Time Series Analysis - Lab 01 (Group 7)

Anubhav Dikshit (anudi287) and Maximilian Pfundstein (maxpf364)

2019-09-11

Contents

1 Computations with Simulated Data

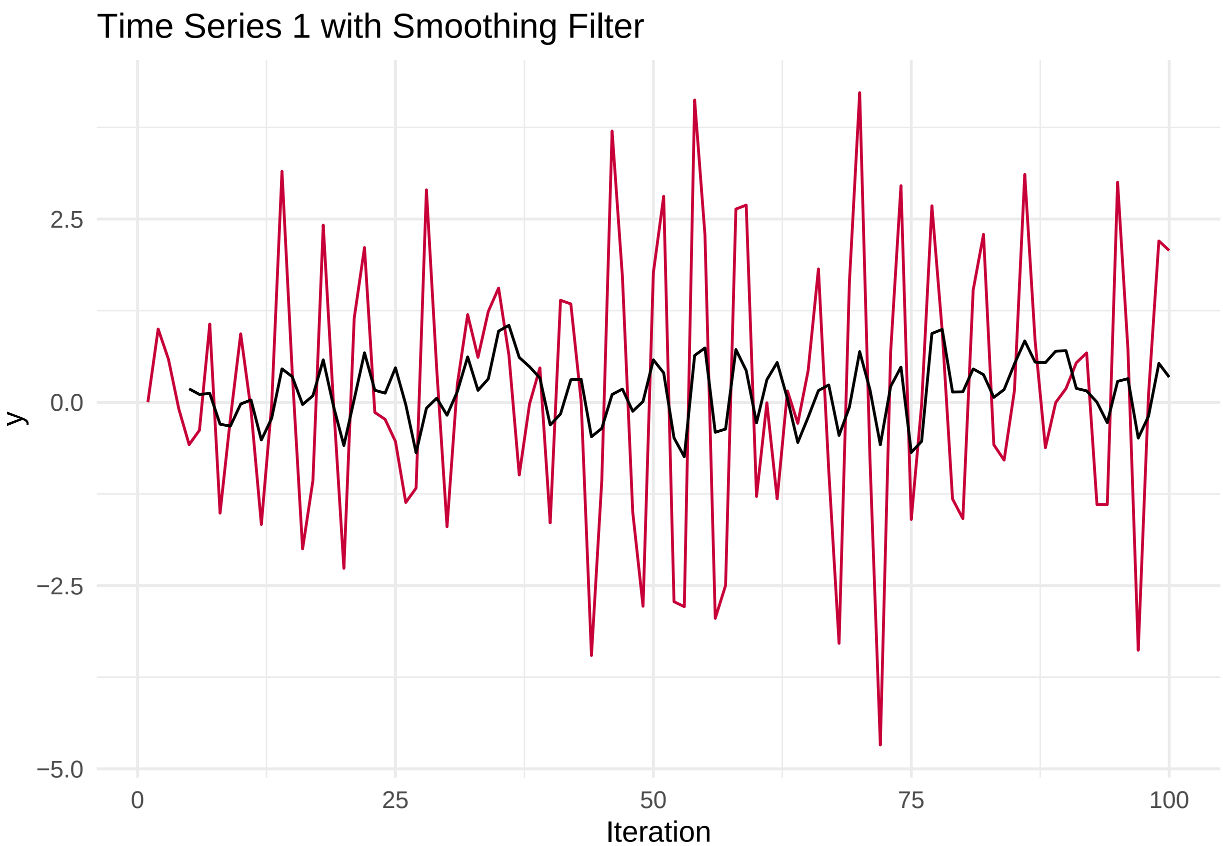
Task a): Generate two time series $x_t = -0.8x_{t-2} + w_t$, where $x_0 = x_1 = 0$ and $x_t = cos(\frac{2\pi t}{5})$ with 100 observations each.

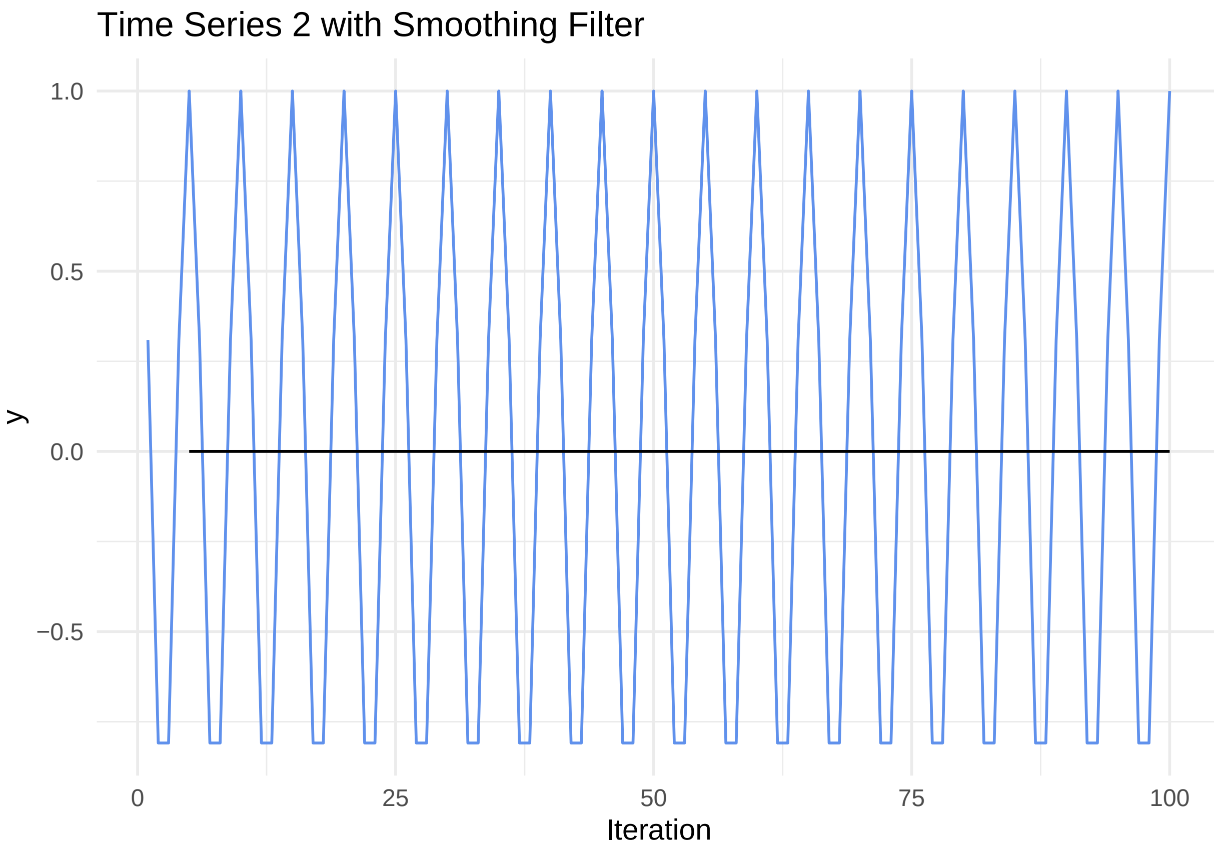
Answer: First we create two functions to sample `n` times from our time series. Then we apply a filter. As default a convoltuion is being used (moving average). With `sides = 1` only past values are considered, which makes sense for a time series.

```
################################################################################
# Exercise 1.a)
################################################################################

x0 = 0
x1 = 0

n = 100

# Series 1
generate_S1 = function(t, x0=0, x1=1) {

  series = vector(length = t)
  series[1] = x0
  series[2] = x1

  for (i in 3:t) {
    series[i] = -0.8 * series[i-2] + rnorm(n=1, mean=0, sd=1)
  }

  return(ts(series))
}

# Series 2
generate_S2 = function(t) {
  series = vector(length = t)
```

```
  for (i in 1:t) {
    series[i] = cos(2 * pi * i / 5)
  }

  return(ts(series))
}

index = c(1:n)

series1 = generate_S1(n)
series2 = generate_S2(n)

series1_filtered = stats::filter(series1, filter = rep(0.2, 5), sides = 1)
series2_filtered = stats::filter(series2, filter = rep(0.2, 5), sides = 1)
```

As we can see the time series have been smoothed a lot, removing extremes. The first smoothed series seems to be slightly shifted. For the second time series we obtain a straight line, as the moving average of an alternating series will be 0.





Task b): Consider time series $x_t - 4_{t-1} + 2x_{t-2} + x_{t-5} = w_t + 3x_{t-2} + w_{t-4} - 4w_{t-6}$. Write an appropriate R code to investigate whether this time series is casual and invertible.

```
################################################################################
# Exercise 1.b)
################################################################################

generate_S3 = function(t, X, W) {
  series = vector(length=t)
  white_noise = vector(length=t)

  series[1:length(X)] = X
  white_noise[1:length(W)] = W

  for (i in 7:t) {
    W[1:6] = W[2:7]
    W[7] = rnorm(1, mean=0, sd=1)
    series[i] = 4 * series[i-1] - 2 * series[i-2] - series[i-5] +
                W[7] + 3 * W[5] + W[2] - 4 * W[1]
  }

  return(ts(series))
}

series3 = generate_S3(t=n, X = rnorm(7, mean=0, sd=1), W = rnorm(7, mean=0, sd=1))
```

Causality

First we rewrite the given time series:

$$x_t = 4x_{t-1} - 2x_{t-2} - x_{t-5} + w_t + 3x_{t-2} + w_{t-4} - 4w_{t-6}$$

Applying the autoregressive operator gives us:

$$\phi(B) = 1 - 4B + 2B^2 + 0B^3 + 0B^4 + B^5$$

So Z_ϕ is given by:

$$Z_\phi = (1, -4, 2, 0, 0, 1)$$

We use the function `polyroot()` to see if any of the (complex) zero points lies withon the unit circle.

```
Z_phi = c(1, -4, 2, 0, 0, 1)

isOutsideUnitCircle = function(Z) {
  return(all(Mod(polyroot(Z)) > 1))
}

isOutsideUnitCircle(Z_phi)
```

```
## [1] FALSE
```

```
polyroot(Z_phi)
```

```
## [1]  0.2936658+0.000000i -1.6793817+0.000000i  1.0000000-0.000000i
## [4]  0.1928579-1.410842i  0.1928579+1.410842i
```

Invertibility

Using the autoregressive operator for θ, we get:

$$\theta(B) = 1 + 0B + 3B^2 + 0B^3 + B^4 + 0B^5 - 4B^6$$

So Z_θ is given by:

$$Z_\theta = (1, 0, 3, 0, -1, 0, -4)$$

```
Z_theta = c(1, 0, 3, 0, -1, 0, 4)

isOutsideUnitCircle(Z_theta)
```

```
## [1] FALSE
```

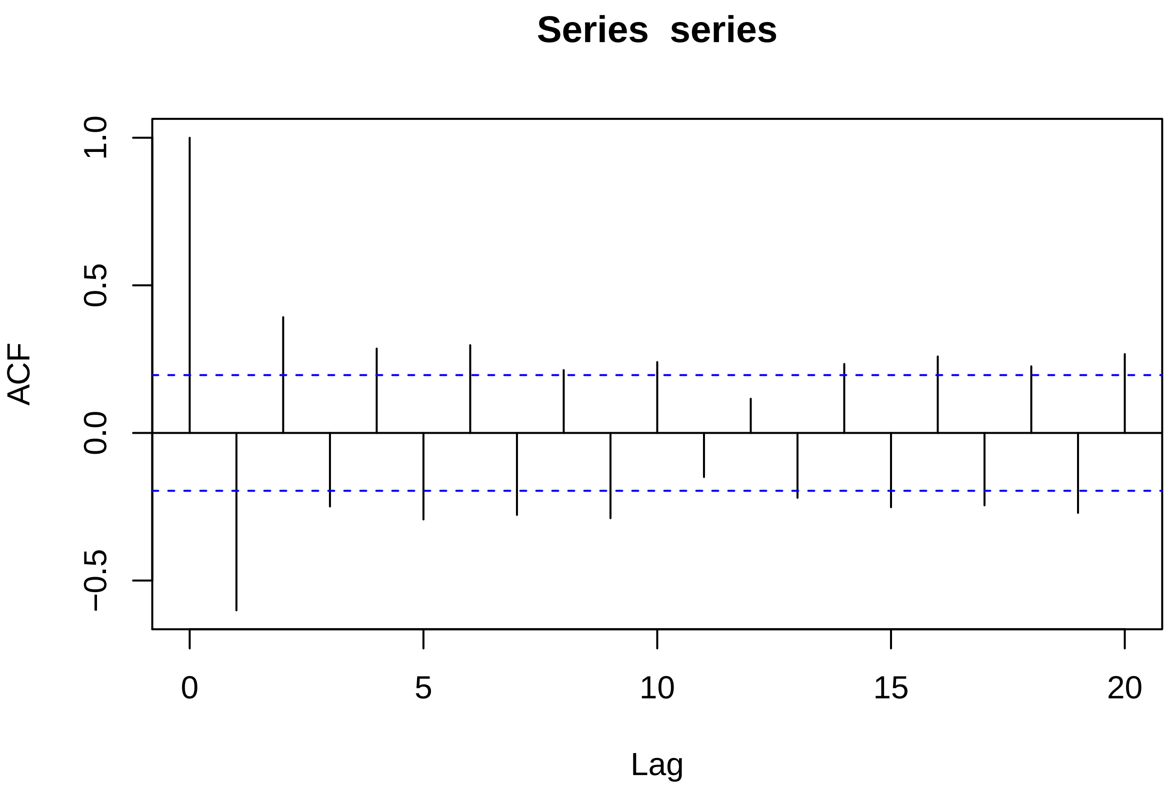
```
polyroot(Z_theta)
```

```
## [1]  0.0000000+0.5278436i -0.7783186+0.5843547i  0.0000000-0.5278436i
## [4]  0.7783186-0.5843547i  0.7783186+0.5843547i -0.7783186-0.5843547i
```

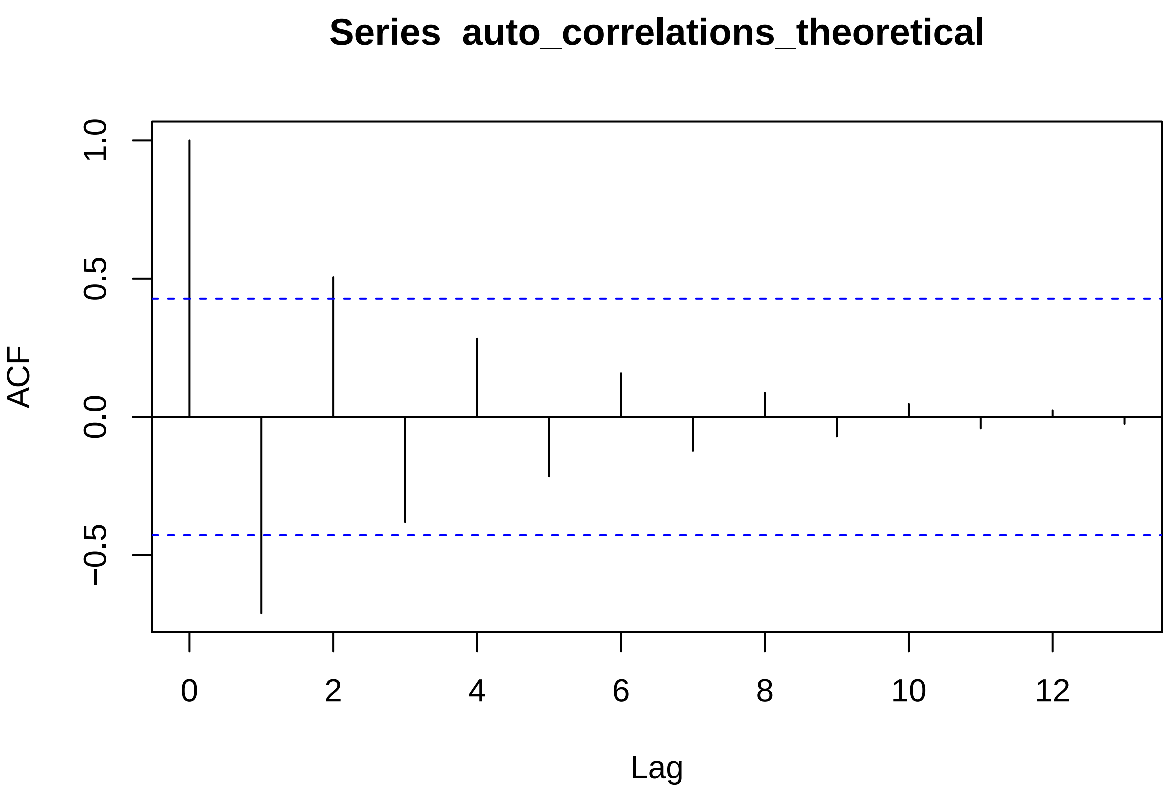
Task: Use built-in R functions to simulate 100 observations from the process $x_t + \frac{3}{4}x_{t-1} = w_t - \frac{1}{9}w_{t-2}$, compute sample ACF and theoretical ACF, use seed 54321. Compare the ACF plots.

```
set.seed(54321)

model = list(ar = c(-3/4), ma = c(0, -1/9))
```

```r
series = arima.sim(model = model, n = 100)

# Sample
auto_correlations_sample = acf(series)
```




```r
# Theoretical
auto_correlations_theoretical = ARMAacf(ar = model$ar, ma = model$ma,
                                        lag.max = 20)
acf(auto_correlations_theoretical)
```



We can see that the theoretical AC is between the blue lines after three iterations, while the sample AC exceeds the lines for a longer period of time.

2 Visualization, detrending and residual analysis of Rhine data

The data set `Rhine.csv` contains monthly concentrations of total nitrogen in the Rhine River in the period 1989-2002.

