(>F)
## 1    471
## 2    472 -1 4.5538 0.03336 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
grangertest(infl_ts~gold_ts, order=1, data=x)
```

```
## Granger causality test
##
## Model 1: infl_ts ~ Lags(infl_ts, 1:1) + Lags(gold_ts, 1:1)
## Model 2: infl_ts ~ Lags(infl_ts, 1:1)
##   Res.Df Df      F Pr(>F)
## 1    471
## 2    472 -1 2.0138 0.1565
```

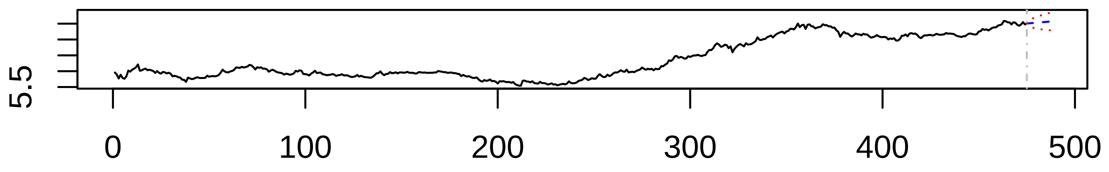
At the *alpha = 0.05* significance level, we reject the null hypothesis for the first Granger test, so inflation influences the log of gold price, but we fail to reject the null for the second Granger test, so the log of gold price doesn't influence inflation.

```
cast_f <- predict(var_est, n.ahead=12, ci = 0.95)
print(cast_f)
```

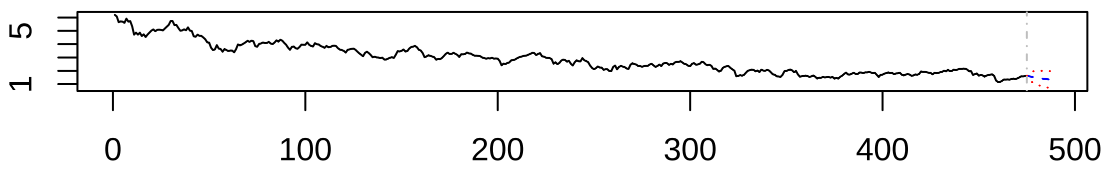
```
## $gold_ts
##           fcst    lower    upper         CI
##  [1,] 7.510452 7.419497 7.601407 0.09095467
##  [2,] 7.515702 7.394894 7.636511 0.12080848
##  [3,] 7.520493 7.375377 7.665608 0.14511591
##  [4,] 7.525415 7.359836 7.690993 0.16557864
##  [5,] 7.530388 7.346818 7.713958 0.18357008
##  [6,] 7.535418 7.335668 7.735168 0.19975009
##  [7,] 7.540498 7.325968 7.755028 0.21453035
##  [8,] 7.545625 7.317442 7.773808 0.22818331
##  [9,] 7.550796 7.309895 7.791697 0.24090116
## [10,] 7.556007 7.303181 7.808832 0.25282555
## [11,] 7.561255 7.297191 7.825320 0.26406455
## [12,] 7.566538 7.291835 7.841241 0.27470287
##
## $infl_ts
##           fcst     lower    upper        CI
##  [1,] 1.595121 1.3517336 1.838508 0.2433873
##  [2,] 1.565051 1.2180807 1.912021 0.3469700
##  [3,] 1.536246 1.1200656 1.952426 0.4161803
##  [4,] 1.509198 1.0422732 1.976122 0.4669246
##  [5,] 1.483772 0.9778416 1.989703 0.5059305
##  [6,] 1.459849 0.9231084 1.996589 0.5367403
##  [7,] 1.437310 0.8758091 1.998811 0.5615009
##  [8,] 1.416048 0.8344110 1.997684 0.5816367
##  [9,] 1.395962 0.7978103 1.994113 0.5981514
## [10,] 1.376960 0.7651771 1.988743 0.6117832
## [11,] 1.358958 0.7358675 1.982049 0.6230909
## [12,] 1.341878 0.7093697 1.974386 0.6325081
```

```
plot(cast_f)
```

## Forecast of series gold_ts



## Forecast of series infl_ts



```
mape_varg <- mean(abs((gold_ts-var_est$varresult$gold_ts$fitted.values)/gold_ts))
```

```
## Warning in ‘-.default‘(gold_ts, var_est$varresult$gold_ts$fitted.values):
## longer object length is not a multiple of shorter object length
```

```
print(paste("MAPE VAR gold: ", mape_varg))
```

```
## [1] "MAPE VAR gold:  0.00589490274708813"
```

```
mape_vari <- mean(abs((infl_ts-var_est$varresult$infl_ts$fitted.values)/infl_ts))
```

```
## Warning in ‘-.default‘(infl_ts, var_est$varresult$infl_ts$fitted.values):
## longer object length is not a multiple of shorter object length
```

```
print(paste("MAPE VAR inflation: ", mape_vari))
```

```
## [1] "MAPE VAR inflation:  0.0487975993614942"
```

The VAR forecast has a higher MAPE than the forecasts obtained from our model, from the ARIMA model, and from the model which combined both ours and the ARIMA model, so in terms of MAPE it performs worse.

## III. Conclusions and future work

We created a model by performing STL decompositions on two time series, one being the log of gold price, the other being the 10-Year Expected Inflation. We then used the trend and seasonal components generated by these decompositions along with an ARIMA model to estimate the cyclic component for each time series to generate a model which outperformed both a pure ARIMA model and a VAR model in forecasting both time series in terms of MAPE. After combining our model with the ARIMA model through a linear regression, we created a forecast which outperformed our initial model's forecast. For future work, combining our model with other forecasts, such as those generated by exponential smoothing or Holt-Winters, could create an even more accurate forecast. Also, finding data with greater frequency (i.e. daily) could lead to more accuracy in forecasts.

## IV. References

FRED Economic Data, St Louis Fed. (2023). *10-Year Expected Inflation* [Data set]. https://fred.stlouisfed.org/series/EXPINF10YR

World Gold Council. (2021). *Monthly Gold Prices (1979-2021)* [Data set]. https://www.kaggle.com/datasets/odins0n/monthly-gold-prices