Macroeconometrics, Empirical project

Samuel, * Hochholzer Matthias†

xxth June 2021

Contents

*student ID 00000000
†student ID 11724853

1 Idea

We want to look at the relationship between certain prices and the respective search interest on Google for these prices. Can we find granger causality for this relationship? What are possible issues? For example: modern trading algorithms scrape data from the internet and then buy or sell based on the sentiment. Large spikes in search interest may trigger such algorithms. As media spreads the news of price increases more people will look up prices of goods and commodities, again triggering the algorithms. This is basically a feedback loop.

2 Data

First some notes on the data. The data on the search index of certain prices is taken from Google trends which collects the search queries of people within a specific region (here: United States of America). This data is aggregated on a monthly basis and normalized with a range from zero to 100. Already filtered out are duplicate searches in the sense that the same user made the same search multiple times within a short time-frame. This way we exclude the users which have already invested and constantly checked the prices to look how their investment is doing. Data points are divided by total searches for the month and region to represent the relative popularity, i.e. no over-weighting of regions with more people than others which would, given the same search behavior, lead to differing popularities otherwise.

The data on gold prices comes form the London Bullion Market Association Gold Price and the Federal Reserve Bank of St. Louis. It is measured in USD per troy, daily at 3:00pm. Aggregation is done via prices at the end of each month and it is not seasonally adjusted.

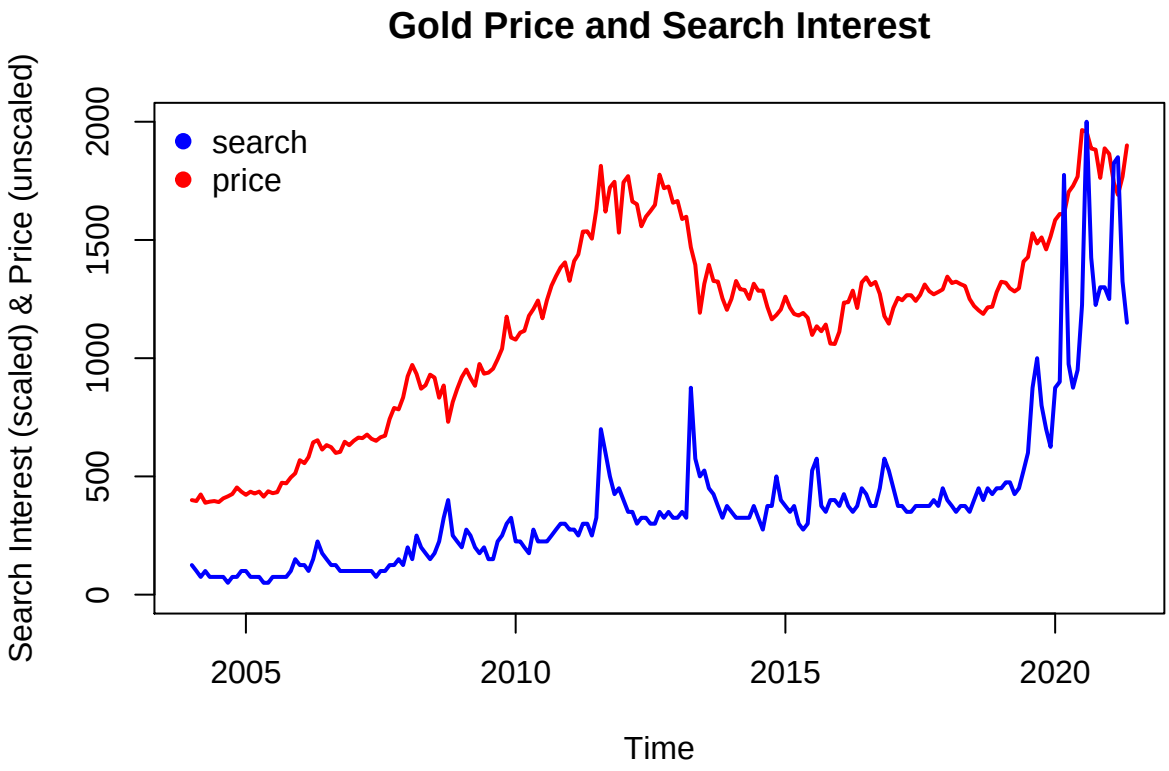
In parts of this project we scale the price and search index to a range from zero to one in order to compare the relative movements more easily.

3 Project Code

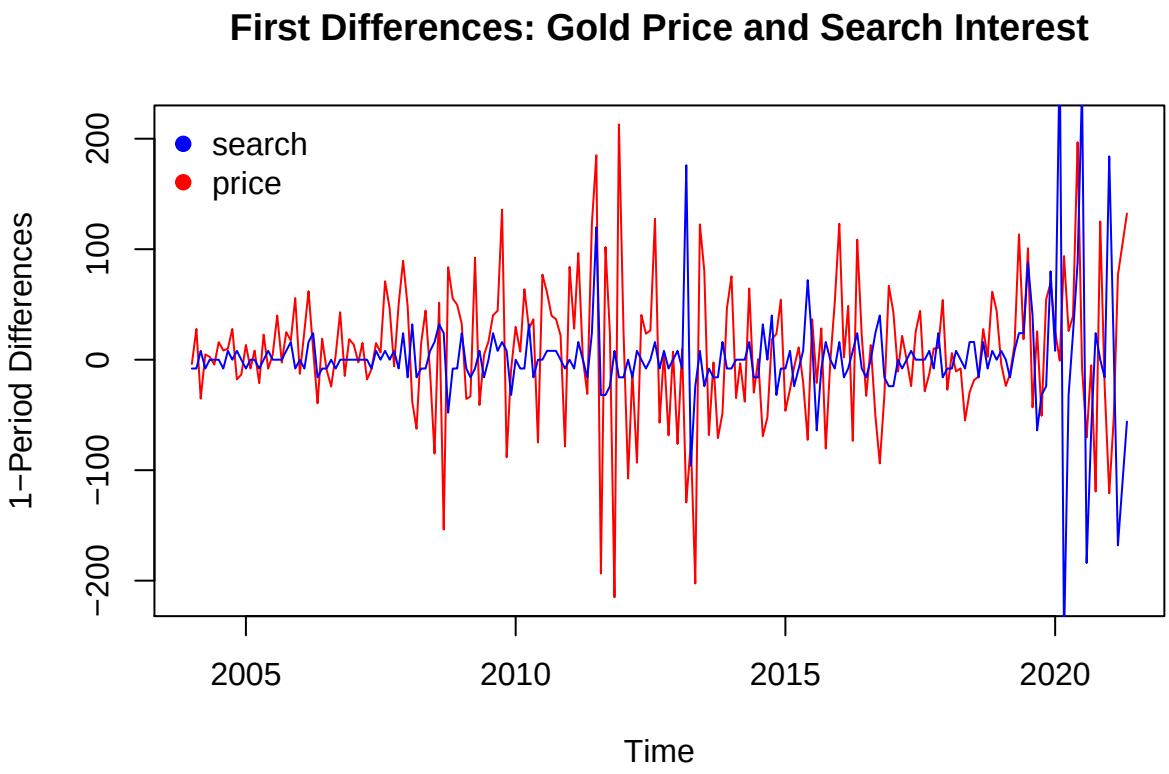
```
# clear workspace
rm(list=ls())

# load needed libraries
library(readr)
library(vars)
library(tseries)

# set working directory
#setwd("/Users/samue/Downloads/Studium/Economics (Master - Vienna)/2. Semester/Macroeconometrics/Projec

# import search trends
data <-  read_csv("btc-vs-gold-2004.csv", col_types = cols(Month = col_date(format = "%Y-%m")))
# import prices data:
gold_pr <- read_csv("gold-2004.csv", col_types = cols(DATE = col_date(format = "%Y-%m-%d")))
# import high-frequency prices for gold:
gold_HF <- read_csv('gold-2001-HF.csv', col_types = cols(DATE = col_date(format = '%Y-%m-%d'),GOLDPMGBD
```

```
# plot gold price on monthly basis
plot(y=gold_pr$GOLDPMGBD228NLBM,x=gold_pr$DATE,type = 'l', lwd = 2, col = 'red',
     ylim = c(0,2000), main = 'Gold Price and Search Interest',
     xlab = 'Time', ylab = 'Search Interest (scaled) & Price (unscaled)')
# add gold search interest scaled up
lines(y=25*data$GOLD,x=gold_pr$DATE, lwd = 2, col = 'blue')
```

```
legend('topleft', legend = c('search','price'),
       col = c('blue','red'), bty = "n", pch = c(19,19))
```