Task a): Import the data to R, convert it appropriately to *ts* object (use function `ts()`) and explore it by plotting the time series, creating scatter plots of x_t against $x_{t-1}, ..., x_{t-12}$. Analyze the time series plot and the scatter plots: Are there any trends, linear or seasonal, in the time series? When during the year is the concentration highest? Are there any special patterns in the data or scatterplots? Does the variance seem to change over time? Which variables in the scatterplots seem to have a significant relation to each other?

Answer: First, we import the data to R and take a look at it.

```
rhine = read_csv2("Rhine.csv")
```

```
## Using ',' as decimal and '.' as grouping mark. Use read_delim() for more control.
```

```
## Parsed with column specification:
## cols(
##   Year = col_double(),
##   Month = col_double(),
##   Time = col_double(),
##   TotN_conc = col_double()
## )
```

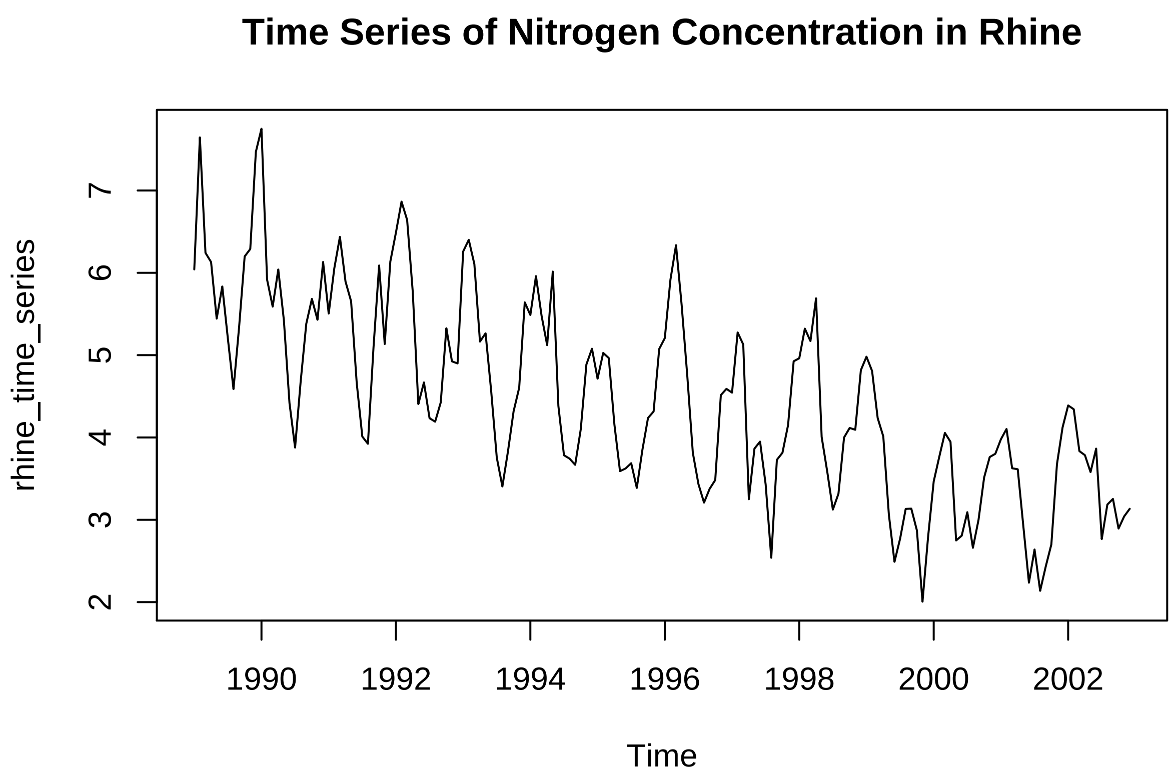
```
head(rhine)
```

```
## # A tibble: 6 x 4
```

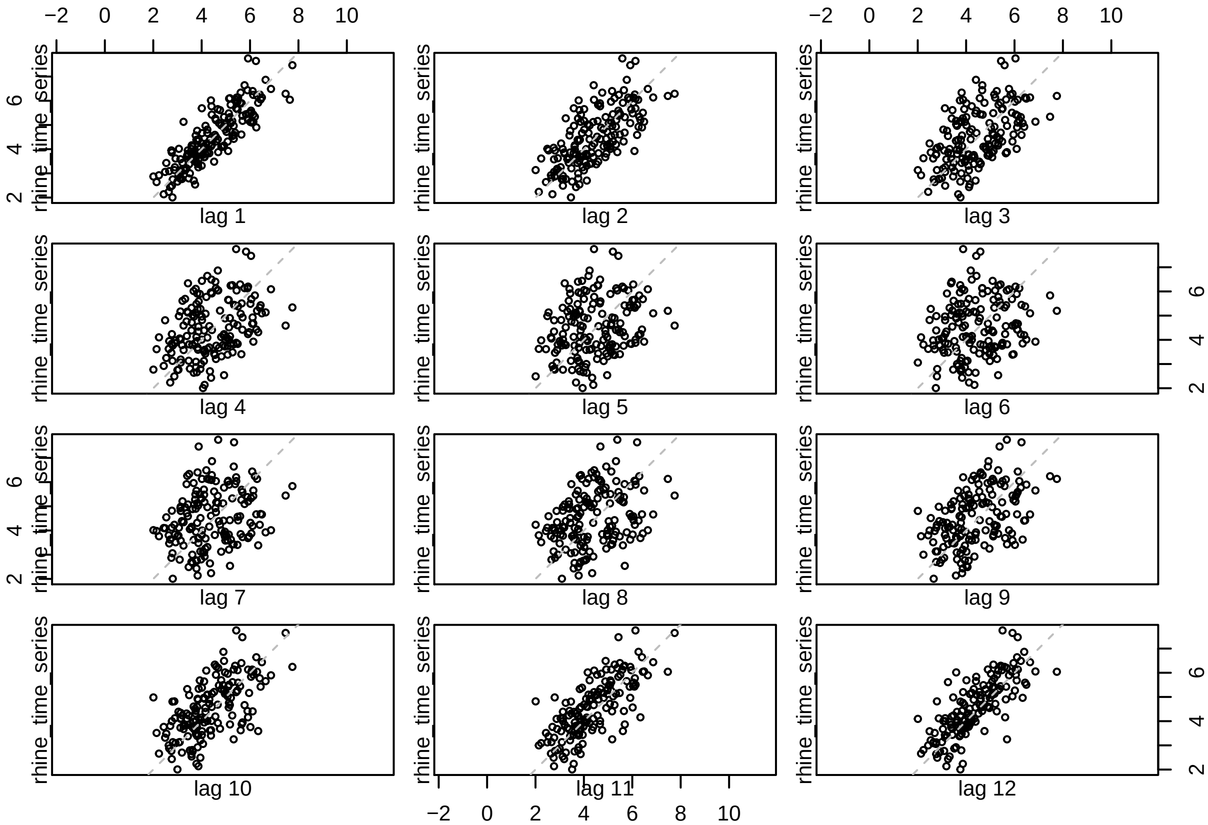
```
##     Year Month  Time TotN_conc
##    <dbl> <dbl> <dbl>     <dbl>
## 1  1989     1 1989.      6.04
## 2  1989     2 1989.      7.64
## 3  1989     3 1989.      6.24
## 4  1989     4 1989.      6.13
## 5  1989     5 1989.      5.44
## 6  1989     6 1989.      5.83
```

Now we make a time series object from the data and convert it to a time series.

```
rhine_time_series = ts(data = rhine$TotN_conc, start = c(1989,1),
                       frequency = 12)

# Normal Time Series
plot(rhine_time_series, main = "Time Series of Nitrogen Concentration in Rhine")
```

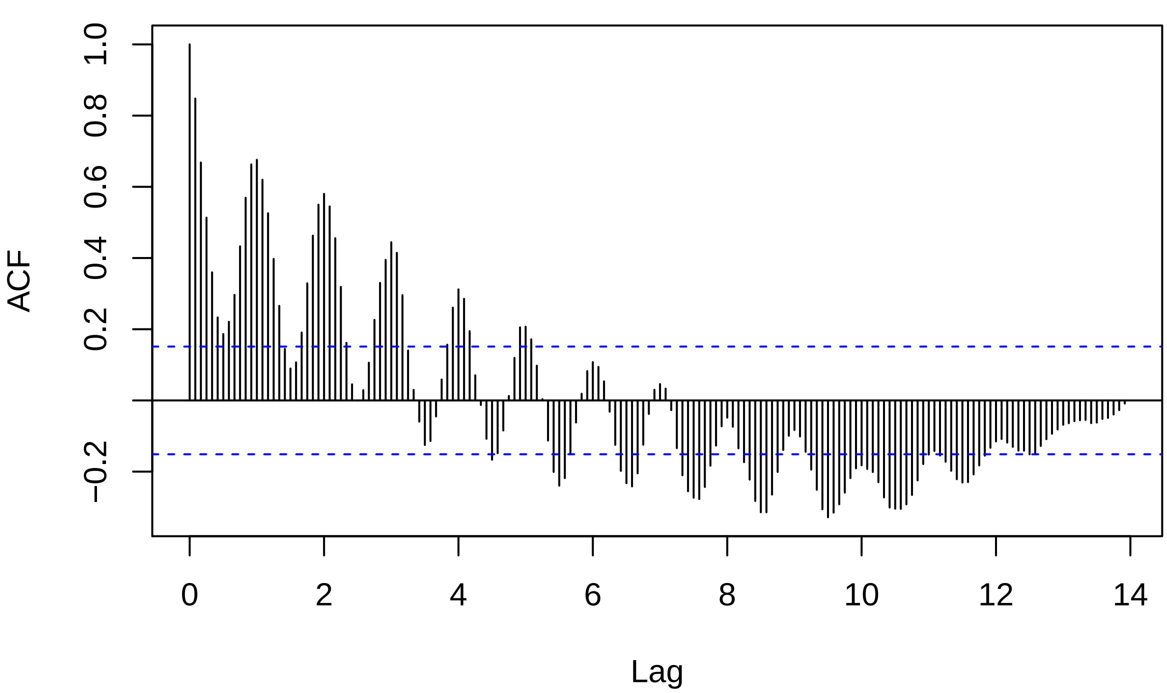



```
# 12 Lags as we have 12 month each year
lag.plot(rhine_time_series, lags = 12)
```




```r
# Autocovariance
acf(rhine_time_series, lag.max = nrow(rhine))
```

Series rhine_time_series



Q: Are there any trends, linear or seasonal, in the time series?

A: When looking directly at the time series, its clearly visible that we have a (linear) downwards-trend over the years. Also we identify the seasonal trend of the data.

Q: When during the year is the concentration highest?

A: The concentrations of total nitrogen is higher during winter and lower during summer.

Q: Are there any special patterns in the data or scatterplots?

A: Looking at the scatterplot we can see that at lag 1 we start with a high correlations which gets lower as the lag increases up to 6. After that the behaviour is reversed, the correlation now getting higher again towards a lag of 12. So here we can as well see the seasonal behaviour.

Q: Does the variance seem to change over time?

A: Yes, it seems to become lower over time. The difference between the minimum and maximum amount of concentrations seems to become lower as the years pass by.

Q: Which variables in the scatterplots seem to have a significant relation to each other?

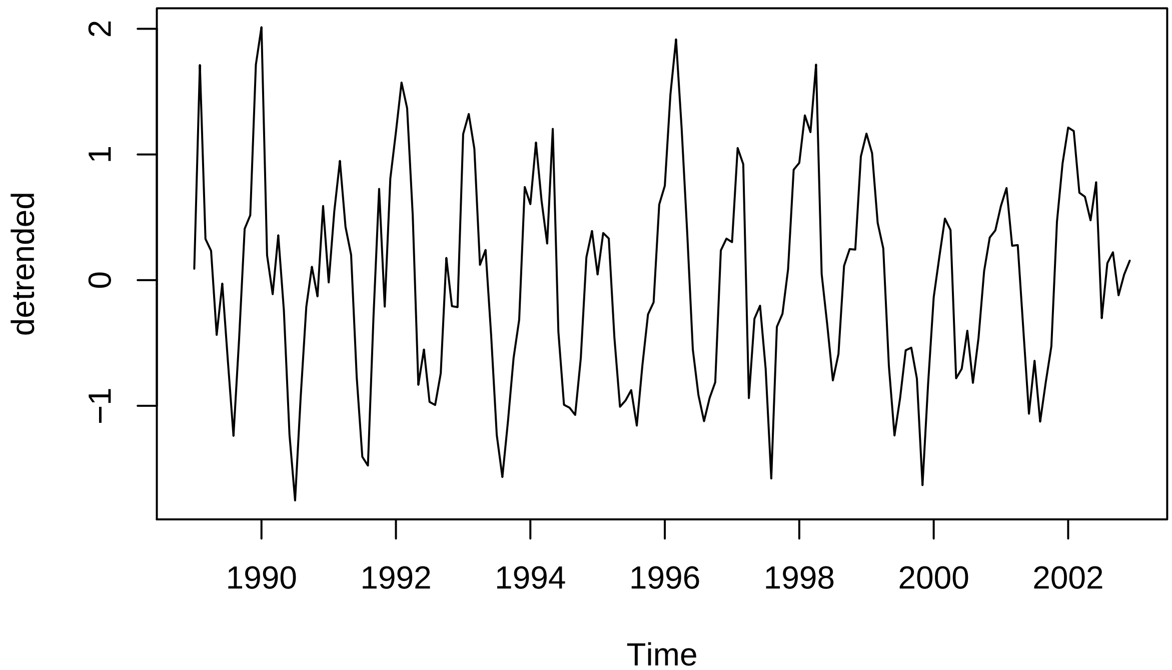
A: As already mentioned in the first question, around lag 1 and lag 12 the relation seems to be high, which makes sense as we have seen a seasonal trend. Lag 1 is kind of normal, even for a non-seasonal time series, but lag 12 suggests a seasonal behaviour.

Task b): Eliminate the trend by fitting a linear model with respect to t to the time series. Is there a significant time trend? Look at the residual pattern and the sample ACF of the residuals and comment how this pattern might be related to seasonality of the series.

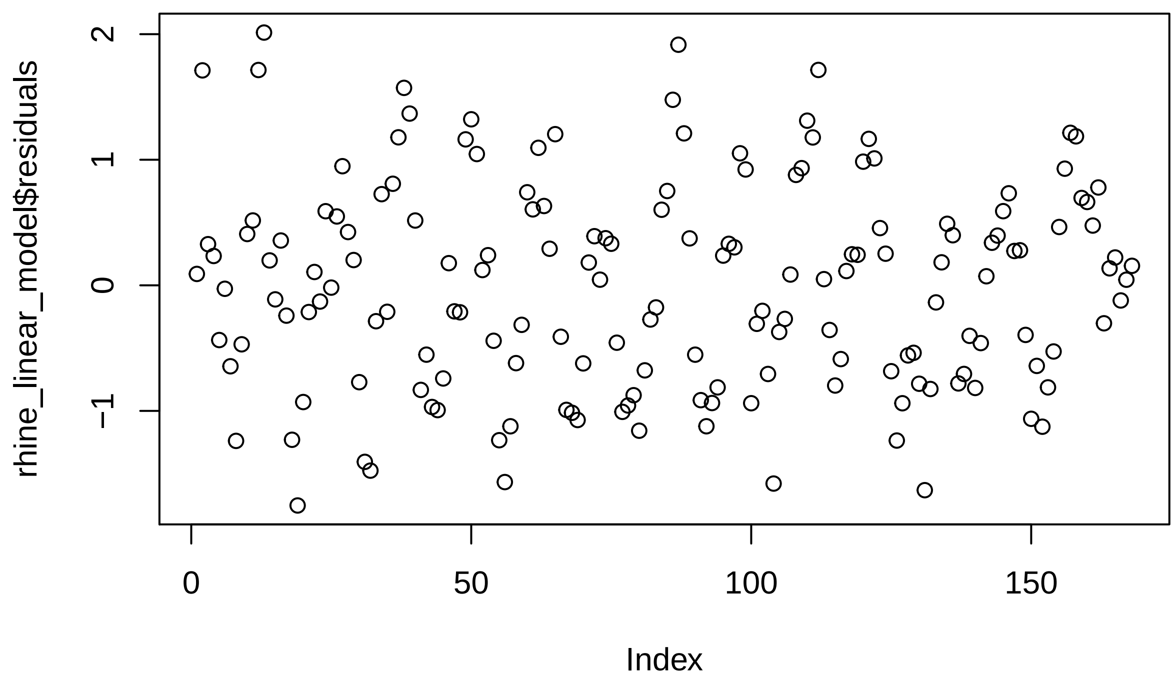
```r
# Linear Model
rhine_linear_model = lm(TotN_conc ~ Time, data=rhine)

summary(rhine_linear_model)
```

```
##
## Call:
## lm(formula = TotN_conc ~ Time, data = rhine)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.75325 -0.65296  0.06071  0.52453  2.01276
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 430.70725   31.26570   13.78   <2e-16 ***
## Time         -0.21355    0.01566  -13.63   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8205 on 166 degrees of freedom
## Multiple R-squared:  0.5282, Adjusted R-squared:  0.5254
## F-statistic: 185.9 on 1 and 166 DF,  p-value: < 2.2e-16
```

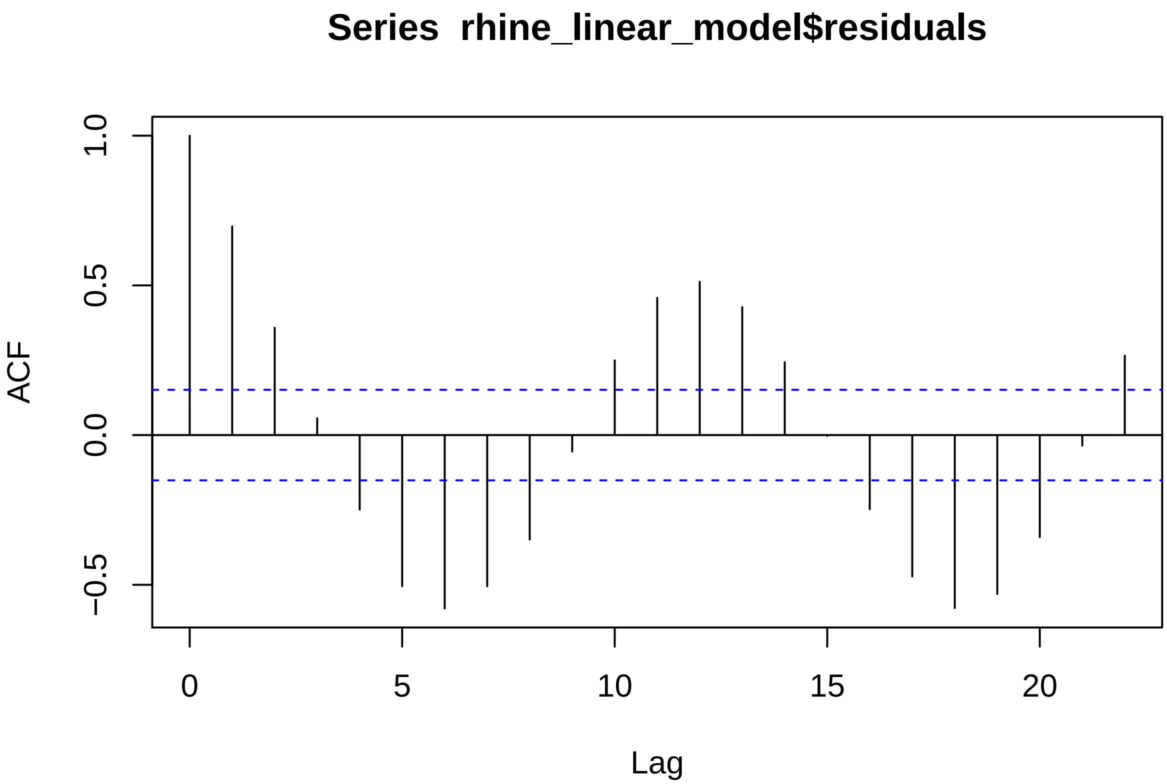
```r
# Difference
detrended = rhine_time_series - rhine_linear_model$fitted.values
plot(detrended)
```




```r
#plot(rhine_linear_model)
plot(rhine_linear_model$residuals)
```




```r
acf(rhine_linear_model$residuals)
```



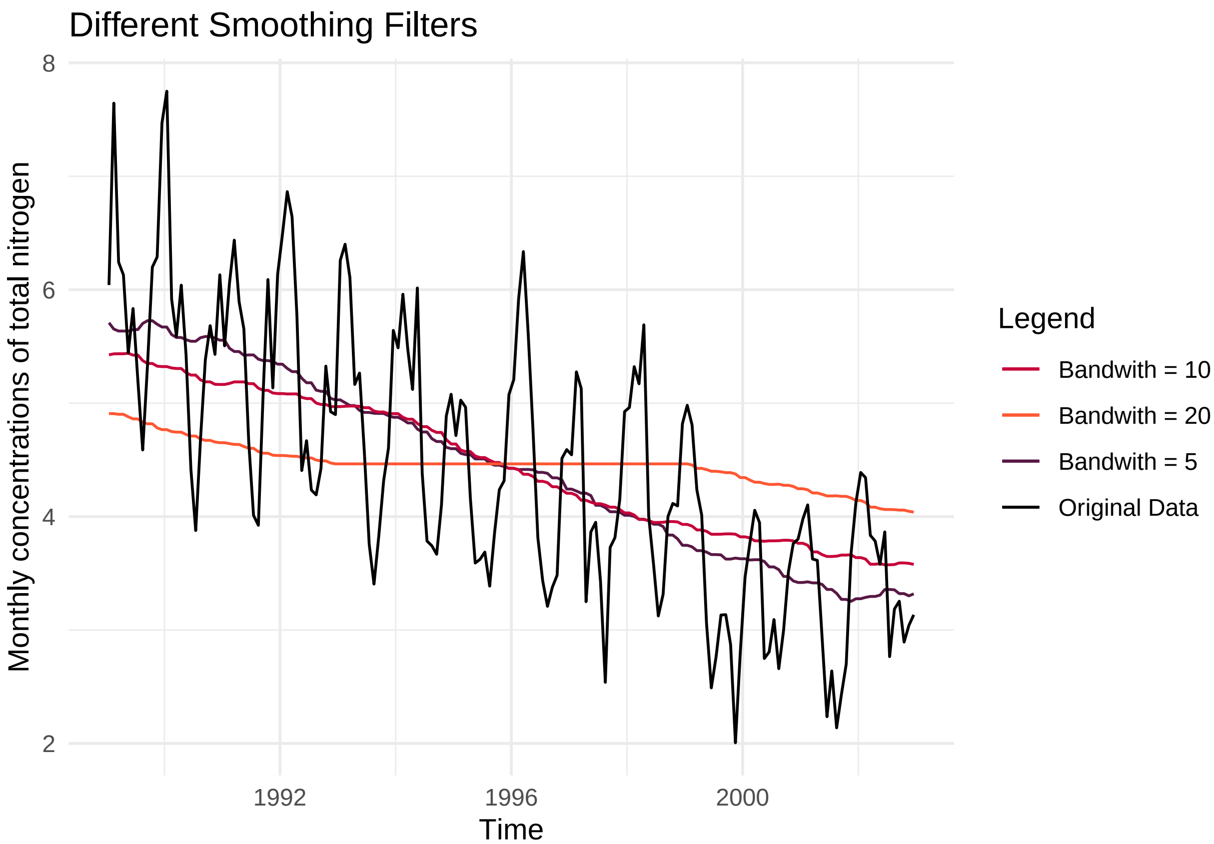

```
# Could also be decomposed by using this
#rhine_decomposed_additive = decompose(rhine_time_series, "additive")
#rhine_decomposed_multiplicative = decompose(rhine_time_series, "multiplicative")

# STL() would also be possible
```

Answer: The first picture shows the detrended data, containing only the seasonality and the error. Clearly, the seasonality is visible. When looking at the ACF of the residuals we can also observe the seasonality and we also see that it's getting lower over time. As the residuals are not normally distributed we assume that there is still more to explain by our model.

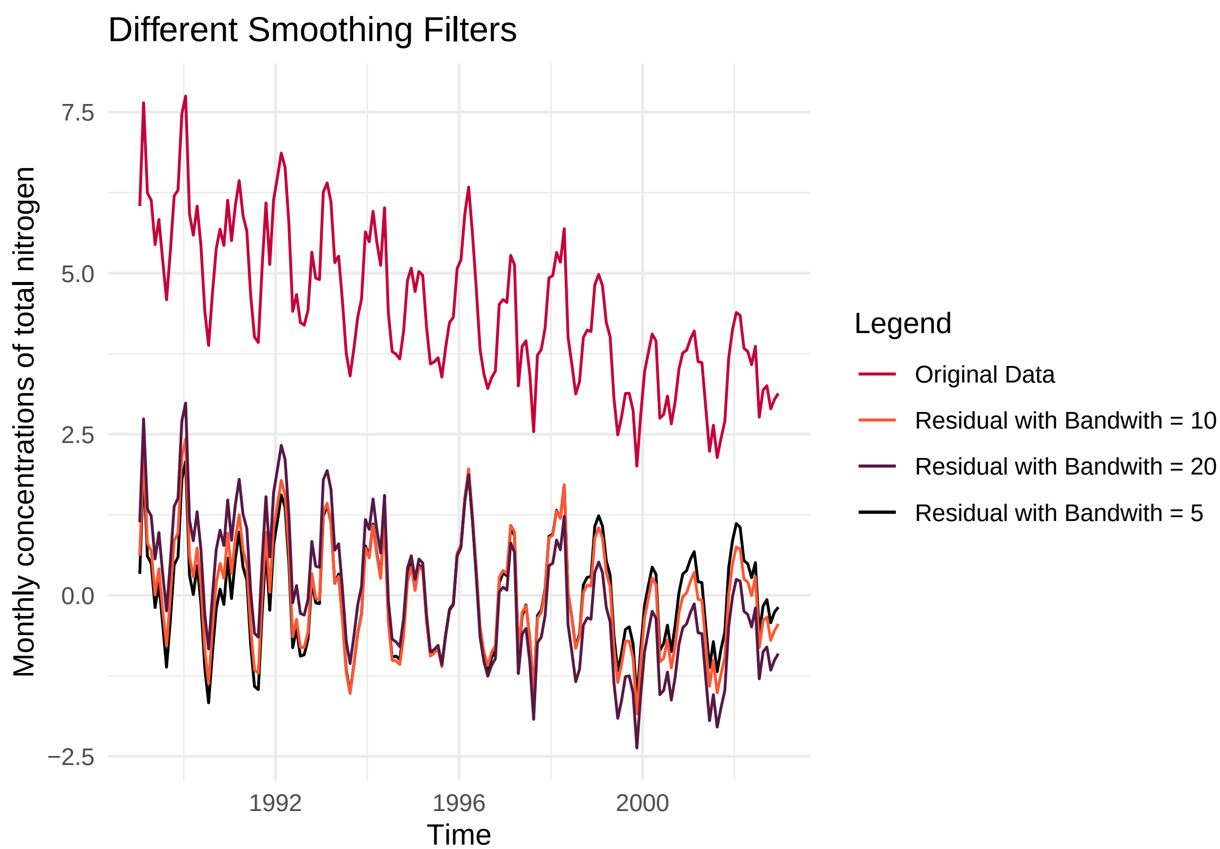
Task c): Eliminate the trend by fitting a kernel smoother with respect to t to the time series (choose a reasonable bandwidth yourself so the fit looks reasonable). Analyze the residual pattern and the sample ACF of the residuals and compare it to the ACF from step b). Conclusions? Do residuals seem to represent a stationary series?

```
rhine_time_series_smoothed_5 = ksmooth(x = rhine$Time,
                                       y = rhine$TotN_conc,
                                       bandwidth=5)

rhine_time_series_smoothed_10 = ksmooth(x = rhine$Time,
                                        y = rhine$TotN_conc,
                                        bandwidth=10)

rhine_time_series_smoothed_20 = ksmooth(x = rhine$Time,
                                        y = rhine$TotN_conc,
                                        bandwidth=20)

residual_k_5 = rhine_time_series - rhine_time_series_smoothed_5$y
residual_k_10 = rhine_time_series - rhine_time_series_smoothed_10$y
```

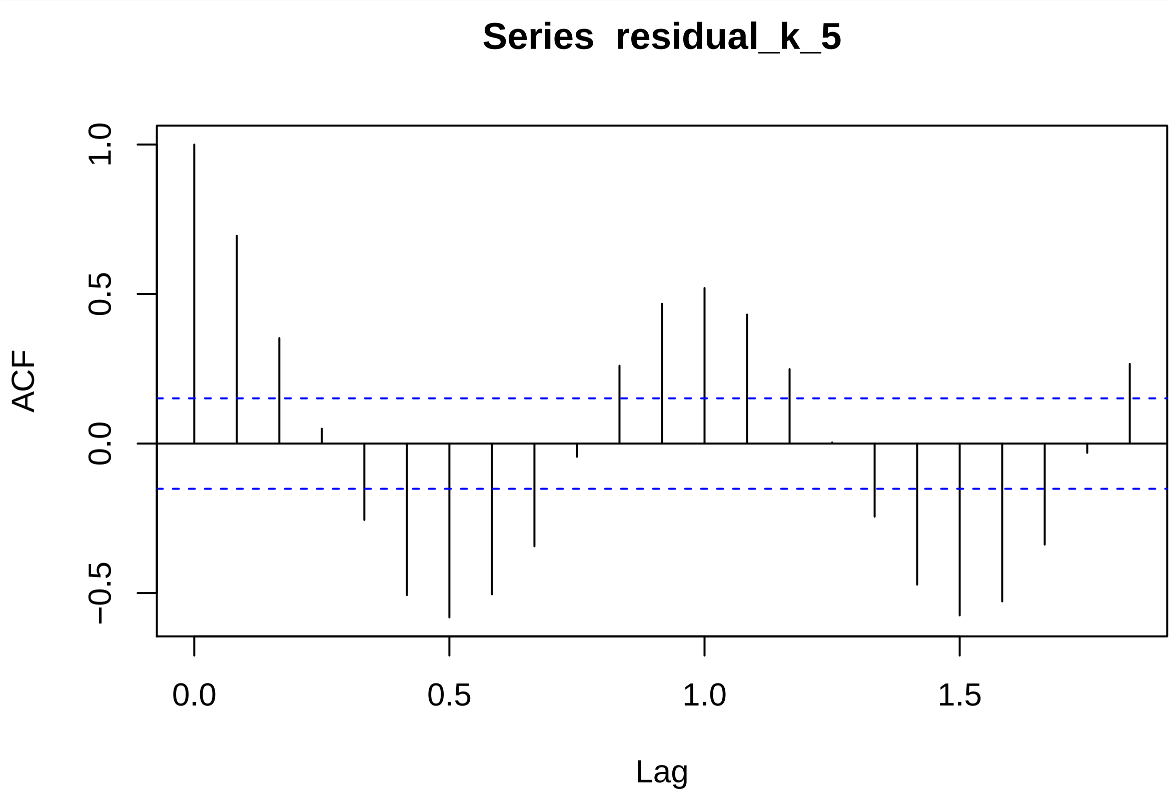
```r
residual_k_20 = rhine_time_series - rhine_time_series_smoothed_20$y
```



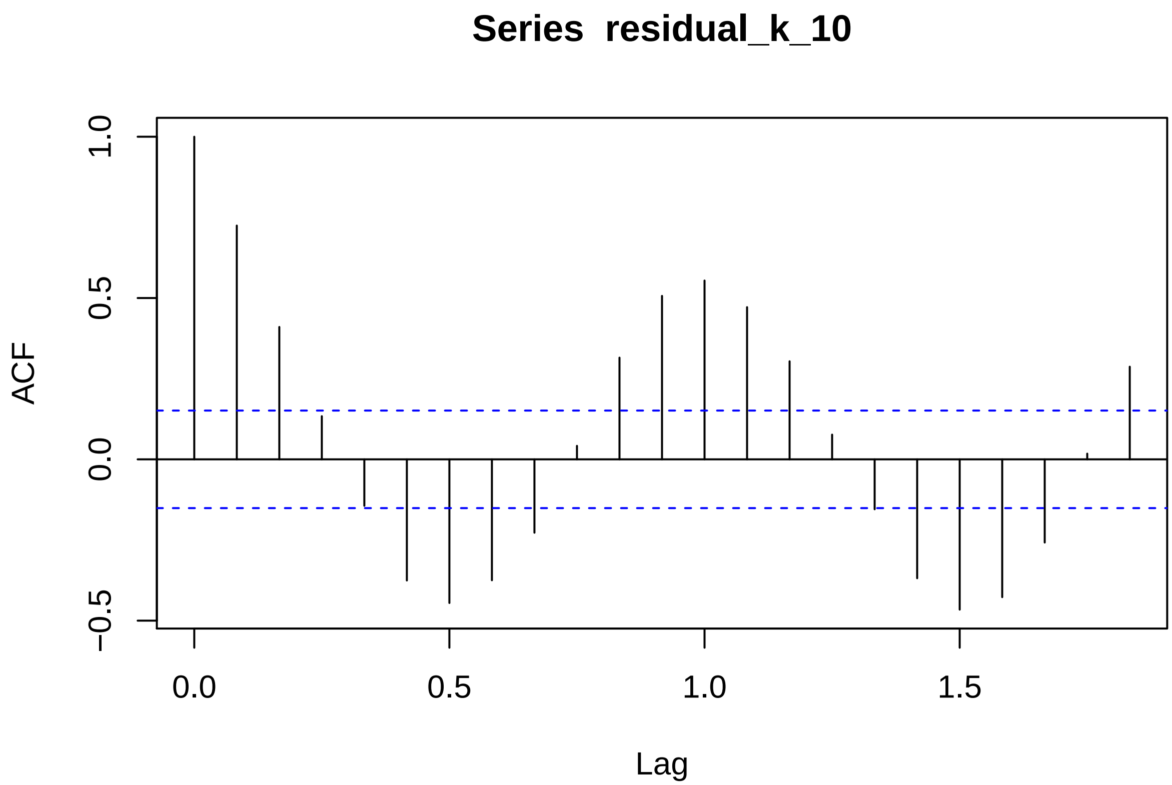

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



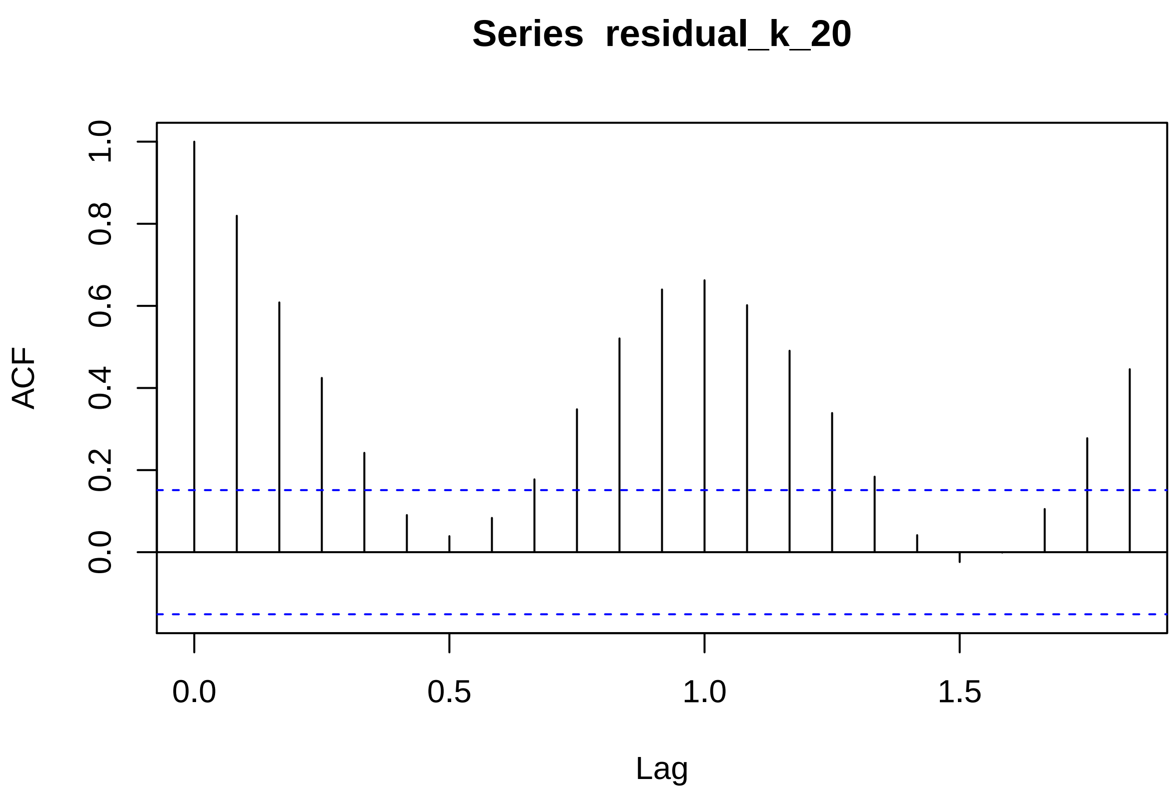

```
acf(residual_k_5)
```



```r
acf(residual_k_10)
```




```r
acf(residual_k_20)
```