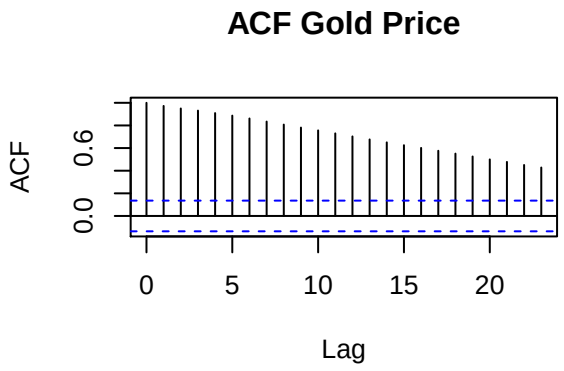


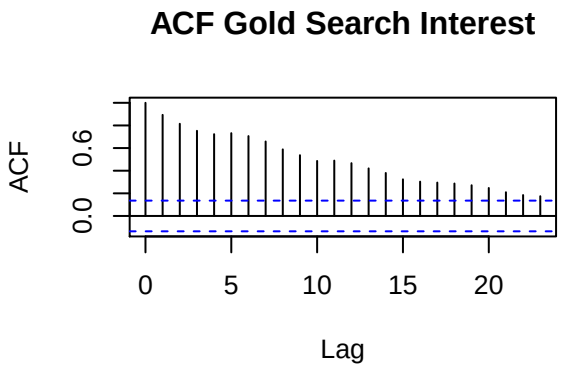

```
# create first differenced prices and search interest
t <- length(gold_pr$DATE)
gold_price_FD <- rep(0,t-1)
for(i in 2:209){gold_price_FD[i-1] <- gold_pr$GOLDPMGBD228NLBM[i]-gold_pr$GOLDPMGBD228NLBM[i-1]}
gold_search_FD <- rep(0,t-1)
for(i in 2:209){gold_search_FD[i-1] <- data$GOLD[i]-data$GOLD[i-1]}
t_1 <- length(gold_HF$DATE)
gold_daily_FD <- rep(0,t_1-1)
for(i in 2:5332){gold_daily_FD[i-1] <- gold_HF$GOLDPMGBD228NLBM[i]-gold_HF$GOLDPMGBD228NLBM[i-1]}

# plot first differenced variables
plot(y=gold_price_FD,x=gold_pr$DATE[1-209], type = 'l', lwd = 1, col = 'red',
     xlab = 'Time', ylab = '1-Period Differences',
     main = 'First Differences: Gold Price and Search Interest')
lines(y=gold_search_FD*8,x=gold_pr$DATE[1-209], lwd = 1, col = 'blue')
legend('topleft', legend = c('search','price'),
       col = c('blue','red'), bty = "n", pch = c(19,19))
```

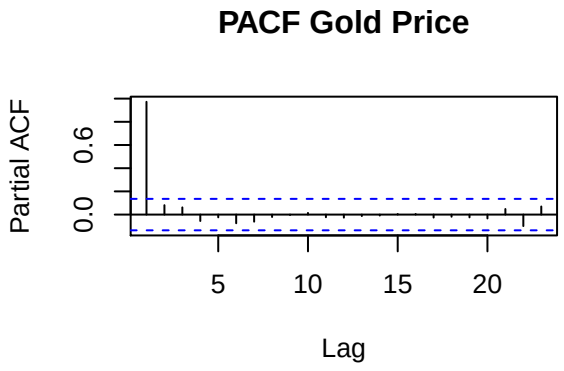


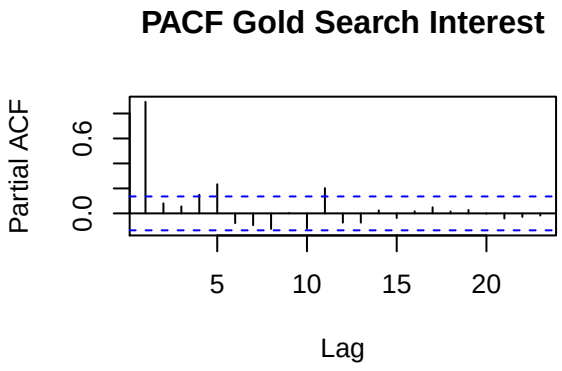
Visually, it appears that the more volatile periods match. An issue seems to be the scaling of the variables.

```
# plot ACF for unmodified variables:
par(mfrow=c(2,2))      # changes the plot layout to more easily compare them
acf(gold_pr$GOLDPMGBD228NLBM, main = 'ACF Gold Price')
acf(data$GOLD, main = 'ACF Gold Search Interest')

# plot PACF for unmodified variables:
pacf(gold_pr$GOLDPMGBD228NLBM, main = 'PACF Gold Price')
pacf(data$GOLD, main = 'PACF Gold Search Interest')
```

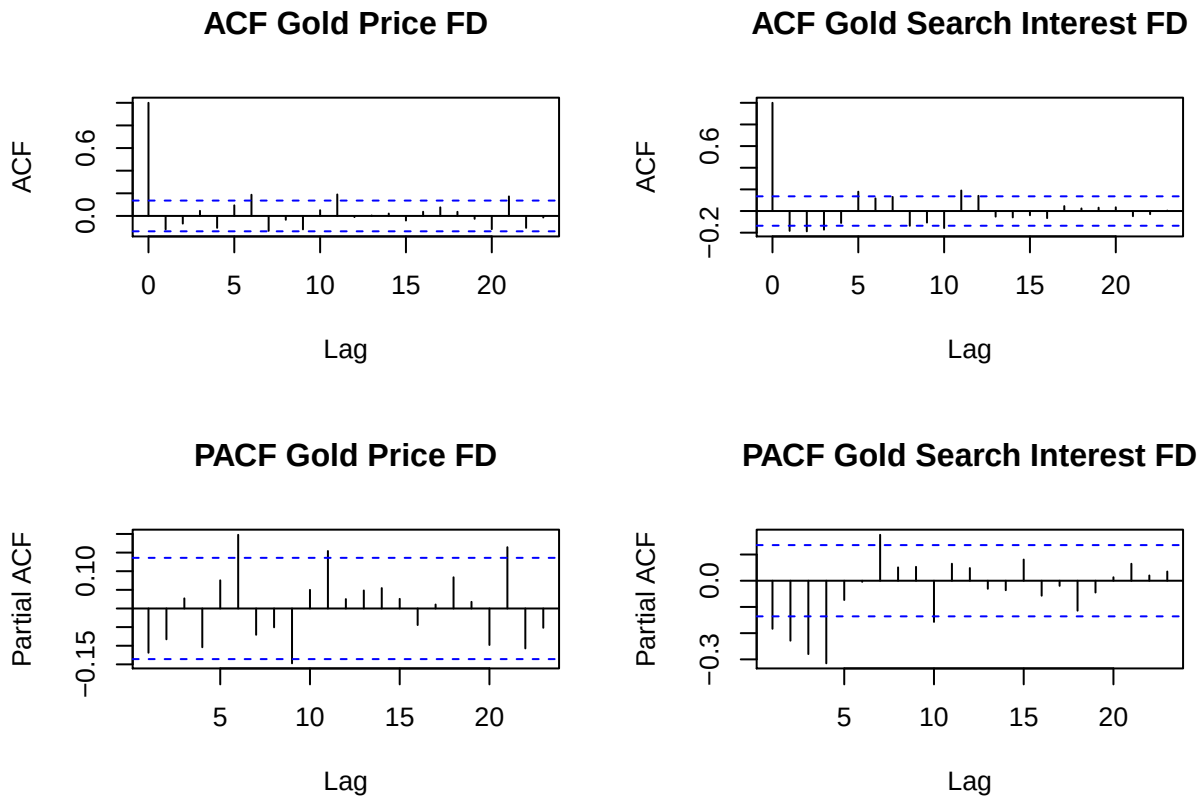









```r
# plot ACF for differenced variables
acf(gold_price_FD,main = 'ACF Gold Price FD')
acf(gold_search_FD, main = 'ACF Gold Search Interest FD')

# plot PACF for differenced variables
pacf(gold_price_FD,main = 'PACF Gold Price FD')
pacf(gold_search_FD, main = 'PACF Gold Search Interest FD')
```