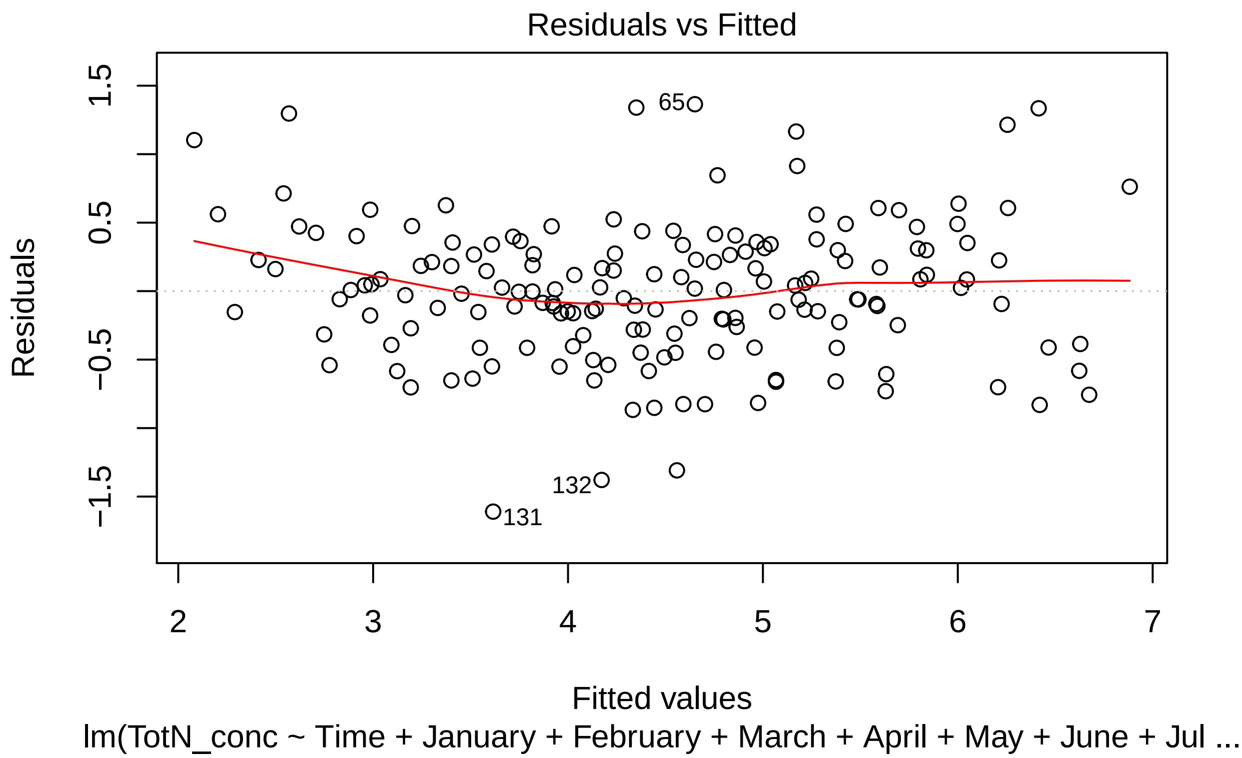
Answer: Looking of the ACF plots of the residuals we observe that their absolute value changes, but it does not seem like the series becomes stationary. Also looking at the residual pattern we see, that it actually gets smoothed to some extend, but the general trend seems to be unaffacted, also the seasonality does not disappear.

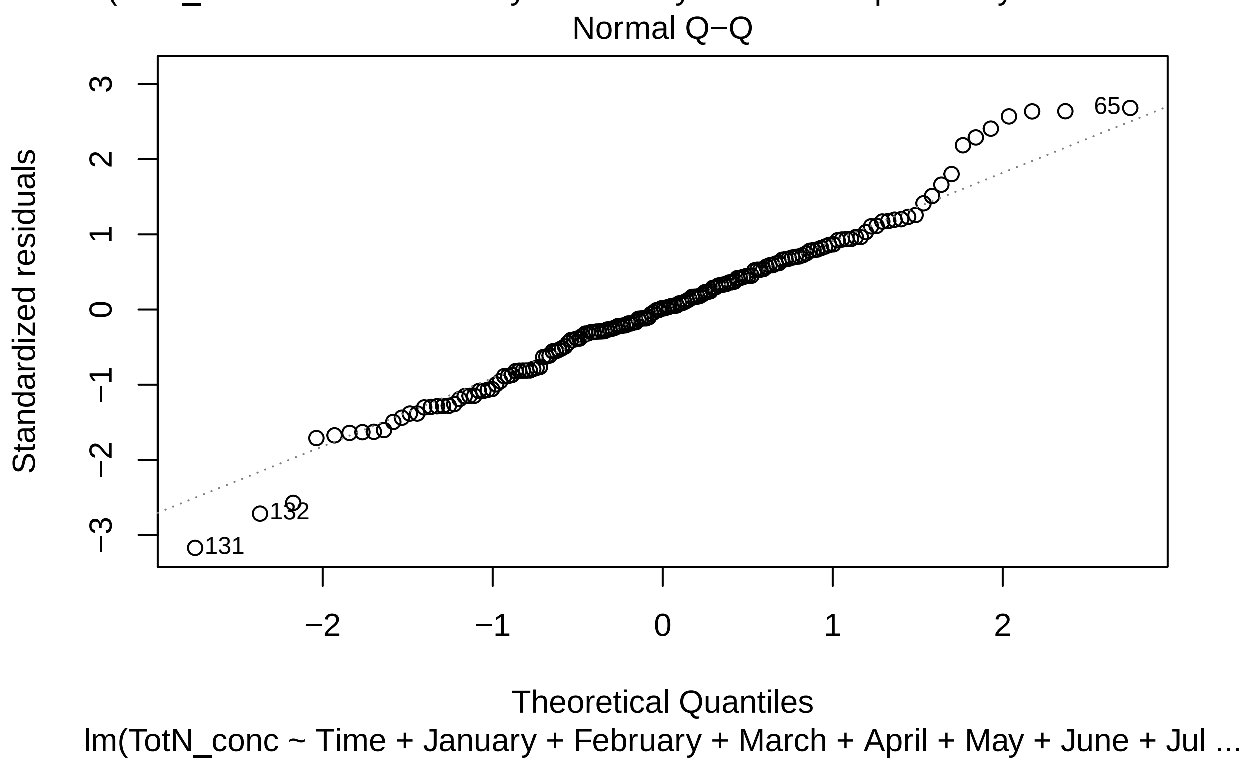
Task d): Eliminate the trend by fitting the following so-called seasonal means model:

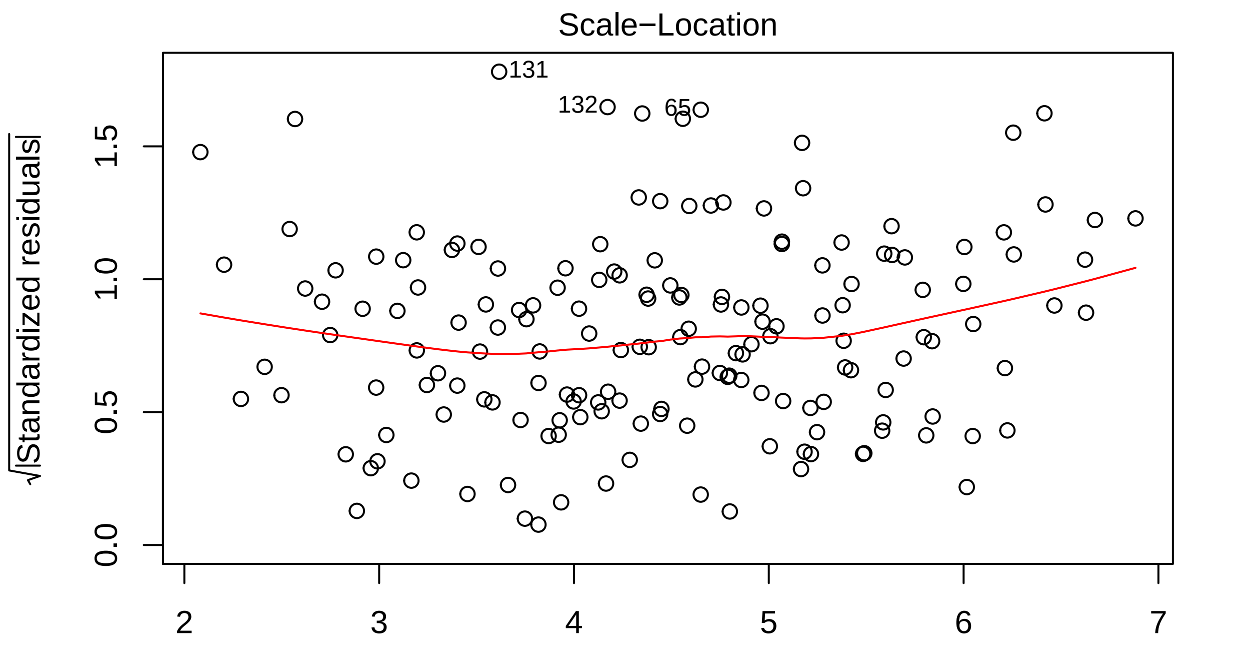
$$x_t = \alpha_0 + \alpha_1 t + \beta_1 I(\text{month} = 1) + ... + \beta_{12} I(\text{month} = 12) + w_t$$

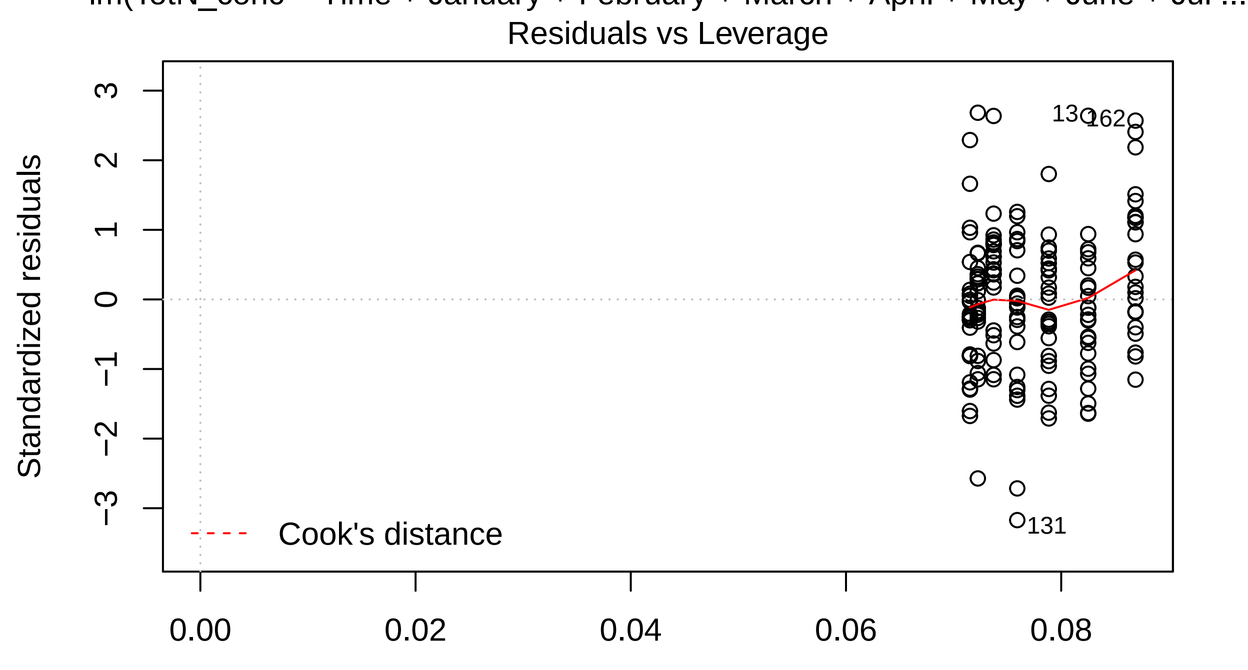
where $I(x) = 1$ if x is true and 0 otherwise. Fitting of this model will require you to augment data with a categorical variable showing the current month, and then fitting a usual linear regression. Analyze the residual pattern and the ACF of residuals.

```
rhine_onehot = rhine

# Could be easier handled using as.factor(rhine$Month) in the formula,
# but then the columns don't have names and the "new" dataframe is not saved,
# so we will stick with this.

rhine_onehot = rhine_onehot %>%
  mutate(January = if_else(Month == 1, TRUE, FALSE)) %>%
  mutate(February = if_else(Month == 2, TRUE, FALSE)) %>%
  mutate(March = if_else(Month == 3, TRUE, FALSE)) %>%
  mutate(April = if_else(Month == 4, TRUE, FALSE)) %>%
  mutate(May = if_else(Month == 5, TRUE, FALSE)) %>%
  mutate(June = if_else(Month == 6, TRUE, FALSE)) %>%
  mutate(July = if_else(Month == 7, TRUE, FALSE)) %>%
  mutate(August = if_else(Month == 8, TRUE, FALSE)) %>%
  mutate(September = if_else(Month == 9, TRUE, FALSE)) %>%
  mutate(October = if_else(Month == 10, TRUE, FALSE)) %>%
  mutate(November = if_else(Month == 11, TRUE, FALSE)) %>%
  mutate(December = if_else(Month == 11, TRUE, FALSE))

seasonal_model = lm(formula = TotN_conc ~ Time + January + February + March + April +
                               May + June + July + August +
                               September + October + November + December,
                    data = rhine_onehot)

detrended2 = rhine_time_series - seasonal_model$fitted.values

#plot(detrended2)
plot(seasonal_model)
```

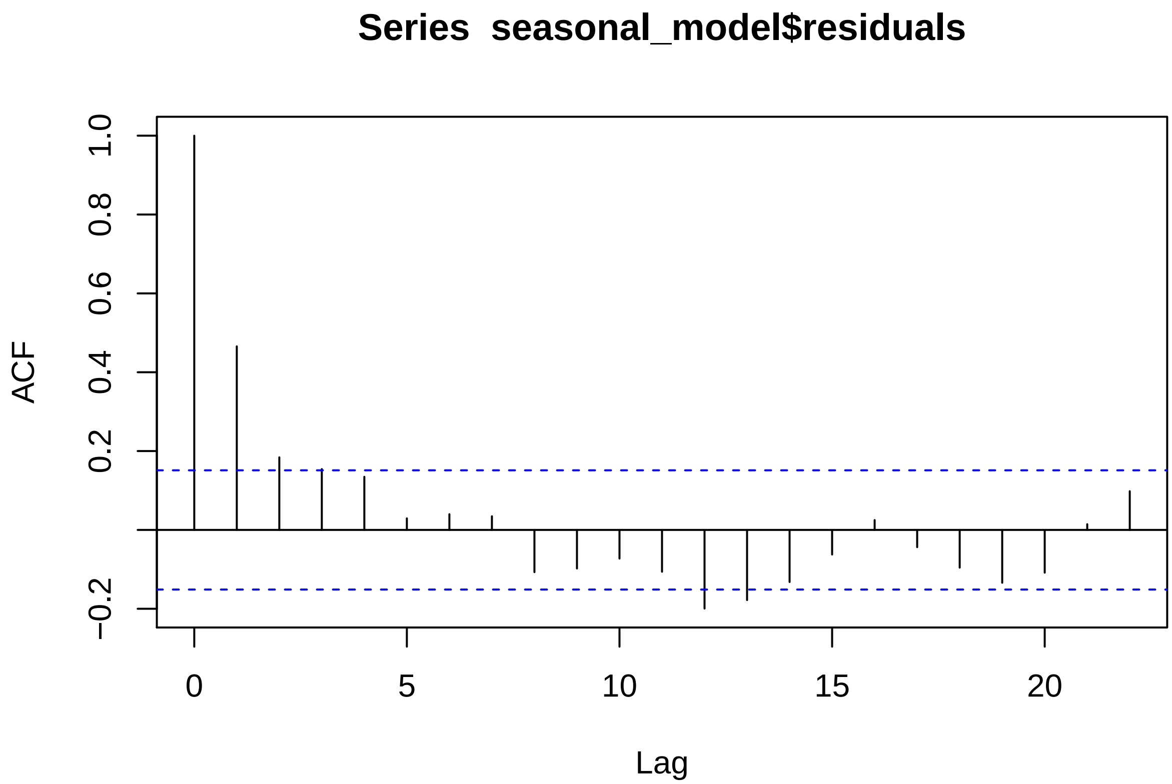

Residuals vs Fitted
Residuals
1.5
0.5
−0.5
−1.5
65
132
131
2
3
4
5
6
7
Fitted values
lm(TotN_conc ~ Time + January + February + March + April + May + June + Jul ...


Normal Q−Q
Standardized residuals
3
2
1
0
−1
−2
−3
65
132
131
−2
−1
0
1
2
Theoretical Quantiles
lm(TotN_conc ~ Time + January + February + March + April + May + June + Jul ...

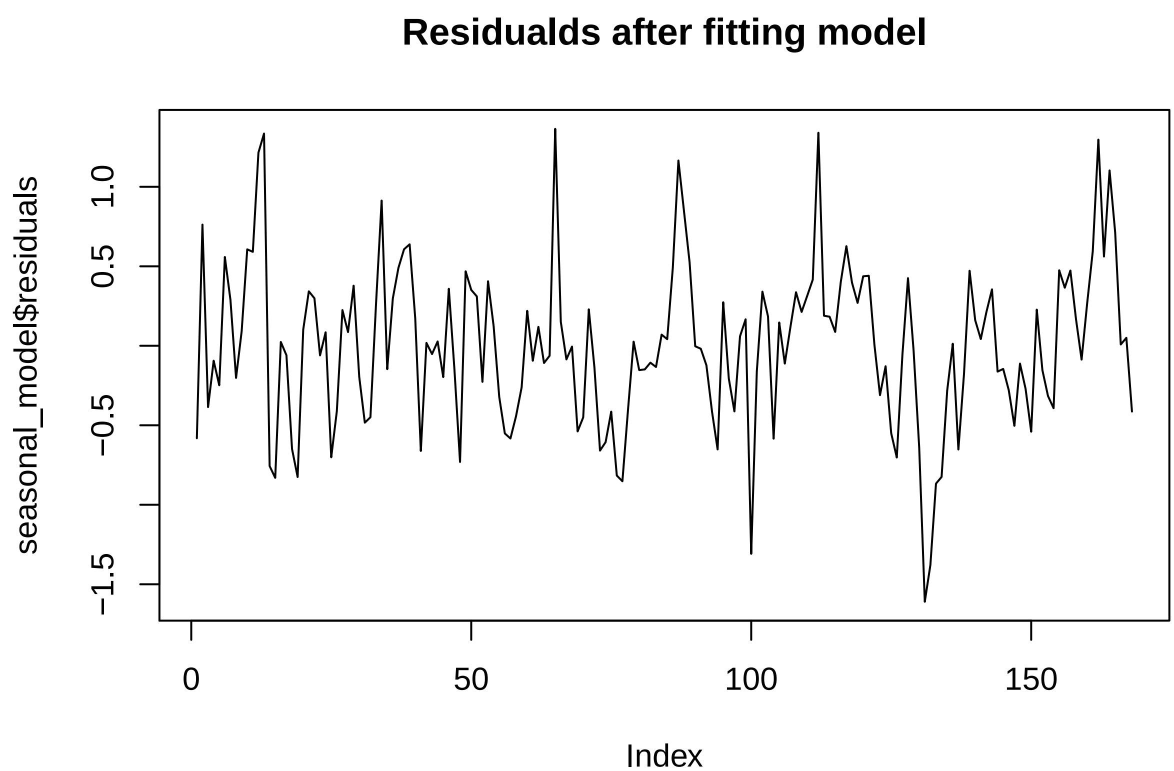





```
acf(seasonal_model$residuals)
```




```r
plot(seasonal_model$residuals, type='l', main = "Residualds after fitting model")
```



Answer: We can clearly see that the trend is gone. Also it looks like that after detrending with this model, the remaining data/model seems to become stationary. Still the errors seems to contain some information as it is not normal and still shows the seasonality. Also we have one component let for low lag.

```r
set.seed(12345)

aic_selection = stepAIC(seasonal_model, scope = list(upper = ~ .,
                                    lower = ~ 1),
                trace = TRUE,
                direction="backward")
```

```
## Start:  AIC=-202.02
## TotN_conc ~ Time + January + February + March + April + May +
##     June + July + August + September + October + November + December
##
##
## Step:  AIC=-202.02
## TotN_conc ~ Time + January + February + March + April + May +
##     June + July + August + September + October + November
##
##             Df Sum of Sq     RSS      AIC
## - April      1     0.200  43.436 -203.249
## - January    1     0.220  43.456 -203.170
## - March      1     0.331  43.567 -202.743
## <none>                    43.237 -202.023
## - February   1     1.440  44.677 -198.517
## - November   1     2.305  45.541 -195.297
## - May        1     3.274  46.511 -191.760
## - October    1     3.401  46.637 -191.303
## - September  1     7.853  51.089 -175.986
## - June       1     8.215  51.452 -174.797
## - July       1    14.321  57.557 -155.959
## - August     1    16.488  59.725 -149.749
## - Time       1   118.387 161.624   17.499
##
## Step:  AIC=-203.25
## TotN_conc ~ Time + January + February + March + May + June +
##     July + August + September + October + November
##
##             Df Sum of Sq     RSS      AIC
## <none>                    43.436 -203.249
## - January    1     0.640  44.077 -202.790
## - March      1     0.851  44.288 -201.988
## - November   1     2.235  45.671 -196.819
## - February   1     2.706  46.142 -195.096
## - May        1     3.355  46.791 -192.748
## - October    1     3.502  46.938 -192.223
## - September  1     8.868  52.304 -174.036
## - June       1     9.317  52.753 -172.602
## - July       1    16.912  60.348 -150.004
## - August     1    19.636  63.072 -142.586
## - Time       1   118.194 161.630   15.506
```

The following features are left in the model:

```r
colnames(aic_selection$model)
```

```
##  [1] "TotN_conc" "Time"      "January"   "February"  "March"
##  [6] "May"       "June"      "July"      "August"    "September"
```

```
## [11] "October"   "November"
```

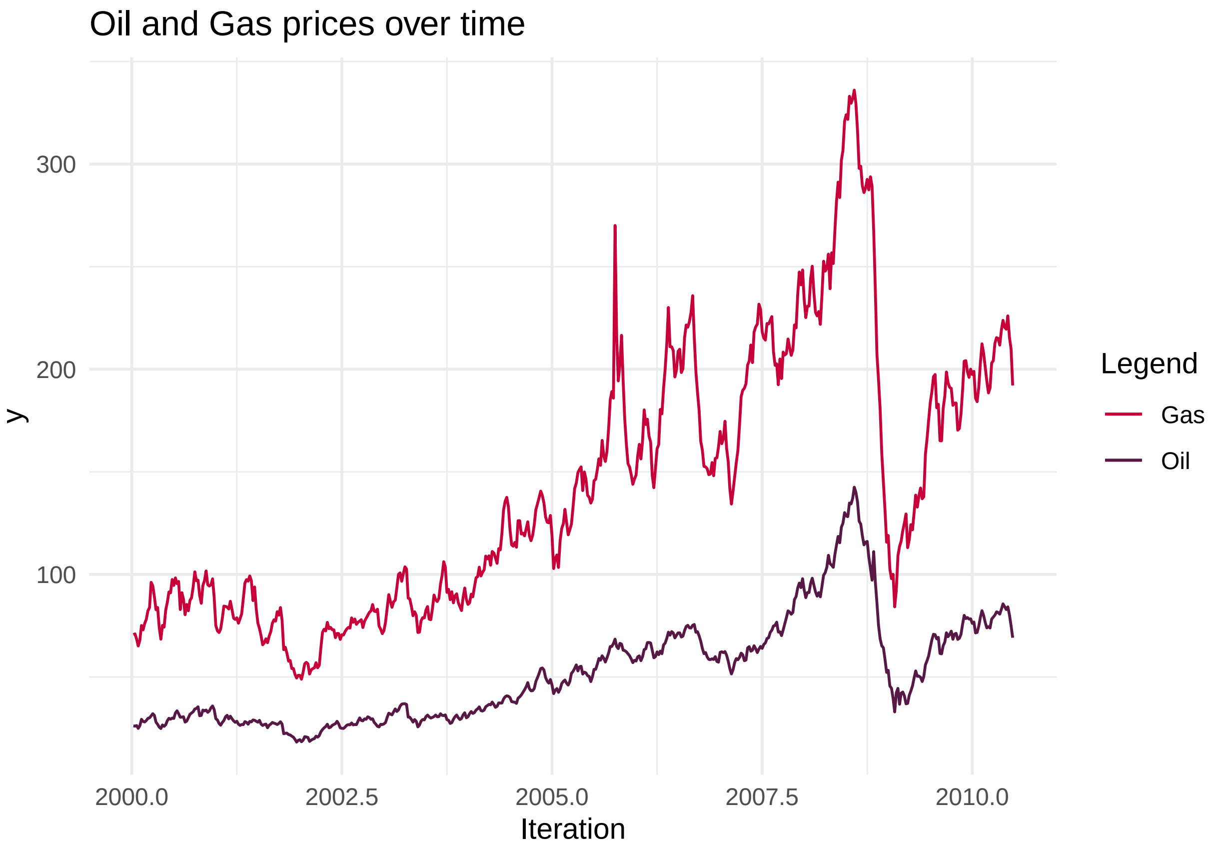
3 Analysis of oil and gas time series

Weekly time series *oil* and *gas* present in the package **astsa** show the oil prices in dollars per barrel and gas prices in cents per dollar.

```
oil_data = astsa::oil
gas_data = astsa::gas

oil_data_ts = ts(oil_data)
gas_data_ts = ts(gas_data)
```

Task a): Plot the given time series in the same graph. Do they look like stationary series? Do the processes seem to be related to each other? Motivate your answer.

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

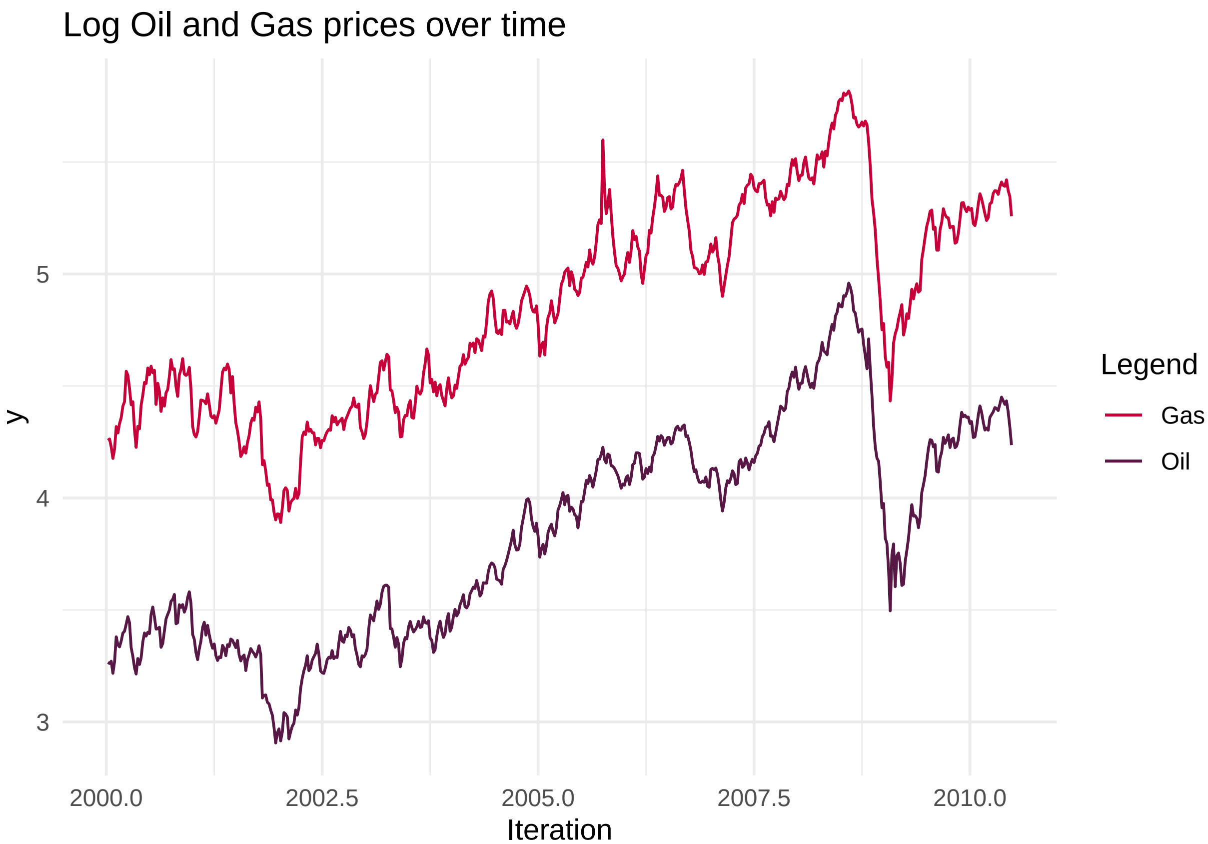


Answer: Both series do not seem stationary as they have an increase in variance and a positive linear trend. Also both series seem to be related, as both have a price drop around year 2008/2009.

Task b): Apply log-transform to the time series and plot the transformed data. In what respect did this transformation made the data easier for the analysis?

```
df$oil_log = log(df$oil)
df$gas_log = log(df$gas)
```

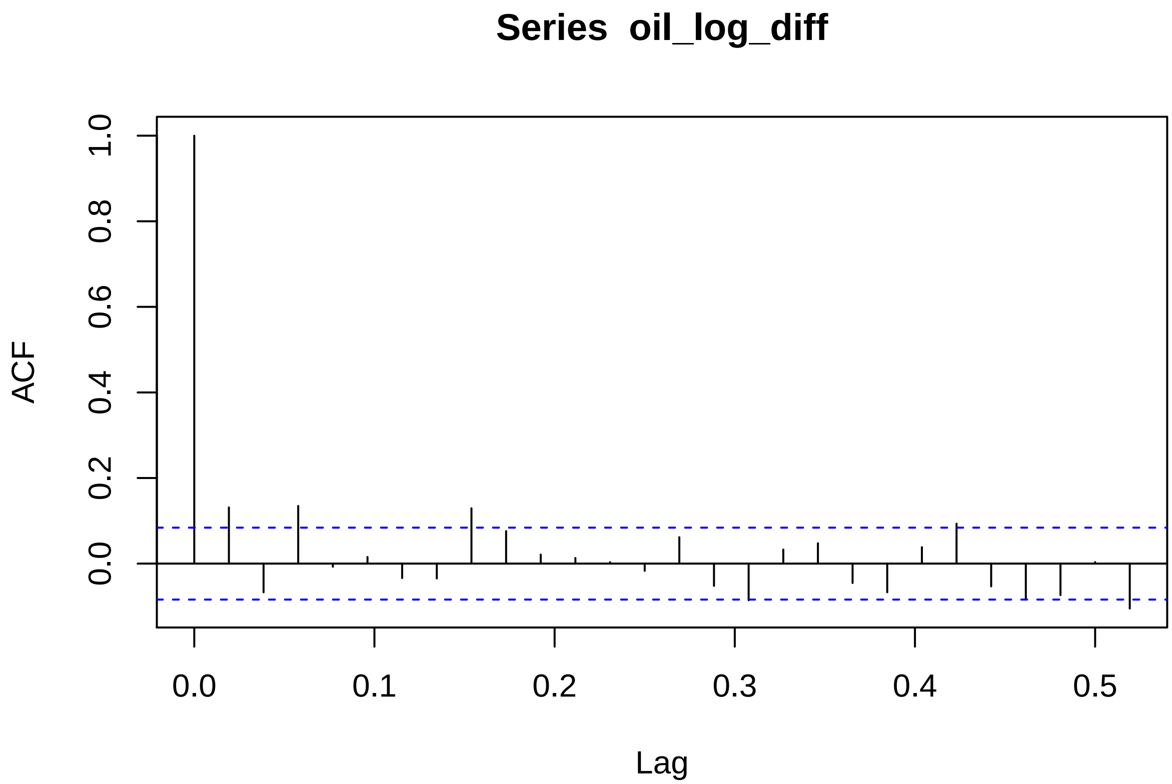
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



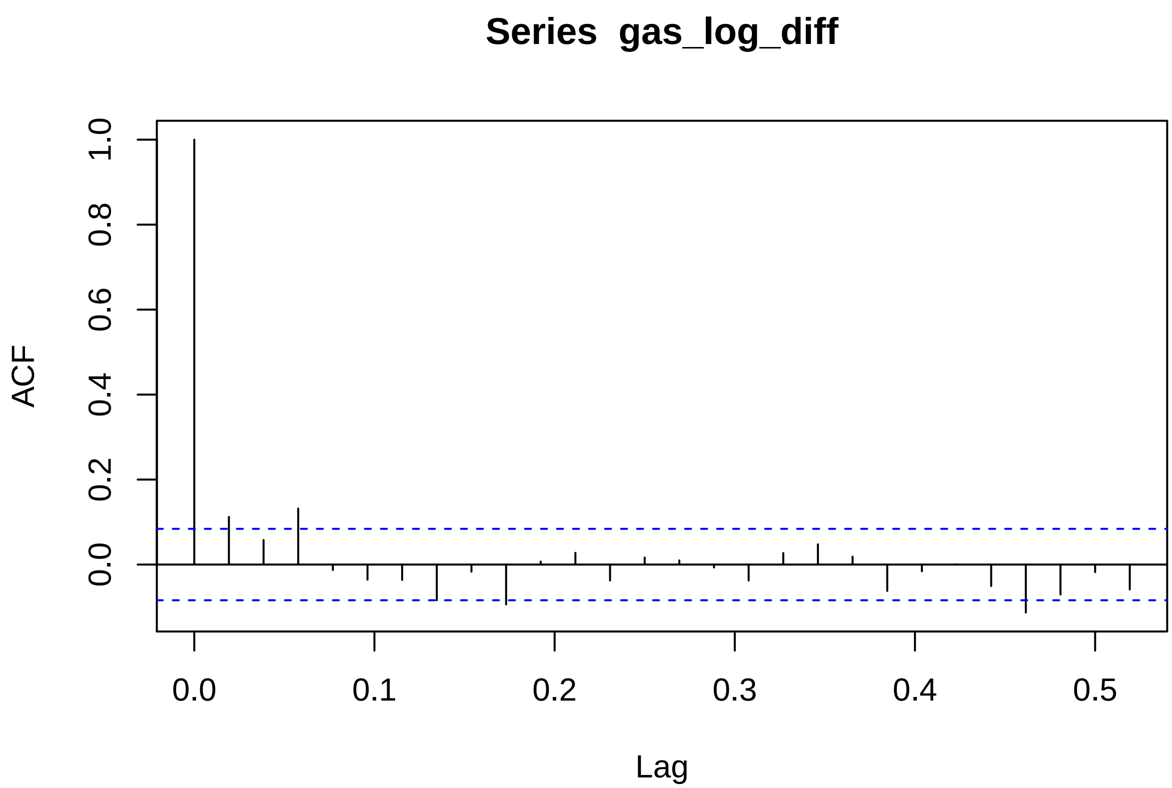
Answer: The log operation reduces the *amplitude* of the variance, thus making it easier to analize the two time series.