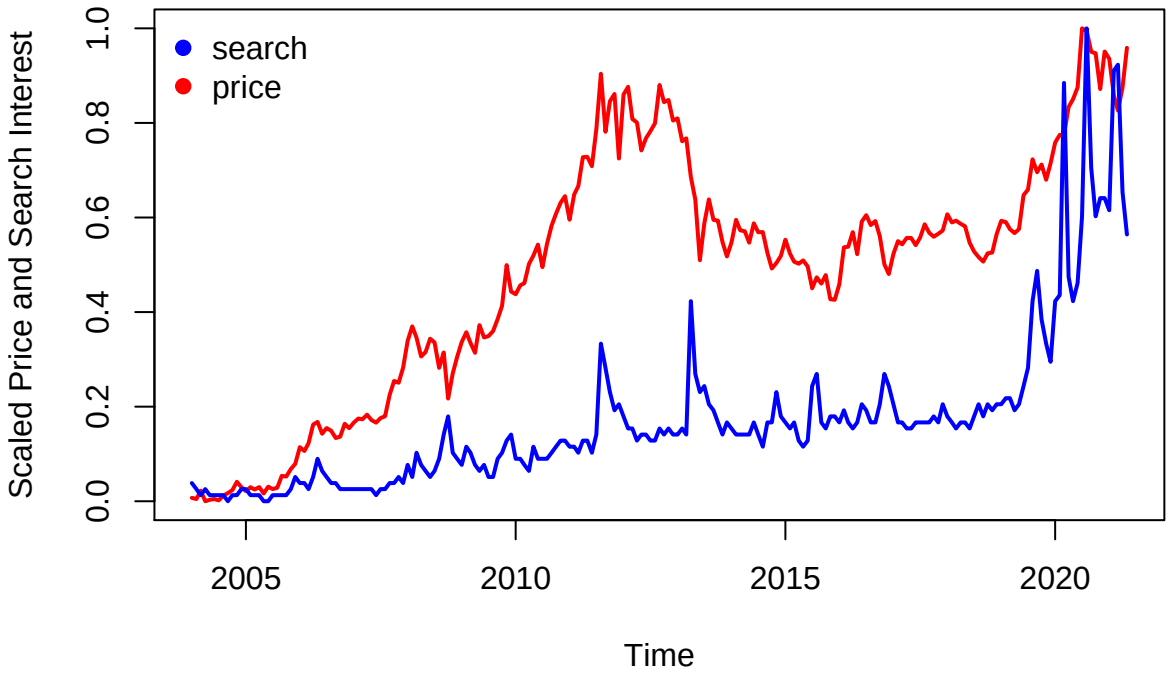



```
par(mfrow = c(1,1))    # revert layout changes
```

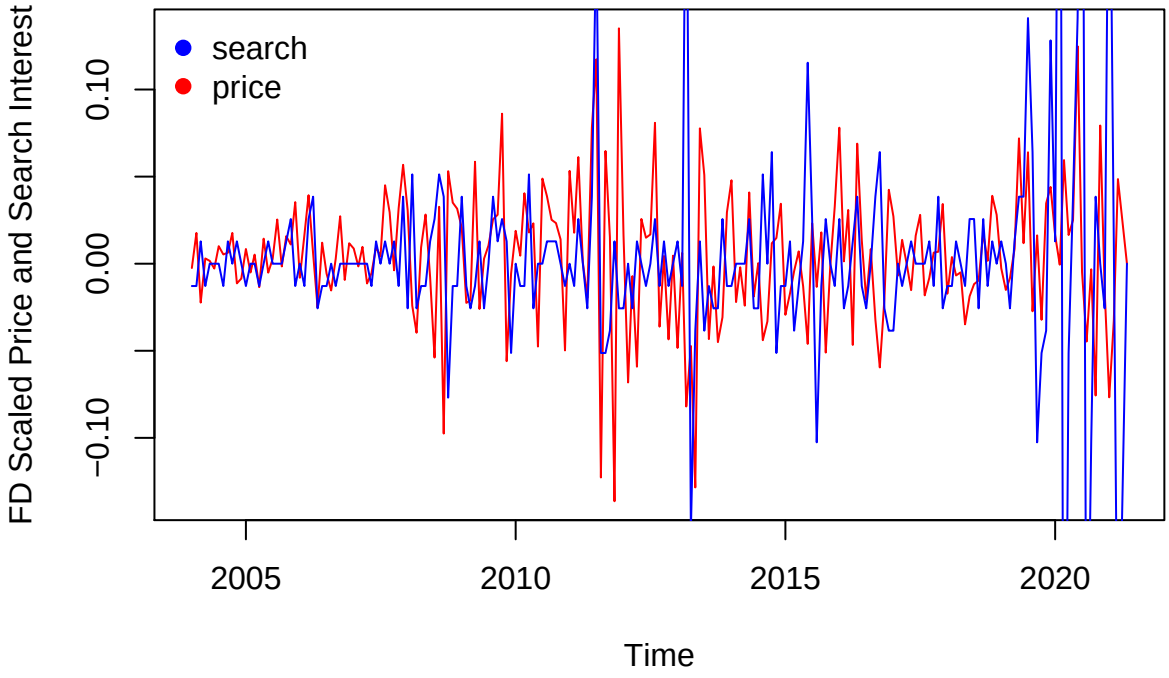
Autocorrelation for the differenced variables seems like no month-on-month relationship between the changes. Kind of like a random walk?

Might help with the interpretation: scale all variables $\mathbf{X}$ such that $X_t \in [0,1] \forall t \in T$.

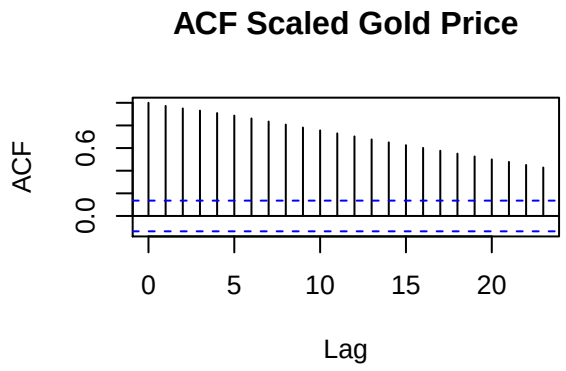
```
range01 <- function(x){(x-min(x))/(max(x)-min(x))}
plot(y=range01(gold_pr$GOLDPMGBD228NLBM),x=gold_pr$DATE, lwd = 2, type = 'l',
     ylab = 'Scaled Price and Search Interest',
     xlab = 'Time', col = 'red')
lines(y=range01(data$GOLD),x=gold_pr$DATE, lwd = 2, col = 'blue')
legend('topleft', legend = c('search','price'),
       col = c('blue','red'), bty = "n", pch = c(19,19))
```

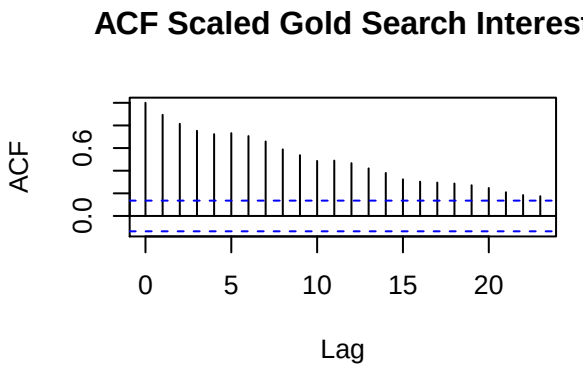


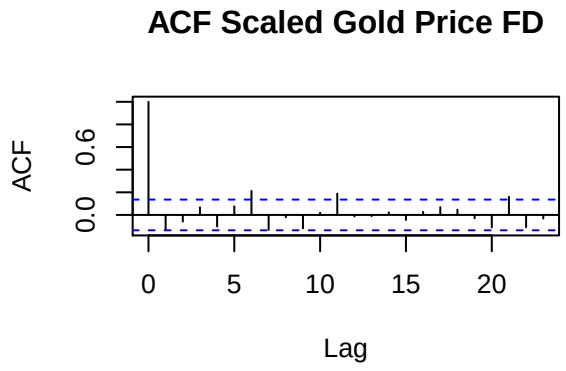

```r
# save scaled variables
gold_price_scaled <- range01(gold_pr$GOLDPMGBD228NLBM)
gold_search_scaled <- range01(data$GOLD)

# create first difference on scaled variables:
gold_search_scaled_FD <- rep(0,t-1)
gold_price_scaled_FD <- rep(0,t-1)

for(i in 2:t-1){
  gold_price_scaled_FD[i-1] <- gold_price_scaled[i]-gold_price_scaled[i-1]
}
for(i in 2:t-1){
  gold_search_scaled_FD[i-1] <- gold_search_scaled[i]-gold_search_scaled[i-1]
}

# plot first differenced:
plot(y=gold_price_scaled_FD, x=gold_pr$DATE[1-209], lwd = 1, type = 'l',
     ylab = 'FD Scaled Price and Search Interest',
     xlab = 'Time', col = 'red')
lines(y= gold_search_scaled_FD, x=gold_pr$DATE[1-209], lwd = 1, col = 'blue')
legend('topleft', legend = c('search','price'),
       col = c('blue','red'), bty = "n", pch = c(19,19))
```