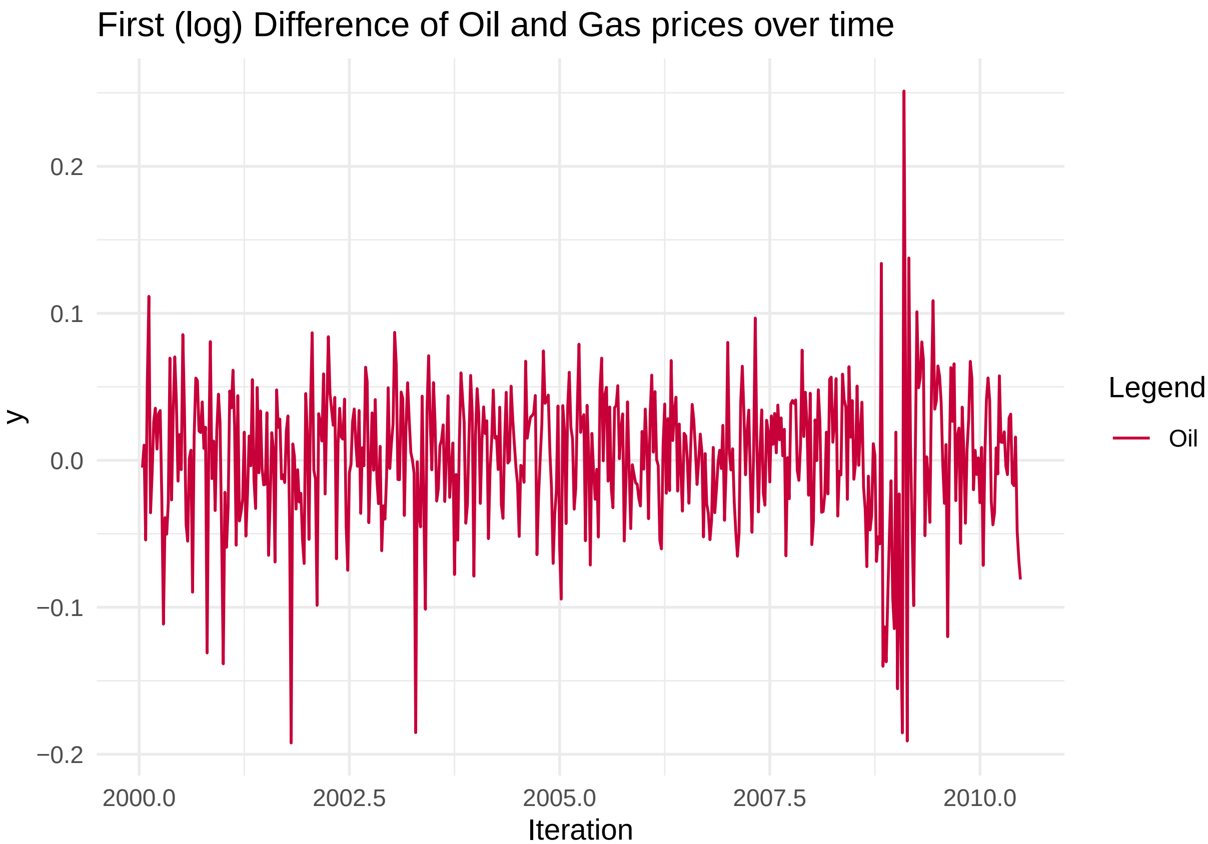
Task c): To eliminate trend, compute the first difference of the transformed data, plot the detrended series, check their ACFs and analyze the obtained plots. Denote the data obtained here as x_t (oil) and y_t (gas).

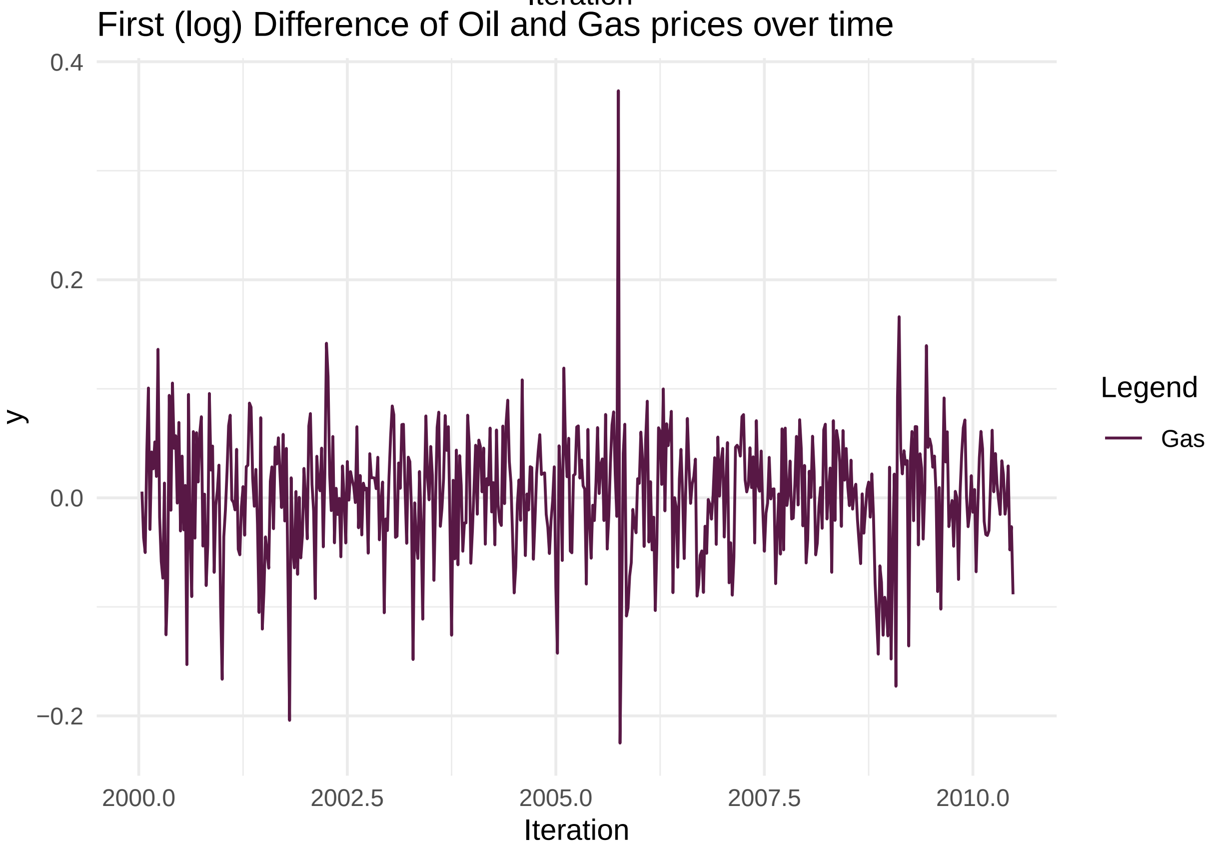
```
oil_log_diff = diff(df$oil_log, differences = 1)
gas_log_diff = diff(df$gas_log, differences = 1)

#oil_log_diff = diff(df$oil, differences = 1)
#gas_log_diff = diff(df$gas, differences = 1)

df$x_t = c(NA, oil_log_diff) #oil
df$y_t = c(NA, gas_log_diff) #gas

acf(oil_log_diff)
```




```
acf(gas_log_diff)
```



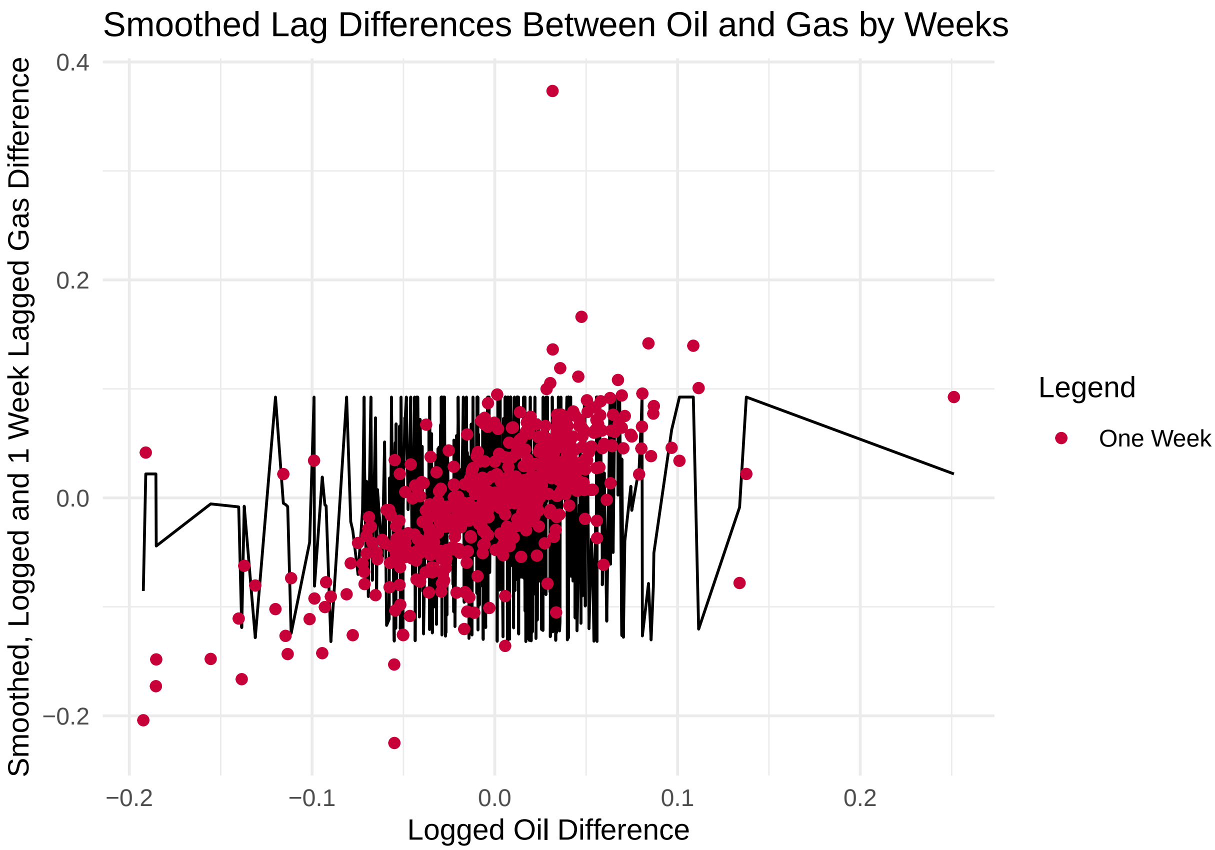


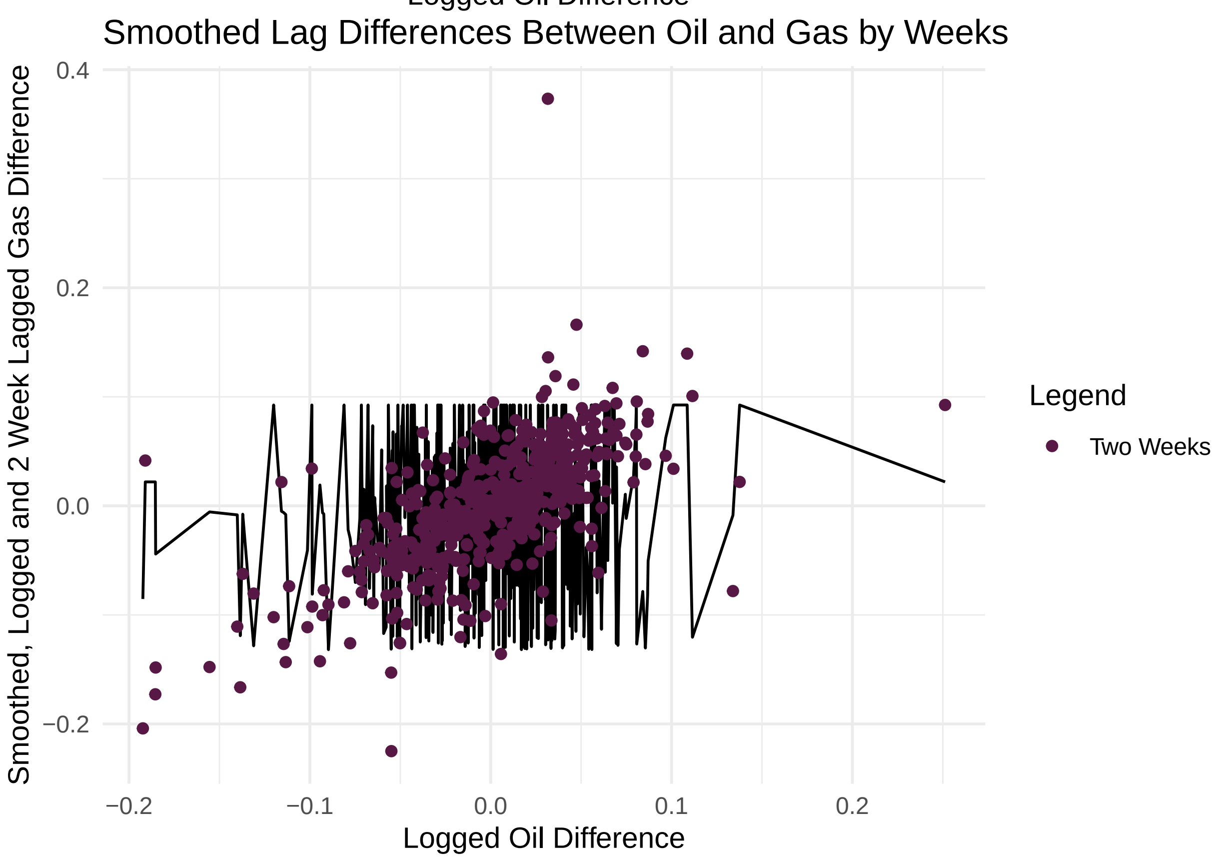


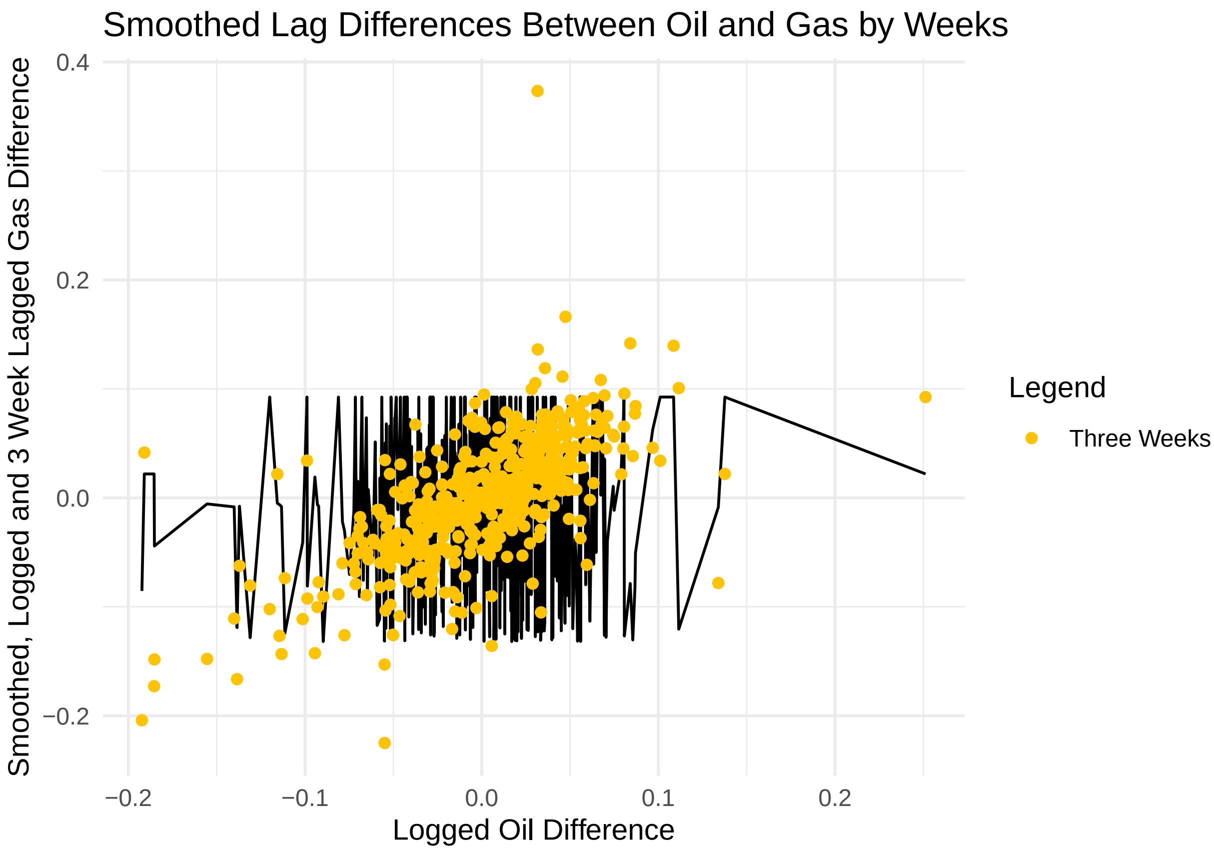
Answer: Now it seems like both series are stationary and also their variance seems (still) to be constant.

Task d): Exhibit scatterplots of x_t and y_t for up to three weeks of lead time of x_t; include a nonparametric smoother in each plot and comment the results: are there outliers? Are the relationships linear? Are there changes in the trend?

```r
#oil_log_diff
#gas_log_diff

df = na.omit(df)

# Creating lags
#df$oil_log_lag_1 = stats::lag(oil_log_diff, 1)
#df$oil_log_lag_2 = stats::lag(oil_log_diff, 2)
#df$oil_log_lag_3 = stats::lag(oil_log_diff, 3)
df$yt_lag_1 = stats::lag(df$y_t, 1)
df$yt_lag_2 = stats::lag(df$y_t, 2)
df$yt_lag_3 = stats::lag(df$y_t, 3)

# Smoothing
df$smooth_one_week = ksmooth(x = df$x_t,
                             y = df$yt_lag_1,
                             bandwidth = 2/52,
                             kernel = "normal")$y

df$smooth_two_week = ksmooth(x = df$x_t,
                             y = df$yt_lag_2,
                             bandwidth = 2/52,
                             kernel = "normal")$y

df$smooth_three_week = ksmooth(x = df$x_t,
                               y = df$yt_lag_3,
                               bandwidth = 2/52,
                               kernel = "normal")$y
```


Smoothed Lag Differences Between Oil and Gas by Weeks
Smoothed, Logged and 1 Week Lagged Gas Difference
Logged Oil Difference
0.4
0.2
0.0
−0.2
−0.2
−0.1
0.0
0.1
0.2
Legend
One Week


Smoothed Lag Differences Between Oil and Gas by Weeks
Smoothed, Logged and 2 Week Lagged Gas Difference
Logged Oil Difference
0.4
0.2
0.0
−0.2
−0.2
−0.1
0.0
0.1
0.2
Legend
Two Weeks



Answer: We have outliers in all three versions, so they don't disappear. The relaionship remains linear for all versions as well. We do not observe any change in the trend.

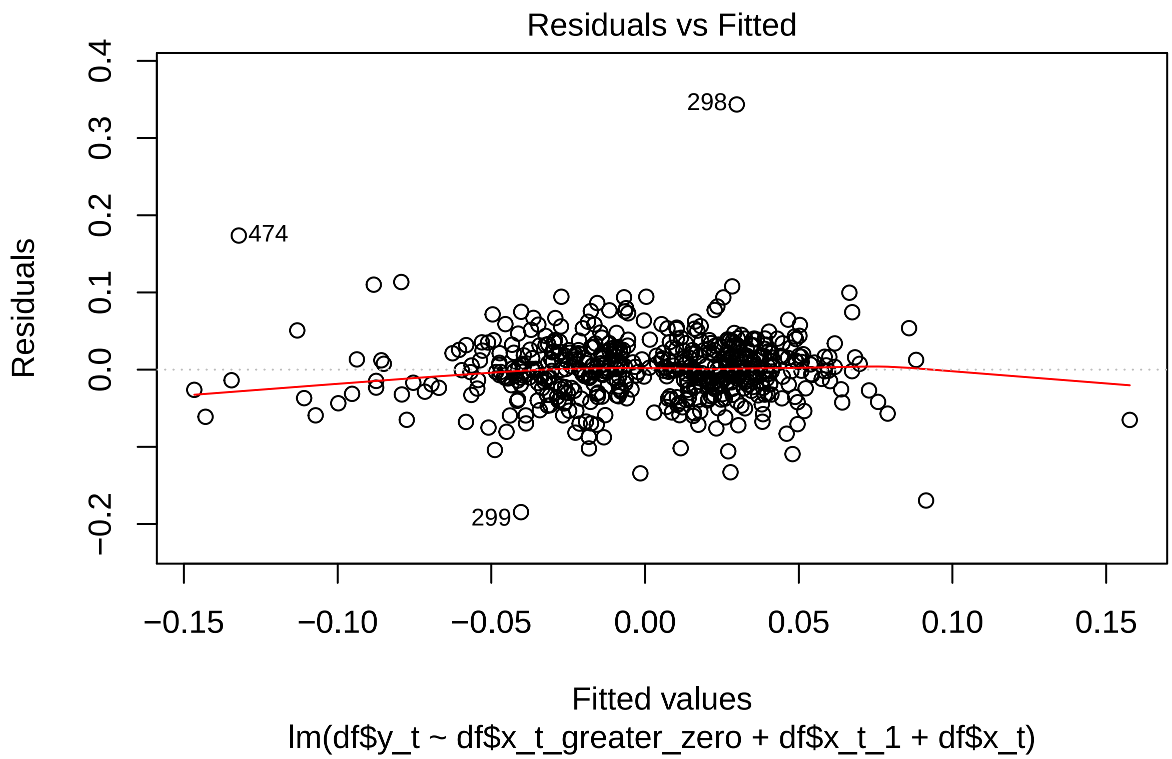
Task e): Fit the following model: $y_t = \alpha_0 + \alpha_1 I(x_t > 0) + \beta_1 x_t + \beta_2 x_{t-1} + w_t$ and check which coefficients seem to be significant. How can this be interpreted? Analyze the residual pattern and the ACF of the residuals.

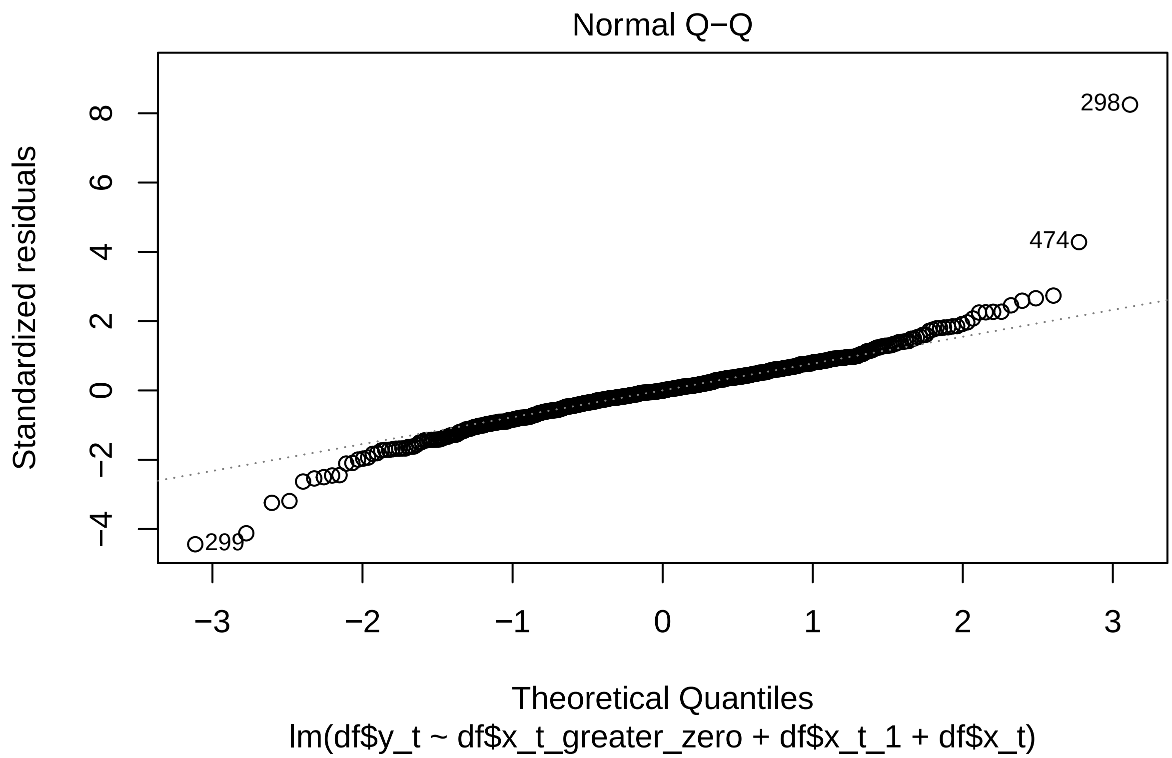
```r
# Greating the Identity
df$x_t_greater_zero = ifelse(df$x_t>0, TRUE, FALSE)

# Creating x_{t-1}
df$x_t_1 = c(NA, df$x_t[1:(length(df$x_t))-1])

model_e = lm(formula = df$y_t ~ df$x_t_greater_zero + df$x_t_1 + df$x_t)

summary(model_e)
```

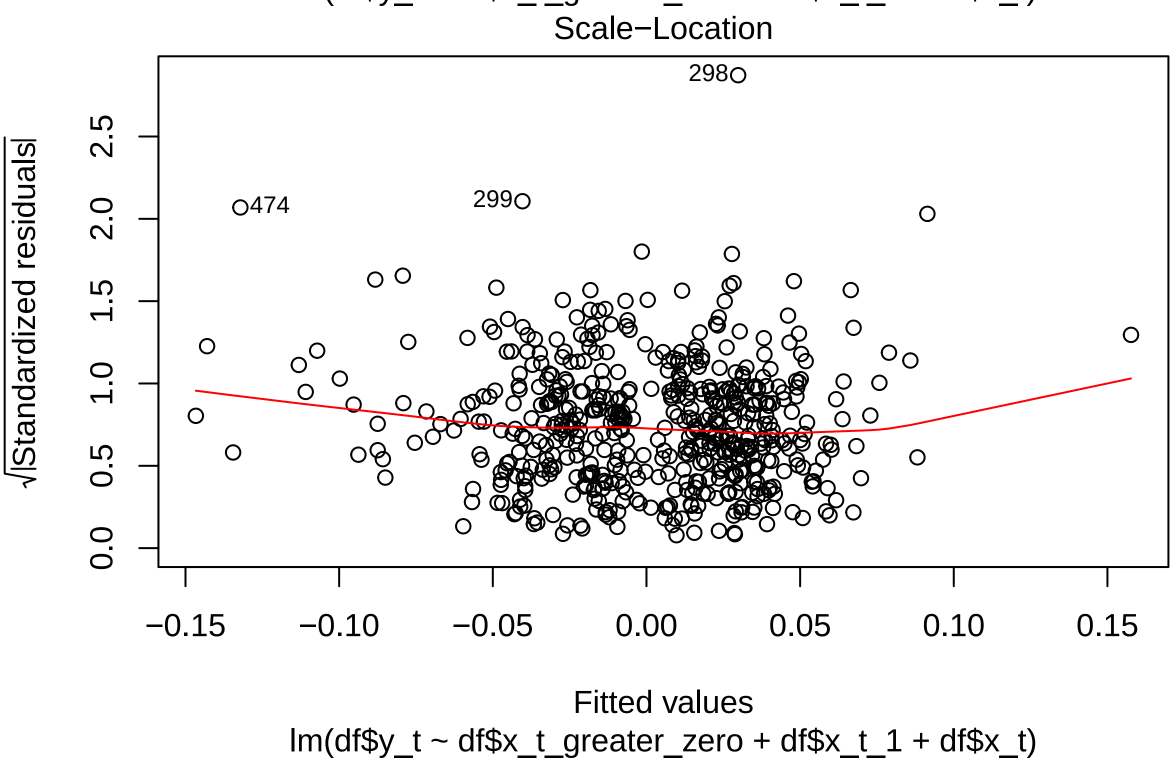
```
##
## Call:
## lm(formula = df$y_t ~ df$x_t_greater_zero + df$x_t_1 + df$x_t)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.18460 -0.02167 -0.00030  0.02176  0.34352
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -0.006110   0.003455  -1.768  0.07759 .
```

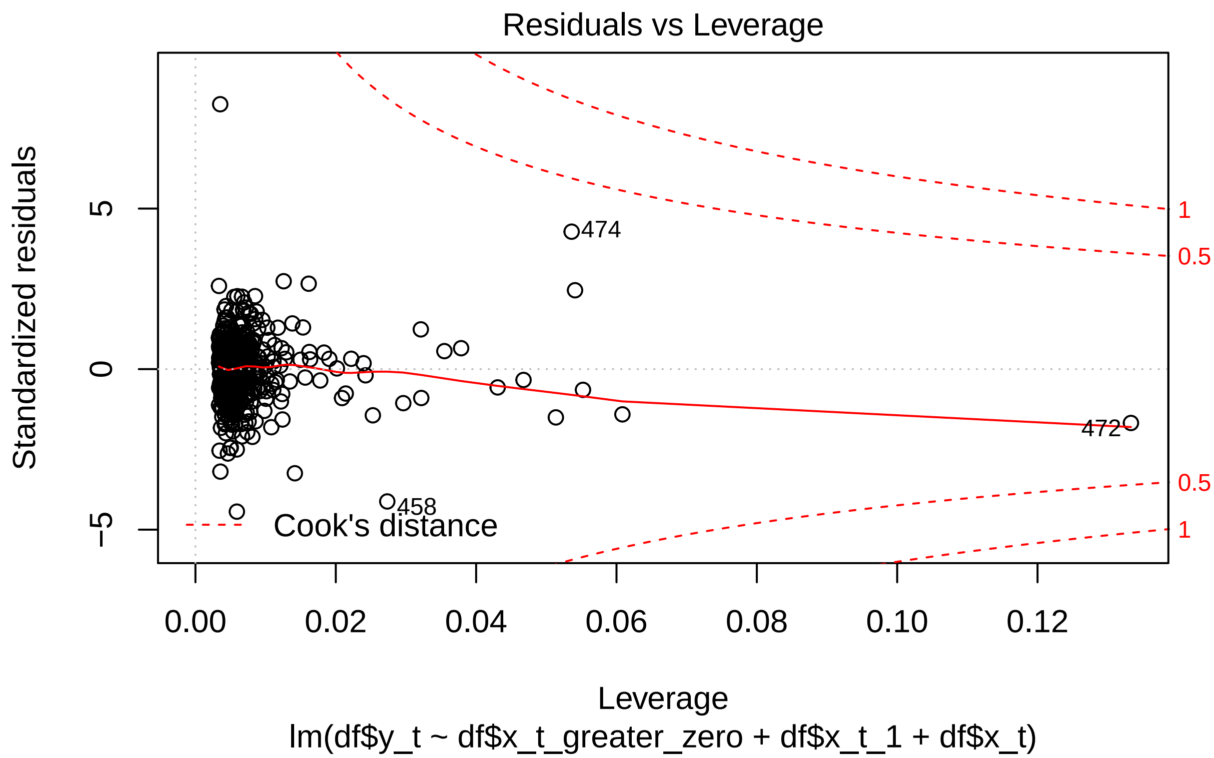
```
## df$x_t_greater_zeroTRUE  0.011785   0.005514   2.137  0.03303 *
## df$x_t_1                 0.112152   0.038570   2.908  0.00379 **
## df$x_t                   0.687749   0.058380  11.781  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.04171 on 539 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.4558, Adjusted R-squared:  0.4528
## F-statistic: 150.5 on 3 and 539 DF,  p-value: < 2.2e-16
```

```r
plot(model_e)
```

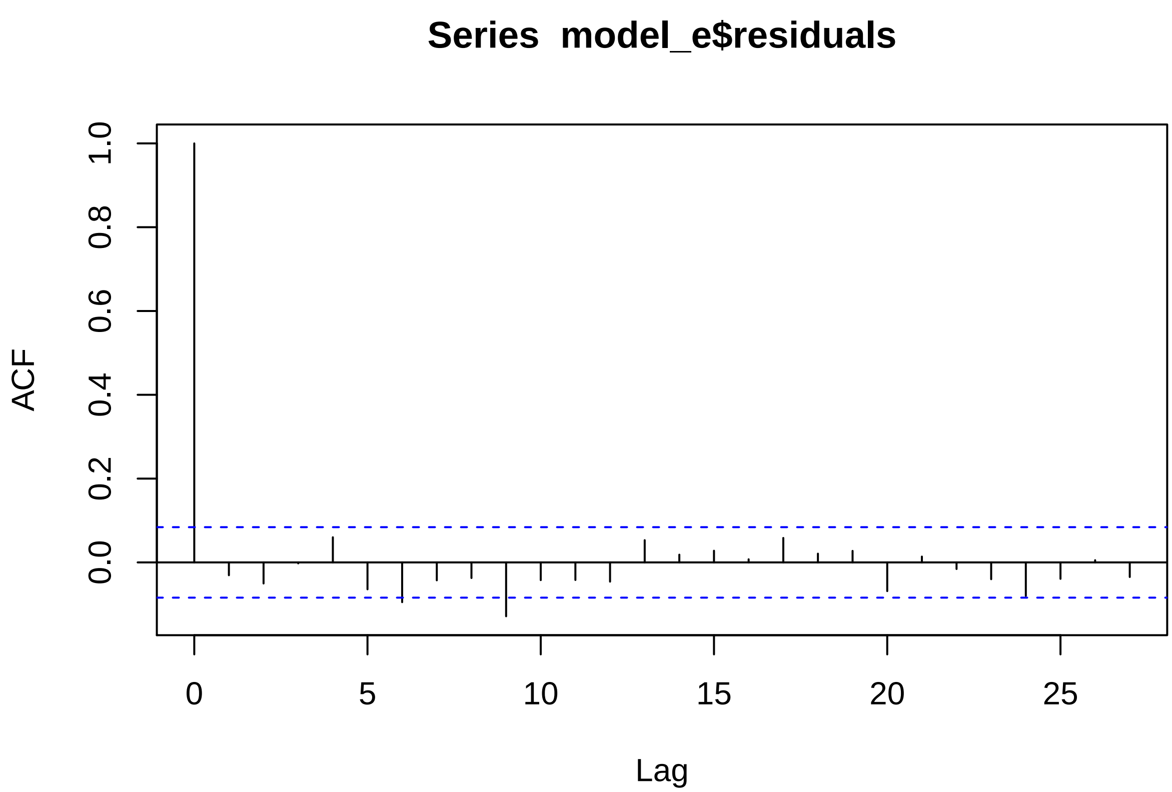



Normal Q−Q
298
474
299
Standardized residuals
−4
−2
0
2
4
6
8
−3
−2
−1
0
1
2
3
Theoretical Quantiles
lm(df$y_t ~ df$x_t_greater_zero + df$x_t_1 + df$x_t)