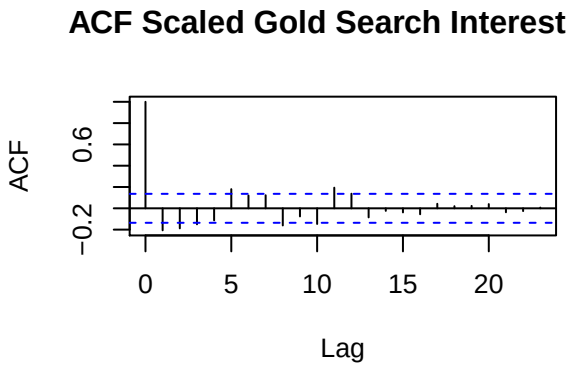



```r
# plot ACFs
par(mfrow=c(2,2))      # changes the plot layout to more easily compare them
acf(gold_price_scaled, main = 'ACF Scaled Gold Price')
acf(gold_search_scaled, main = 'ACF Scaled Gold Search Interest')
acf(gold_price_scaled_FD,main = 'ACF Scaled Gold Price FD')
acf(gold_search_scaled_FD, main = 'ACF Scaled Gold Search Interest FD')
```

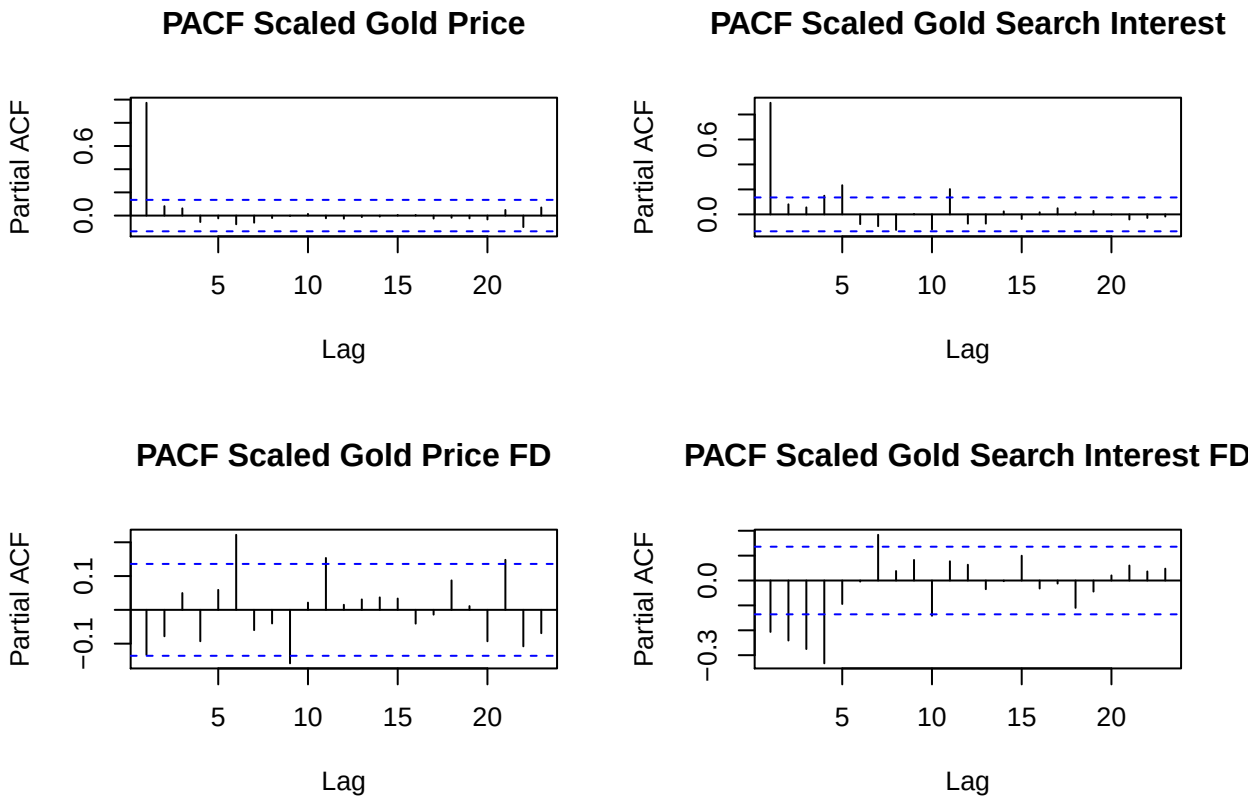









```r
par(mfrow = c(1,1))    # revert layout changes


# plot PACFs
par(mfrow=c(2,2))      # changes the plot layout to more easily compare them
pacf(gold_price_scaled, main = 'PACF Scaled Gold Price')
pacf(gold_search_scaled, main = 'PACF Scaled Gold Search Interest')
pacf(gold_price_scaled_FD,main = 'PACF Scaled Gold Price FD')
pacf(gold_search_scaled_FD, main = 'PACF Scaled Gold Search Interest FD')
```




```
par(mfrow = c(1,1))    # revert layout changes
```

Unsurprisingly the rescaling does not matter for the autocorrelation as it is a scaled measure of linear relationships anyway.

ACF Scaled Gold Search Interest FD together with PACF Scaled Gold Search Interest FD gives evidence for an AR(4).

For the Gold Price it's as you say. Could be a MA(1), AR(1) or an ARMA.

Should we also do DF for gold_search?

```
#######################################################
####### From here on: data saved as time series #####
#######################################################

# save variable vectors as time series format:
gold_price_scaled <- ts(gold_price_scaled, frequency = 12,
                        start = c(2004, 1), end = c(2021, 5))
gold_search_scaled <- ts(gold_search_scaled, frequency = 12,
                         start = c(2004,1), end = c(2021,5))

# set up data for estimation using `VAR()`
VAR_data <- window(ts.union(gold_price_scaled, gold_search_scaled),
                   start = c(2004, 1), end = c(2021, 5))

# estimate model coefficients using `VAR()`
VAR_est <- VAR(y = VAR_data, p = 5)
```

```r
summary(VAR_est)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: gold_price_scaled, gold_search_scaled
## Deterministic variables: const
## Sample size: 204
## Log Likelihood: 655.565
## Roots of the characteristic polynomial:
## 1.019 0.9733 0.8642 0.8642 0.7279 0.7279 0.5834 0.5834 0.4733 0.4733
## Call:
## VAR(y = VAR_data, p = 5)
##
##
## Estimation results for equation gold_price_scaled:
## ==================================================
## gold_price_scaled = gold_price_scaled.l1 + gold_search_scaled.l1 + gold_price_scaled.l2 + gold_searcl
##
##                          Estimate Std. Error t value Pr(>|t|)
## gold_price_scaled.l1   0.8680354  0.0727396  11.933   <2e-16 ***
## gold_search_scaled.l1 -0.0308545  0.0425133  -0.726   0.4689
## gold_price_scaled.l2   0.0525540  0.0958267   0.548   0.5840
## gold_search_scaled.l2 -0.0002992  0.0492484  -0.006   0.9952
## gold_price_scaled.l3   0.0990654  0.0952912   1.040   0.2998
## gold_search_scaled.l3  0.0072362  0.0503455   0.144   0.8859
## gold_price_scaled.l4  -0.1320176  0.0958383  -1.378   0.1700
## gold_search_scaled.l4  0.1023473  0.0512323   1.998   0.0472 *
## gold_price_scaled.l5   0.0837110  0.0742230   1.128   0.2608
## gold_search_scaled.l5 -0.0335239  0.0469061  -0.715   0.4757
## const                  0.0125843  0.0061555   2.044   0.0423 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.03944 on 193 degrees of freedom
## Multiple R-Squared: 0.977,   Adjusted R-squared: 0.9758
## F-statistic: 819.4 on 10 and 193 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation gold_search_scaled:
## ==================================================
## gold_search_scaled = gold_price_scaled.l1 + gold_search_scaled.l1 + gold_price_scaled.l2 + gold_searc
##
##                         Estimate Std. Error t value Pr(>|t|)
## gold_price_scaled.l1   0.229715   0.116612   1.970   0.0503 .
## gold_search_scaled.l1  0.551036   0.068155   8.085 6.70e-14 ***
## gold_price_scaled.l2   0.008144   0.153624   0.053   0.9578
## gold_search_scaled.l2 -0.021149   0.078952  -0.268   0.7891
## gold_price_scaled.l3  -0.031885   0.152766  -0.209   0.8349
## gold_search_scaled.l3  0.010771   0.080711   0.133   0.8940
## gold_price_scaled.l4  -0.136519   0.153643  -0.889   0.3754
## gold_search_scaled.l4  0.105523   0.082133   1.285   0.2004
## gold_price_scaled.l5  -0.044632   0.118990  -0.375   0.7080
```

```
## gold_search_scaled.l5  0.375179   0.075198   4.989 1.35e-06 ***
## const                 -0.009031   0.009868  -0.915   0.3612
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 0.06323 on 193 degrees of freedom
## Multiple R-Squared: 0.8718,  Adjusted R-squared: 0.8651
## F-statistic: 131.2 on 10 and 193 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##                     gold_price_scaled gold_search_scaled
## gold_price_scaled          0.0015554         -0.0001557
## gold_search_scaled        -0.0001557          0.0039975
##
## Correlation matrix of residuals:
##                     gold_price_scaled gold_search_scaled
## gold_price_scaled             1.00000           -0.06244
## gold_search_scaled           -0.06244            1.00000
```

```
# augmented df test on only the gold price
df_test_gold_price <- urca::ur.df(gold_price_scaled, type = c('trend'),
                      selectlags = 'BIC')
summary(df_test_gold_price)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.134978 -0.021219 -0.001006  0.022695  0.130576
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.833e-03  5.949e-03   1.485    0.139
## z.lag.1     -2.821e-02  1.742e-02  -1.620    0.107
## tt           9.382e-05  7.500e-05   1.251    0.212
## z.diff.lag  -1.077e-01  7.042e-02  -1.529    0.128
##
## Residual standard error: 0.0394 on 203 degrees of freedom
## Multiple R-squared:  0.02698,    Adjusted R-squared:  0.0126
## F-statistic: 1.876 on 3 and 203 DF,  p-value: 0.1348
##
##
## Value of test-statistic is: -1.6196 2.0259 1.3129
```

```
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

```
# the null is always random walk with drift/null

#########################################################
####### I also included DF for gold_search #####
#########################################################

# augmented df test on only the gold search
df_test_gold_search <- urca::ur.df(gold_search_scaled, type = c('trend'),
                      selectlags = 'BIC')
summary(df_test_gold_search)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression trend
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 + 1 + tt + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.28046 -0.02637 -0.00703  0.00958  0.45812
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.0101620  0.0101515  -1.001 0.318001
## z.lag.1     -0.1821778  0.0453131  -4.020 8.19e-05 ***
## tt           0.0004192  0.0001244   3.371 0.000897 ***
## z.diff.lag  -0.0896148  0.0708442  -1.265 0.207337
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07057 on 203 degrees of freedom
## Multiple R-squared:  0.1063, Adjusted R-squared:  0.09308
## F-statistic: 8.047 on 3 and 203 DF,  p-value: 4.307e-05
##
##
## Value of test-statistic is: -4.0204 5.6202 8.2243
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau3 -3.99 -3.43 -3.13
## phi2  6.22  4.75  4.07
## phi3  8.43  6.49  5.47
```

For the price we cannot reject the null of a non-stationary process (the random walk with drift+trend is the null), seems to fit conventional wisdom on prices. For the search index we reject the null given the data.

For both, the null cannot be rejected given the data, the null is non-stationarity. Not very unexpected for prices, as they are often thought about as following a random walk and thus being non-stationary. But we can also look at difference-stationarity to check.

```
# augmented df test on only the differenced gold price
df_test_gold_price_FD <- urca::ur.df(gold_price_scaled_FD, type = 'none',
                                     selectlags = 'BIC')
summary(df_test_gold_price_FD)
```

```
##
## ###############################################
## # Augmented Dickey-Fuller Test Unit Root Test #
## ###############################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.139705 -0.013946  0.004984  0.026891  0.129077
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -1.19119    0.10475 -11.372   <2e-16 ***
## z.diff.lag  0.06367    0.07010   0.908    0.365
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0393 on 204 degrees of freedom
## Multiple R-squared:  0.562,  Adjusted R-squared:  0.5577
## F-statistic: 130.9 on 2 and 204 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -11.3722
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

```
#######################################################
####### Same here, I included DF for gold_search #####
#######################################################

# augmented df test on only the differenced gold price
df_test_gold_search_FD <- urca::ur.df(gold_search_scaled_FD, type = 'none',
                                     selectlags = 'BIC')
summary(df_test_gold_search_FD)
```

```
##
## ###############################################
```

```
## # Augmented Dickey-Fuller Test Unit Root Test #
## ################################################
##
## Test regression none
##
##
## Call:
## lm(formula = z.diff ~ z.lag.1 - 1 + z.diff.lag)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29183 -0.01619 -0.00326  0.01282  0.48462
##
## Coefficients:
##            Estimate Std. Error t value Pr(>|t|)
## z.lag.1    -1.51101    0.10758  -14.04  < 2e-16 ***
## z.diff.lag  0.25435    0.07026    3.62 0.000371 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07044 on 204 degrees of freedom
## Multiple R-squared:  0.6264, Adjusted R-squared:  0.6227
## F-statistic:   171 on 2 and 204 DF,  p-value: < 2.2e-16
##
##
## Value of test-statistic is: -14.0449
##
## Critical values for test statistics:
##       1pct  5pct 10pct
## tau1 -2.58 -1.95 -1.62
```

As the test rejects, given the data we cannot say that the data is not stationary. Which gives evidence for both being I(1).

```
# VAR model with unscaled prices
# save variable vectors as time series format:
gold_price <- ts(gold_pr$GOLDPMGBD228NLBM, frequency = 12,
                         start = c(2004, 1), end = c(2021, 5))
gold_search <- ts(data$GOLD, frequency = 12,
                          start = c(2004,1), end = c(2021,5))

# set up data for estimation using `VAR()`
VAR_data <- window(ts.union(gold_price, gold_search),
                   start = c(2004, 1), end = c(2021, 5))

# estimate model coefficients using `VAR()`
VAR_est <- VAR(y = VAR_data, p = 1, type = 'both')
summary(VAR_est)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: gold_price, gold_search
## Deterministic variables: both
## Sample size: 208
```

```
## Log Likelihood: -1799.017
## Roots of the characteristic polynomial:
## 0.9694 0.7759
## Call:
## VAR(y = VAR_data, p = 1, type = "both")
##
##
## Estimation results for equation gold_price:
## ===========================================
## gold_price = gold_price.l1 + gold_search.l1 + const + trend
##
##                Estimate Std. Error t value Pr(>|t|)
## gold_price.l1   0.96788    0.01798  53.837   <2e-16 ***
## gold_search.l1  0.10532    0.50011   0.211   0.8334
## const          25.95499   13.78646   1.883   0.0612 .
## trend           0.15617    0.13017   1.200   0.2317
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 62.32 on 204 degrees of freedom
## Multiple R-Squared: 0.9772,  Adjusted R-squared: 0.9769
## F-statistic:  2913 on 3 and 204 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation gold_search:
## ============================================
## gold_search = gold_price.l1 + gold_search.l1 + const + trend
##
##                 Estimate Std. Error t value Pr(>|t|)
## gold_price.l1   0.002848   0.001578   1.805   0.0725 .
## gold_search.l1  0.777456   0.043888  17.715   <2e-16 ***
## const          -2.158855   1.209834  -1.784   0.0758 .
## trend           0.023485   0.011423   2.056   0.0411 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 5.469 on 204 degrees of freedom
## Multiple R-Squared: 0.8359,  Adjusted R-squared: 0.8335
## F-statistic: 346.5 on 3 and 204 DF,  p-value: < 2.2e-16
##
##
##
## Covariance matrix of residuals:
##             gold_price gold_search
## gold_price      3883.7      -12.50
## gold_search      -12.5       29.91
##
## Correlation matrix of residuals:
##             gold_price gold_search
## gold_price     1.00000    -0.03669
## gold_search   -0.03669     1.00000
```