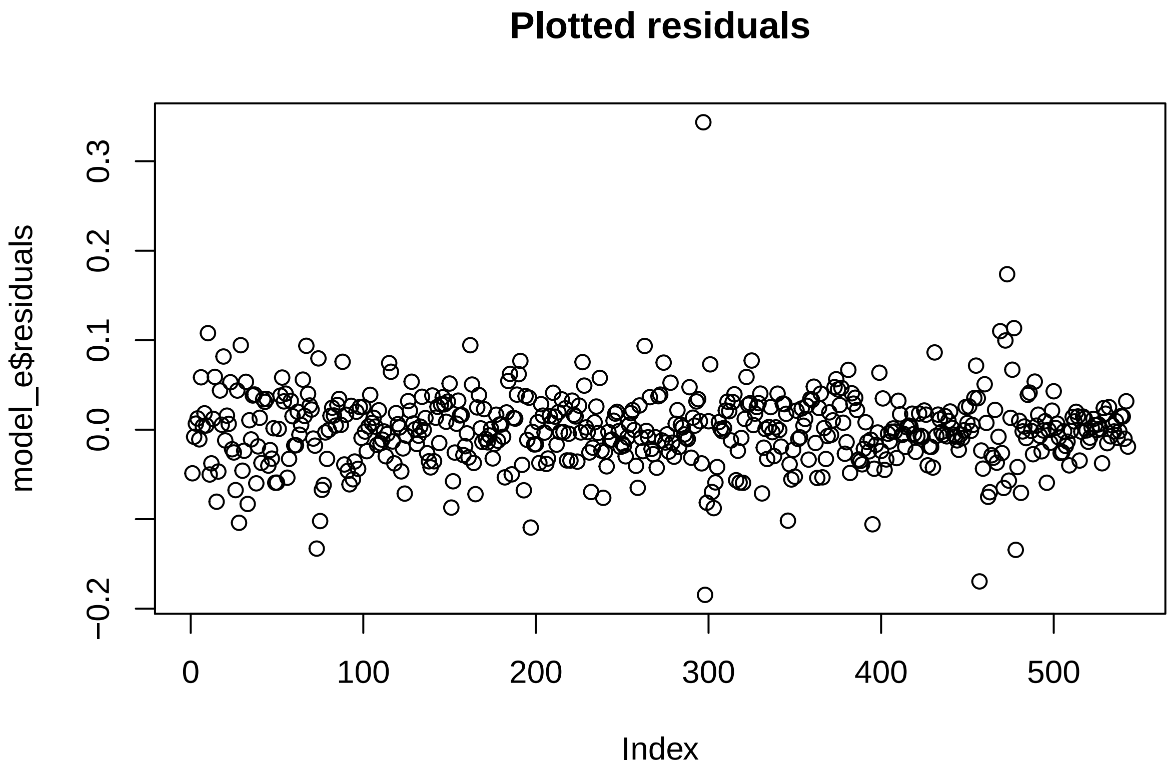

Scale−Location
298
474
299
√|Standardized residuals|
0.0
0.5
1.0
1.5
2.0
2.5
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
Fitted values
lm(df$y_t ~ df$x_t_greater_zero + df$x_t_1 + df$x_t)




```
acf(model_e$residuals)
```




```
plot(model_e$residuals, main = "Plotted residuals")
```



Answer: It seems like the feature x_t is significant. The residuals look like white noise (with a specific σ), which means that we explained everything in the data that is predictable. Looking at the ACF also confirms this and shows that the error seems to be stationary.

4 Source Code

```
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(MASS)
library(astsa)
set.seed(12345)

################################################################################
# Exercise 1.a)
################################################################################

x0 = 0
x1 = 0

n = 100

# Series 1
generate_S1 = function(t, x0=0, x1=1) {

  series = vector(length = t)
  series[1] = x0
  series[2] = x1
```

```
  for (i in 3:t) {
    series[i] = -0.8 * series[i-2] + rnorm(n=1, mean=0, sd=1)
  }

  return(ts(series))
}

# Series 2
generate_S2 = function(t) {
  series = vector(length = t)

  for (i in 1:t) {
    series[i] = cos(2 * pi * i / 5)
  }

  return(ts(series))
}

index = c(1:n)

series1 = generate_S1(n)
series2 = generate_S2(n)

series1_filtered = stats::filter(series1, filter = rep(0.2, 5), sides = 1)
series2_filtered = stats::filter(series2, filter = rep(0.2, 5), sides = 1)


df = data.frame(index,
                series1 = as.numeric(series1),
                series2 = as.numeric(series2),
                series1_filtered = as.numeric(series1_filtered),
                series2_filtered = as.numeric(series2_filtered))

ggplot(df) +
  geom_line(aes(x = index, y = series1), color = "#C70039") +
  geom_line(aes(x = index, y = series1_filtered), color = "#000000") +
  labs(title = "Time Series 1 with Smoothing Filter", y = "y",
  x = "Iteration", color = "Legend") +
  theme_minimal()

ggplot(df) +
  geom_line(aes(x = index, y = series2), color = "#6091EC") +
  geom_line(aes(x = index, y = series2_filtered), color = "#000000") +
  labs(title = "Time Series 2 with Smoothing Filter", y = "y",
  x = "Iteration", color = "Legend") +
  theme_minimal()


################################################################################
# Exercise 1.b)
################################################################################

generate_S3 = function(t, X, W) {
```

```
  series = vector(length=t)
  white_noise = vector(length=t)

  series[1:length(X)] = X
  white_noise[1:length(W)] = W

  for (i in 7:t) {
    W[1:6] = W[2:7]
    W[7] = rnorm(1, mean=0, sd=1)
    series[i] = 4 * series[i-1] - 2 * series[i-2] - series[i-5] +
                  W[7] + 3 * W[5] + W[2] - 4 * W[1]
  }

  return(ts(series))
}

series3 = generate_S3(t=n, X = rnorm(7, mean=0, sd=1), W = rnorm(7, mean=0, sd=1))


Z_phi = c(1, -4, 2, 0, 0, 1)

isOutsideUnitCircle = function(Z) {
  return(all(Mod(polyroot(Z)) > 1))
}

isOutsideUnitCircle(Z_phi)
polyroot(Z_phi)


Z_theta = c(1, 0, 3, 0, -1, 0, 4)

isOutsideUnitCircle(Z_theta)
polyroot(Z_theta)


set.seed(54321)

model = list(ar = c(-3/4), ma = c(0, -1/9))
series = arima.sim(model = model, n = 100)

# Sample
auto_correlations_sample = acf(series)

# Theoretical
auto_correlations_theoretical = ARMAacf(ar = model$ar, ma = model$ma,
                                        lag.max = 20)
acf(auto_correlations_theoretical)


rhine = read_csv2("Rhine.csv")
head(rhine)
```

```r
rhine_time_series = ts(data = rhine$TotN_conc, start = c(1989,1),
                       frequency = 12)

# Normal Time Series
plot(rhine_time_series, main = "Time Series of Nitrogen Concentration in Rhine")

# 12 Lags as we have 12 month each year
lag.plot(rhine_time_series, lags = 12)

# Autocovariance
acf(rhine_time_series, lag.max = nrow(rhine))


# Linear Model
rhine_linear_model = lm(TotN_conc ~ Time, data=rhine)

summary(rhine_linear_model)

# Difference
detrended = rhine_time_series - rhine_linear_model$fitted.values
plot(detrended)

#plot(rhine_linear_model)
plot(rhine_linear_model$residuals)
acf(rhine_linear_model$residuals)

# Could also be decomposed by using this
#rhine_decomposed_additive = decompose(rhine_time_series, "additive")
#rhine_decomposed_multiplicative = decompose(rhine_time_series, "multiplicative")

# STL() would also be possible


rhine_time_series_smoothed_5 = ksmooth(x = rhine$Time,
                                       y = rhine$TotN_conc,
                                       bandwidth=5)

rhine_time_series_smoothed_10 = ksmooth(x = rhine$Time,
                                       y = rhine$TotN_conc,
                                       bandwidth=10)

rhine_time_series_smoothed_20 = ksmooth(x = rhine$Time,
                                       y = rhine$TotN_conc,
                                       bandwidth=20)

residual_k_5 = rhine_time_series - rhine_time_series_smoothed_5$y
residual_k_10 = rhine_time_series - rhine_time_series_smoothed_10$y
residual_k_20 = rhine_time_series - rhine_time_series_smoothed_20$y


df = data.frame(x = rhine_time_series_smoothed_5$x,
                s5 = rhine_time_series_smoothed_5$y,
                s10 = rhine_time_series_smoothed_10$y,
```

```
                 s20 = rhine_time_series_smoothed_20$y,
                 rhine$TotN_conc)

ggplot(df) +
  geom_line(aes(x = x, y = s5, colour = "Bandwith = 5")) +
  geom_line(aes(x = x, y = s10, colour = "Bandwith = 10")) +
  geom_line(aes(x = x, y = s20, colour = "Bandwith = 20")) +
  geom_line(aes(x = x, y = rhine.TotN_conc, colour = "Original Data")) +
  labs(title = "Different Smoothing Filters",
       y = "Monthly concentrations of total nitrogen",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("#C70039", "#FF5733", "#581845", "#000000")) +
  theme_minimal()


df = data.frame(x = rhine$Time,
                 k5 = residual_k_5,
                 k10 = residual_k_10,
                 k20 = residual_k_20,
                 rhine$TotN_conc)

ggplot(df) +
  geom_line(aes(x = x, y = k5, colour = "Residual with Bandwith = 5")) +
  geom_line(aes(x = x, y = k10, colour = "Residual with Bandwith = 10")) +
  geom_line(aes(x = x, y = k20, colour = "Residual with Bandwith = 20")) +
  geom_line(aes(x = x, y = rhine.TotN_conc, colour = "Original Data")) +
  labs(title = "Different Smoothing Filters",
       y = "Monthly concentrations of total nitrogen",
       x = "Time", color = "Legend") +
  scale_color_manual(values = c("#C70039", "#FF5733", "#581845", "#000000")) +
  theme_minimal()


acf(residual_k_5)
acf(residual_k_10)
acf(residual_k_20)


rhine_onehot = rhine

# Could be easier handled using as.factor(rhine$Month) in the formula,
# but then the columns don't have names and the "new" dataframe is not saved,
# so we will stick with this.

rhine_onehot = rhine_onehot %>%
  mutate(January = if_else(Month == 1, TRUE, FALSE)) %>%
  mutate(February = if_else(Month == 2, TRUE, FALSE)) %>%
  mutate(March = if_else(Month == 3, TRUE, FALSE)) %>%
  mutate(April = if_else(Month == 4, TRUE, FALSE)) %>%
  mutate(May = if_else(Month == 5, TRUE, FALSE)) %>%
  mutate(June = if_else(Month == 6, TRUE, FALSE)) %>%
  mutate(July = if_else(Month == 7, TRUE, FALSE)) %>%
  mutate(August = if_else(Month == 8, TRUE, FALSE)) %>%
```

```
  mutate(September = if_else(Month == 9, TRUE, FALSE)) %>%
  mutate(October = if_else(Month == 10, TRUE, FALSE)) %>%
  mutate(November = if_else(Month == 11, TRUE, FALSE)) %>%
  mutate(December = if_else(Month == 11, TRUE, FALSE))

seasonal_model = lm(formula = TotN_conc ~ Time + January + February + March + April +
                                May + June + July + August +
                                September + October + November + December,
                    data = rhine_onehot)

detrended2 = rhine_time_series - seasonal_model$fitted.values

#plot(detrended2)
plot(seasonal_model)

acf(seasonal_model$residuals)
plot(seasonal_model$residuals, type='l', main = "Residualds after fitting model")


set.seed(12345)

aic_selection = stepAIC(seasonal_model, scope = list(upper = ~ .,
                                    lower = ~ 1),
                trace = TRUE,
                direction="backward")


colnames(aic_selection$model)


oil_data = astsa::oil
gas_data = astsa::gas

oil_data_ts = ts(oil_data)
gas_data_ts = ts(gas_data)


df = data.frame(index = 1:length(oil_data),
                date = seq(from = start(oil)[1] + start(oil)[2]/52,
                           to = end(oil)[1] + end(oil)[2]/52,
                           by=1/52),
                oil = oil_data,
                gas = gas_data)

ggplot(df) +
  geom_line(aes(x = date, y = oil, colour = "Oil")) +
  geom_line(aes(x = date, y = gas, colour = "Gas")) +
  labs(title = "Oil and Gas prices over time", y = "y",
  x = "Iteration", color = "Legend") +
  scale_color_manual(values = c("#C70039", "#581845")) +
  theme_minimal()
```

```r
df$oil_log = log(df$oil)
df$gas_log = log(df$gas)


ggplot(df) +
  geom_line(aes(x = date, y = oil_log, colour = "Oil")) +
  geom_line(aes(x = date, y = gas_log, colour = "Gas")) +
  labs(title = "Log Oil and Gas prices over time", y = "y",
  x = "Iteration", color = "Legend") +
  scale_color_manual(values = c("#C70039", "#581845")) +
  theme_minimal()


oil_log_diff = diff(df$oil_log, differences = 1)
gas_log_diff = diff(df$gas_log, differences = 1)

#oil_log_diff = diff(df$oil, differences = 1)
#gas_log_diff = diff(df$gas, differences = 1)

df$x_t = c(NA, oil_log_diff) #oil
df$y_t = c(NA, gas_log_diff) #gas

acf(oil_log_diff)
acf(gas_log_diff)


ggplot(df) +
  geom_line(aes(x = date, y = x_t, colour = "Oil")) +
  labs(title = "First (log) Difference of Oil and Gas prices over time", y = "y",
  x = "Iteration", color = "Legend") +
  scale_color_manual(values = c("#C70039", "#581845")) +
  theme_minimal()

ggplot(df) +
  geom_line(aes(x = date, y = y_t, colour = "Gas")) +
  labs(title = "First (log) Difference of Oil and Gas prices over time", y = "y",
  x = "Iteration", color = "Legend") +
  scale_color_manual(values = c("#581845", "#C70039")) +
  theme_minimal()


#oil_log_diff
#gas_log_diff

df = na.omit(df)

# Creating lags
#df$oil_log_lag_1 = stats::lag(oil_log_diff, 1)
#df$oil_log_lag_2 = stats::lag(oil_log_diff, 2)
#df$oil_log_lag_3 = stats::lag(oil_log_diff, 3)
df$yt_lag_1 = stats::lag(df$y_t, 1)
df$yt_lag_2 = stats::lag(df$y_t, 2)
df$yt_lag_3 = stats::lag(df$y_t, 3)
```

```r
# Smoothing
df$smooth_one_week = ksmooth(x = df$x_t,
                             y = df$yt_lag_1,
                             bandwidth = 2/52,
                             kernel = "normal")$y

df$smooth_two_week = ksmooth(x = df$x_t,
                             y = df$yt_lag_2,
                             bandwidth = 2/52,
                             kernel = "normal")$y

df$smooth_three_week = ksmooth(x = df$x_t,
                               y = df$yt_lag_3,
                               bandwidth = 2/52,
                               kernel = "normal")$y


ggplot(df) +
  geom_line(aes(x = df$x_t, y = df$smooth_one_week)) +
  geom_point(aes(x = df$x_t, y = df$yt_lag_1, colour = "One Week")) +
  labs(title = "Smoothed Lag Differences Between Oil and Gas by Weeks",
       y = "Smoothed, Logged and 1 Week Lagged Gas Difference",
       x = "Logged Oil Difference", color = "Legend") +
  scale_color_manual(values = c("#C70039")) +
  theme_minimal()

ggplot(df) +
  geom_line(aes(x = df$x_t, y = df$smooth_two_week)) +
  geom_point(aes(x = df$x_t, y = df$yt_lag_2, colour = "Two Weeks")) +
  labs(title = "Smoothed Lag Differences Between Oil and Gas by Weeks",
       y = "Smoothed, Logged and 2 Week Lagged Gas Difference",
       x = "Logged Oil Difference", color = "Legend") +
  scale_color_manual(values = c("#581845")) +
  theme_minimal()

ggplot(df) +
  geom_line(aes(x = df$x_t, y = df$smooth_three_week)) +
  geom_point(aes(x = df$x_t, y = df$yt_lag_3, colour = "Three Weeks")) +
  labs(title = "Smoothed Lag Differences Between Oil and Gas by Weeks",
       y = "Smoothed, Logged and 3 Week Lagged Gas Difference",
       x = "Logged Oil Difference", color = "Legend") +
  scale_color_manual(values = c("#FFC300")) +
  theme_minimal()



# Greating the Identity
df$x_t_greater_zero = ifelse(df$x_t>0, TRUE, FALSE)

# Creating x_{t-1}
df$x_t_1 = c(NA, df$x_t[1:(length(df$x_t))-1])

model_e = lm(formula = df$y_t ~ df$x_t_greater_zero + df$x_t_1 + df$x_t)
```

```r
summary(model_e)
plot(model_e)

acf(model_e$residuals)
plot(model_e$residuals, main = "Plotted residuals")
```