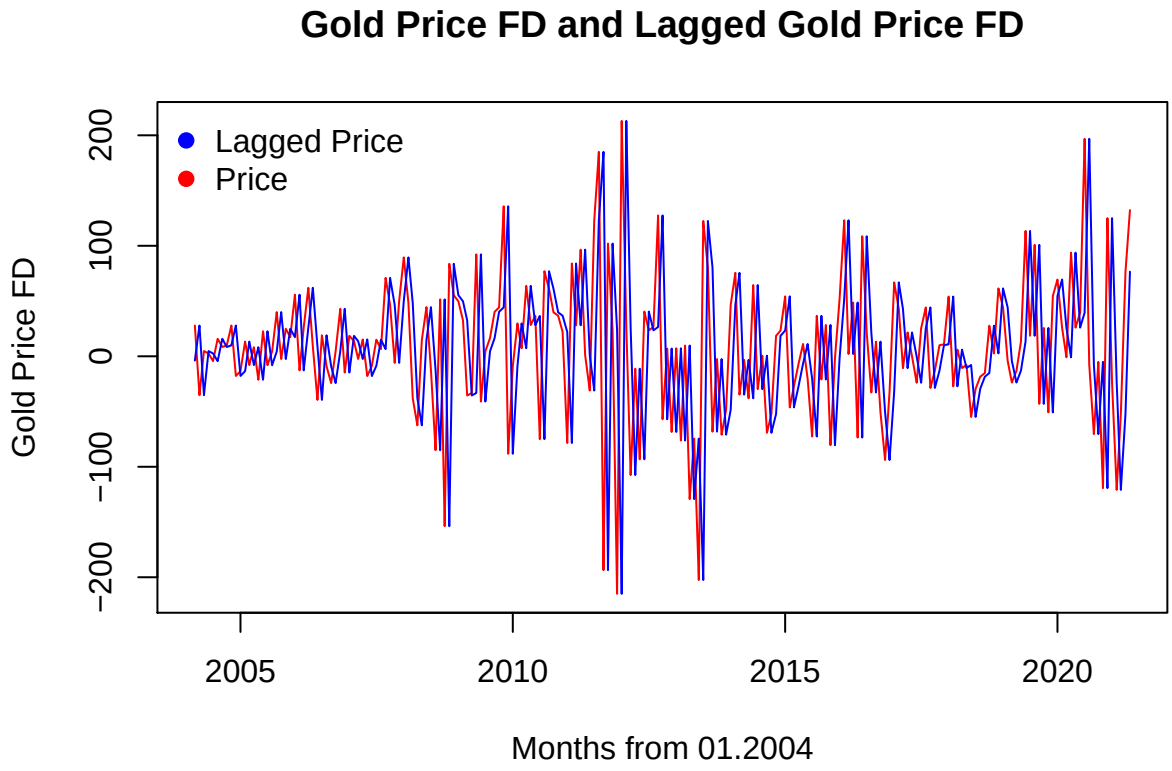
```
#########################################################
####### Sollten wir hier beim AR(1) nicht die First differences verwenden.
# Weil wir ja einen I(1) prozess haben. Und sollten wir nicht einfach mit
# dem besteren ARMA modell arbeiten und nicht AR(1) ? #####
#########################################################

# compare the VAR to the AR(1) model for the prices
T <-length(gold_price)
gold_price_2 <- as.numeric(gold_price[-1])
gold_price_lagged <- as.numeric(gold_price[-T])

plot(y=gold_price_2,x=gold_pr$DATE[1-209], type = 'l', lwd = 1, col = 'red',
     main = 'Gold Price and Lagged Gold Price',
     ylab = 'Gold Price', xlab = 'Months from 01.2004')
lines(y=gold_price_lagged,x=gold_pr$DATE[1-209], lwd = 1, col = 'blue')
legend('topleft', legend = c('Lagged Price','Price'),
       col = c('blue','red'), bty = "n", pch = c(19,19))
```


```
# estimate model
gold_price_AR1 <- lm(gold_price_2 ~ gold_price_lagged)
# estimate robust standard errors
coeftest(gold_price_AR1, vcov. = vcovHC, type = "HC1")
```

```
##
## t test of coefficients:
##
```

```
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       20.49216   10.59726  1.9337  0.05452 .
## gold_price_lagged  0.98841    0.01114 88.7232  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#########################################################
# Hier mal die abgeänderte Version mit AR(1) und FD #####



# compare the VAR to the AR(1) model for the prices first-differences
T <-length(gold_price_FD)
gold_price_FD_2 <- as.numeric(gold_price_FD[-1])
gold_price_FD_lagged <- as.numeric(gold_price_FD[-T])

plot(y=gold_price_FD_2,x=gold_pr$DATE[3:209], type = 'l', lwd = 1, col = 'red',
     main = 'Gold Price FD and Lagged Gold Price FD',
     ylab = 'Gold Price FD', xlab = 'Months from 01.2004')
lines(y=gold_price_FD_lagged,x=gold_pr$DATE[3:209], lwd = 1, col = 'blue')
legend('topleft', legend = c('Lagged Price','Price'),
       col = c('blue','red'), bty = "n", pch = c(19,19))
```




```
# estimate model
gold_price_FD_AR1 <- lm(gold_price_FD_2 ~ gold_price_FD_lagged)
# estimate robust standard errors
coeftest(gold_price_FD_AR1, vcov. = vcovHC, type = "HC1")
```

```
##
## t test of coefficients:
##
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)            8.066698   4.377330  1.8428   0.0668 .
## gold_price_FD_lagged -0.121139   0.096469 -1.2557   0.2106
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The values on the intercept seem to differ, but the estimated coefficient on the lag seems to fit.

```
# verify the 'by-hand' results with built-in function
ar.ols(gold_price, order.max = 1, intercept = T)
```

```
##
## Call:
## ar.ols(x = gold_price, order.max = 1, intercept = T)
##
## Coefficients:
##      1
## 0.9884
##
## Intercept: 7.171 (4.301)
##
## Order selected 1  sigma^2 estimated as  3848
```

```
forecast::auto.arima(gold_price, ic = 'aic')
```

```
## Series: gold_price
## ARIMA(0,1,1) with drift
##
## Coefficients:
##           ma1   drift
##       -0.1411  7.1279
## s.e.   0.0740  3.6766
##
## sigma^2 estimated as 3842:  log likelihood=-1152.52
## AIC=2311.05   AICc=2311.16   BIC=2321.06
```

The last model is automated to difference such that the data is stationary, then the function finds the best forecasting model via the AIC. Here this would be an ARMA(0,1) model:

$$\widehat{\Delta \text{gold price}}_t = \underset{3.6766}{(7.1279)} + \epsilon_t + \underset{(0.0740)}{(-0.1411)}\epsilon_{t-1}$$

4 Phillips-Ouliaris Cointegration Test

Q: Ist es richtig hier VAR_data zu verwenden?

```
po.test(VAR_data, demean = TRUE, lshort = TRUE)
```

```
##
##  Phillips-Ouliaris Cointegration Test
##
## data:  VAR_data
## Phillips-Ouliaris demeaned = -19.431, Truncation lag parameter = 2,
## p-value = 0.06318
```

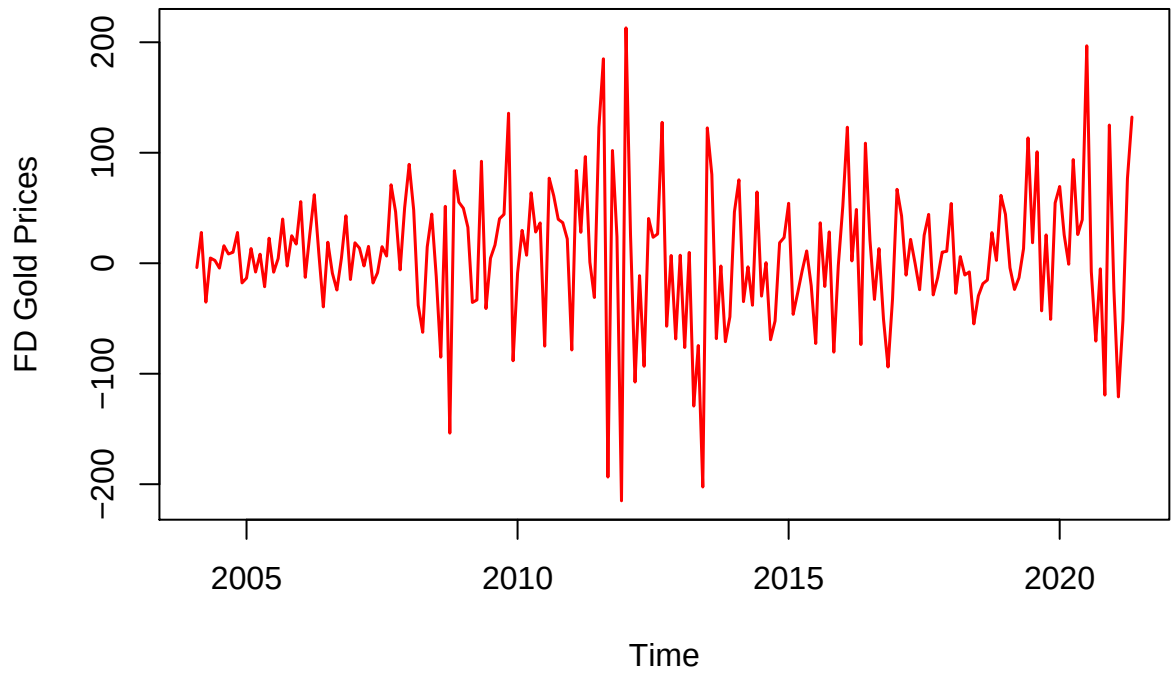
We cannot reject the null of the residuals being I(1). Thus we cannot rule out the case of a spurious or unbalanced regression. Note: as we only have one I(1) process and one non I(1) process this test makes no real sense.

Sollten wir hier nicht mit MA(1) weiterarbeiten? Ich habs mal weiter unten mit MA(1) gemacht. Den AR(1) part aber noch nicht gelöscht

5 Differenced AR(1) and ARCH Model for Gold Prices

TODO: add ARCH, EGARCH, etc.

```
# Differenced AR(1) and ARCH Model for Gold Prices
gold_price_FD <- ts(gold_price_FD, frequency = 12,
                    start = c(2004, 2), end = c(2021, 5))
plot(gold_price_FD, ylab = 'FD Gold Prices', col = 'red', lwd = 1.5)
```



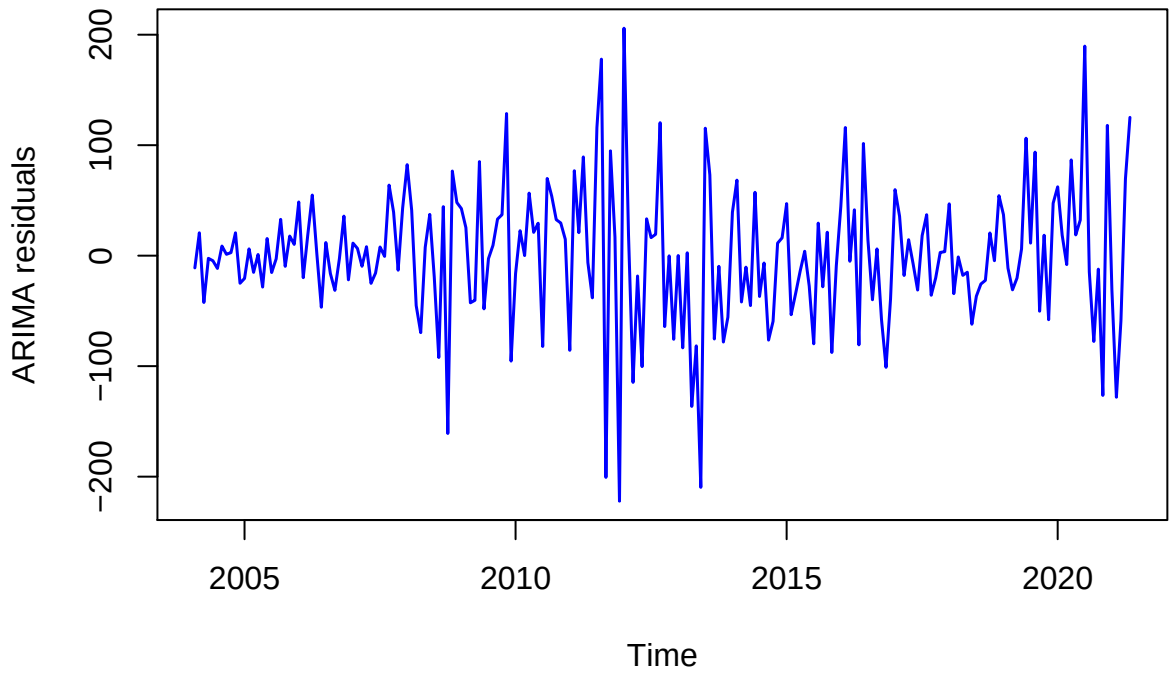

```
ar1mod_FD <- arima(gold_price_FD, order = c(1,0,0))
ar1mod_FD
```

```
##
## Call:
## arima(x = gold_price_FD, order = c(1, 0, 0))
##
## Coefficients:
##           ar1  intercept
##       -0.1205     7.1540
## s.e.   0.0694     3.8237
```

```
##
## sigma^2 estimated as 3814:  log likelihood = -1152.78,  aic = 2311.57
```

```
plot(forecast::arima.errors(ar1mod_FD),type = 'l', lwd = 1.5, col = 'blue',
     ylab = 'ARIMA residuals')
```

```
## Deprecated, use residuals.Arima(object, type='regression') instead
```




```
mean(forecast::arima.errors(ar1mod_FD))
```

```
## Deprecated, use residuals.Arima(object, type='regression') instead
```

```
## [1] 0.05853392
```

Going by the plot, it does not appear that the variance of the residuals is constant over time but rather has times of higher and lower volatility.

```
resi_ar1_FD_2 <- (forecast::arima.errors(ar1mod_FD))^2
```

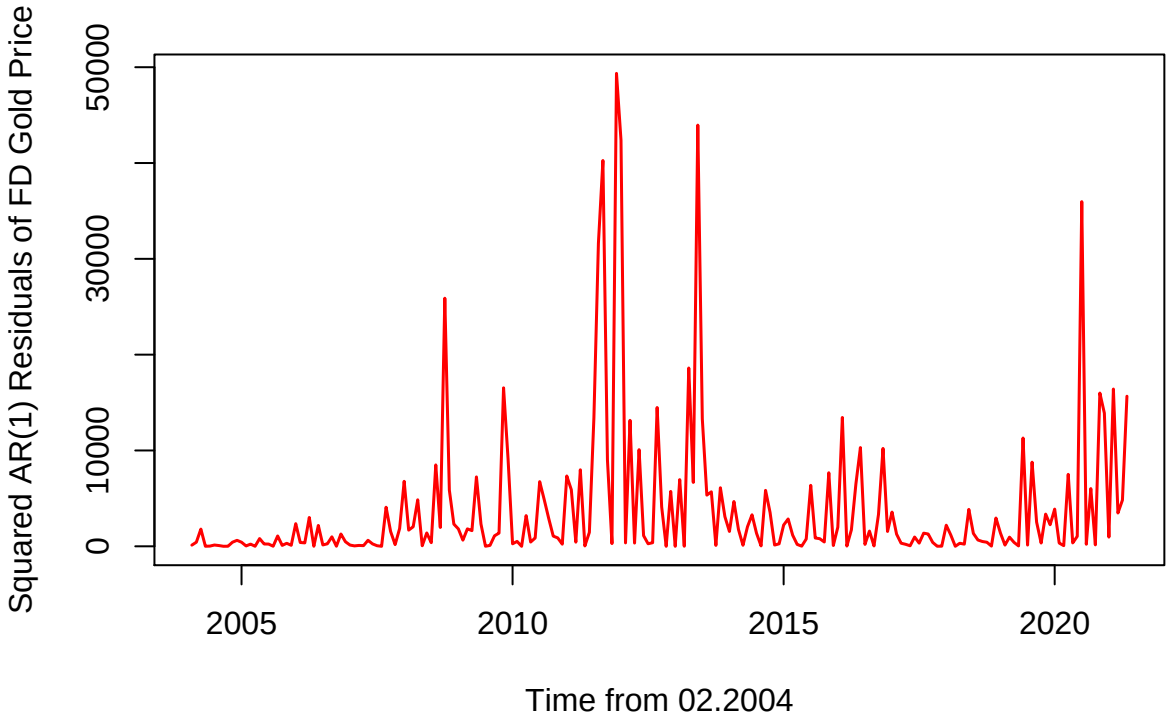
```
## Deprecated, use residuals.Arima(object, type='regression') instead
```

```
resi_arch1_FD_2_model <- arima(resi_ar1_FD_2, order = c(1,0,0))
resi_arch1_FD_2_model
```

```
##
## Call:
## arima(x = resi_ar1_FD_2, order = c(1, 0, 0))
##
## Coefficients:
##          ar1  intercept
```

```
##       0.3095  3887.0806
## s.e.  0.0662   727.6653
##
## sigma^2 estimated as 52737140:  log likelihood = -2144.4,  aic = 4294.79
```

```
# plot the squared residuals:
plot(resi_ar1_FD_2, ylab = 'Squared AR(1) Residuals of FD Gold Price',
     xlab = 'Time from 02.2004', col = 'red', lwd = 1.5)
```

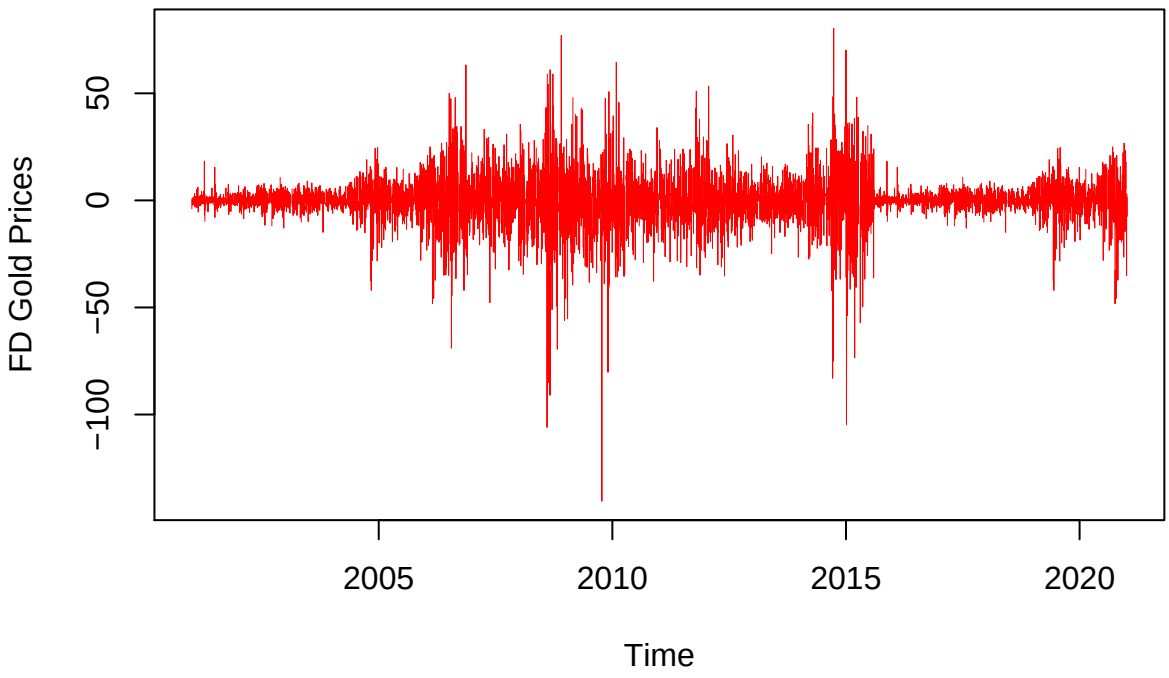


Mit MA(1), ändert nicht viel

6 Ideas for the first part: Univariate Time Series

7 Differenced MA(1) and ARCH Model for Gold Prices

```
# Differenced MA(1) and ARCH Model for Gold Prices


gold_price_FD <- ts(gold_daily_FD, frequency = 365,
                    start = c(2001, 1, 1), end = c(2021, 6, 10))
plot(gold_price_FD, ylab = 'FD Gold Prices', col = 'red', lwd = 0.5)
```

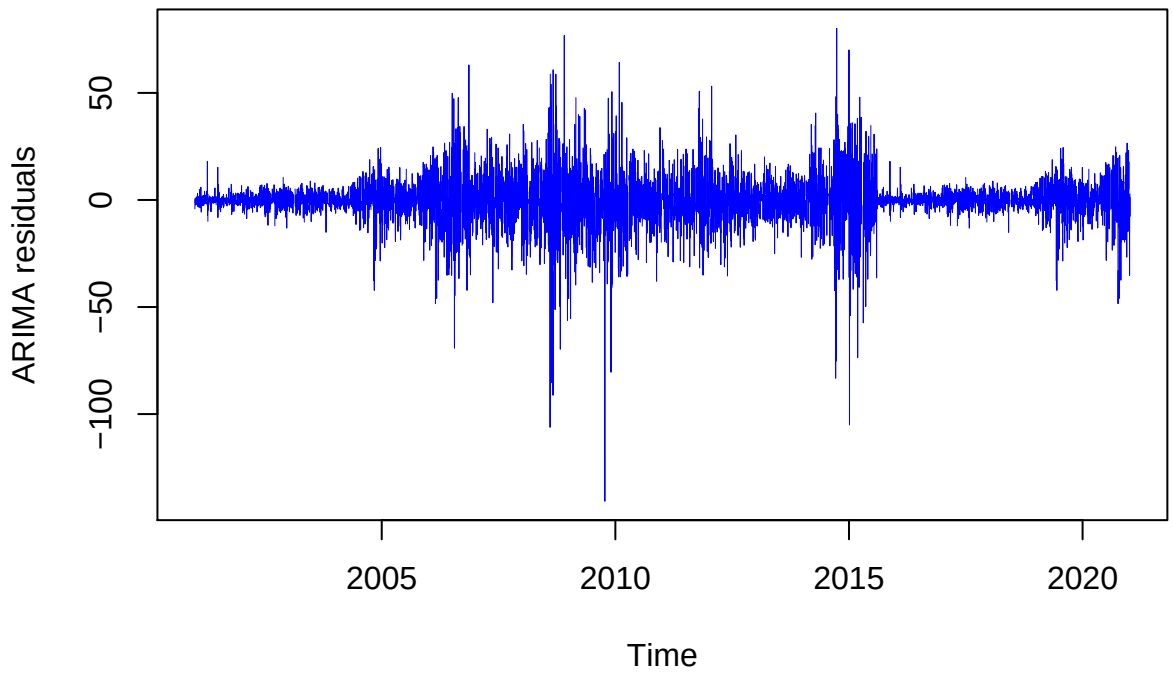



```
model_HF_FD <- forecast::auto.arima(gold_price_FD, ic = 'aic')
model_HF_FD
```

```
## Series: gold_price_FD
## ARIMA(4,0,2) with non-zero mean
##
## Coefficients:
##           ar1     ar2      ar3      ar4     ma1      ma2    mean
##       -0.0270  0.6155  -0.0033  -0.0384  0.0278  -0.6164  0.2022
## s.e.   0.2394  0.2277   0.0162   0.0162  0.2398   0.2285  0.1213
##
## sigma^2 estimated as 120.9:  log likelihood=-26138.47
## AIC=52292.95   AICc=52292.97   BIC=52348.12
```

```
plot(forecast::arima.errors(model_HF_FD),type = 'l', lwd = 0.5, col = 'blue',
     ylab = 'ARIMA residuals')
```

```
## Deprecated, use residuals.Arima(object, type='regression') instead
```



Going by the plot, it does not appear that the variance of the residuals is constant over time but rather has times of higher and lower volatility.

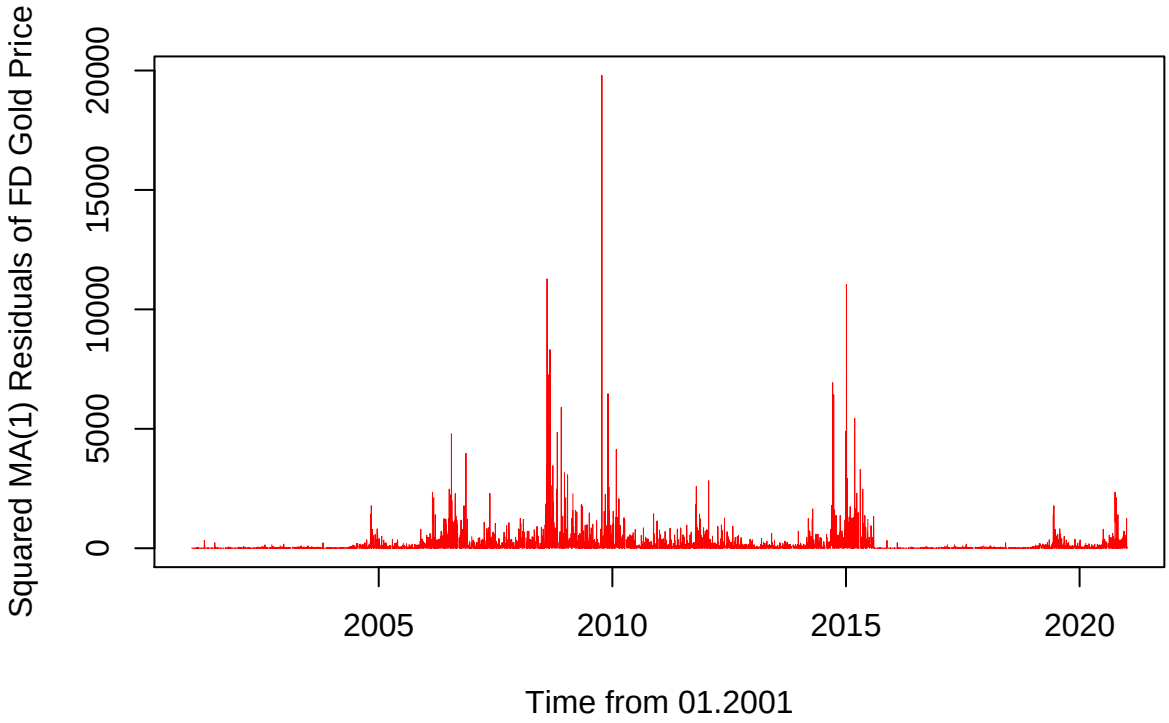
```
resi_ma1_FD_2 <- (forecast::arima.errors(model_HF_FD))^2
```

```
## Deprecated, use residuals.Arima(object, type='regression') instead
```

```
resi_arch1_FD_2_model <- arima(resi_ma1_FD_2, order = c(1,0,0))
resi_arch1_FD_2_model
```

```
##
## Call:
## arima(x = resi_ma1_FD_2, order = c(1, 0, 0))
##
## Coefficients:
##         ar1  intercept
##       0.112   121.0271
## s.e.  0.012     6.4950
##
## sigma^2 estimated as 229290:  log likelihood = -51994.71,  aic = 103995.4
```

```
# plot the squared residuals:
plot(resi_ma1_FD_2, ylab = 'Squared MA(1) Residuals of FD Gold Price',
     xlab = 'Time from 01.2001', col = 'red', lwd = 0.5)
```


Squared MA(1) Residuals of FD Gold Price
0
5000
10000
15000
20000
2005
2010
2015
2020
Time from 01.2001