Time Series Analysis - lab01

Lennart Schilling (lensc874)

2019-09-17

Contents

Lab01

Assignment 1. Computations with simulated data.

1a. Generating two time series. Applying smoothing filter.

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. Apply a smoothing filter $v_t = 0.2(x_t + x_{t-1} + x_{t-2} + x_{t-3} + x_{t-4})$ to these two series and compare how the filter has affected them.

Generating two time series.

First, a function for each given time series models will be implemented. Using these functions makes it possible to generate both time series afterwards. w_t is expected to be normally distributed. From now on, the first given time series will be named as *ts1* and the second as *ts2*.

```r
# Implementing function to generate t1.
generate_ts1 = function(n) {
  # Initializing x_0, x_1 and vector to store generated values for x_t.
  x_0 = 0
  x_1 = 0
  x_t = c(x_0, x_1)
  # Generating further n-2 values for x_t.
  for (i in 1:(n-2)) {
    t = length(x_t)
    x_t = c(x_t, -0.8*x_t[t-1] + rnorm(n = 1))
  }
  # Returning generated time series as data frame.
  return(data.frame(t = 0:(n-1), x_t = x_t))
}

# Generating with 100 observations.
ts1 = generate_ts1(n = 100)

# Implementing function to generate ts2.
generate_ts2 = function(n) {
  # Initializing vector to store generated values for x_t.
  x_t = c()
  # Generating n values for x_t.
  for (t in 0:(n-1)) {
    x_t = c(x_t, cos((2*pi*t)/5))
  }
  # Returning generated time series as data frame.
  return(data.frame(t = 0:(n-1), x_t = x_t))
}

# Generating with 100 observations.
ts2 = generate_ts2(n = 100)
```

Applying smoothing filter.

Again, a function will be implemented to apply the smoothing filter on the specified time series. Since v_t will be calculated using the values from x_t to x_{t-4} and since both generated time series only consists of values for $t >= 0$, smoothing is only possible for $t >= 4$. The smoothed values will be added to the dataframes.

```r
# Implementing function to apply smoothing filter on input time series.
add_smooth_ts = function(ts) {
  # Initializing vector to store smoothed values v_t.
  v_t = c()
  # Generating smoothed time series from input time series.
  for (obs in 5:nrow(ts)) {
    v_t = c(v_t, 0.2*(ts$x_t[obs] + ts$x_t[obs-1] + ts$x_t[obs-2] + ts$x_t[obs-3] + ts$x_t[obs-4]))
  }
  # Adding smoothed time series to data frame.
  ts_smoothed = data.frame(t = 4:(nrow(ts)-1),
                           v_t = v_t)
  ts_combined = merge(x = ts,
                      y = ts_smoothed,
                      by = "t",
                      all.x = TRUE)
  return(ts_combined)
}

# Smoothing ts1.
ts1 = add_smooth_ts(ts1)

# Smoothing ts2.
ts2 = add_smooth_ts(ts2)
```

Plotting time series.

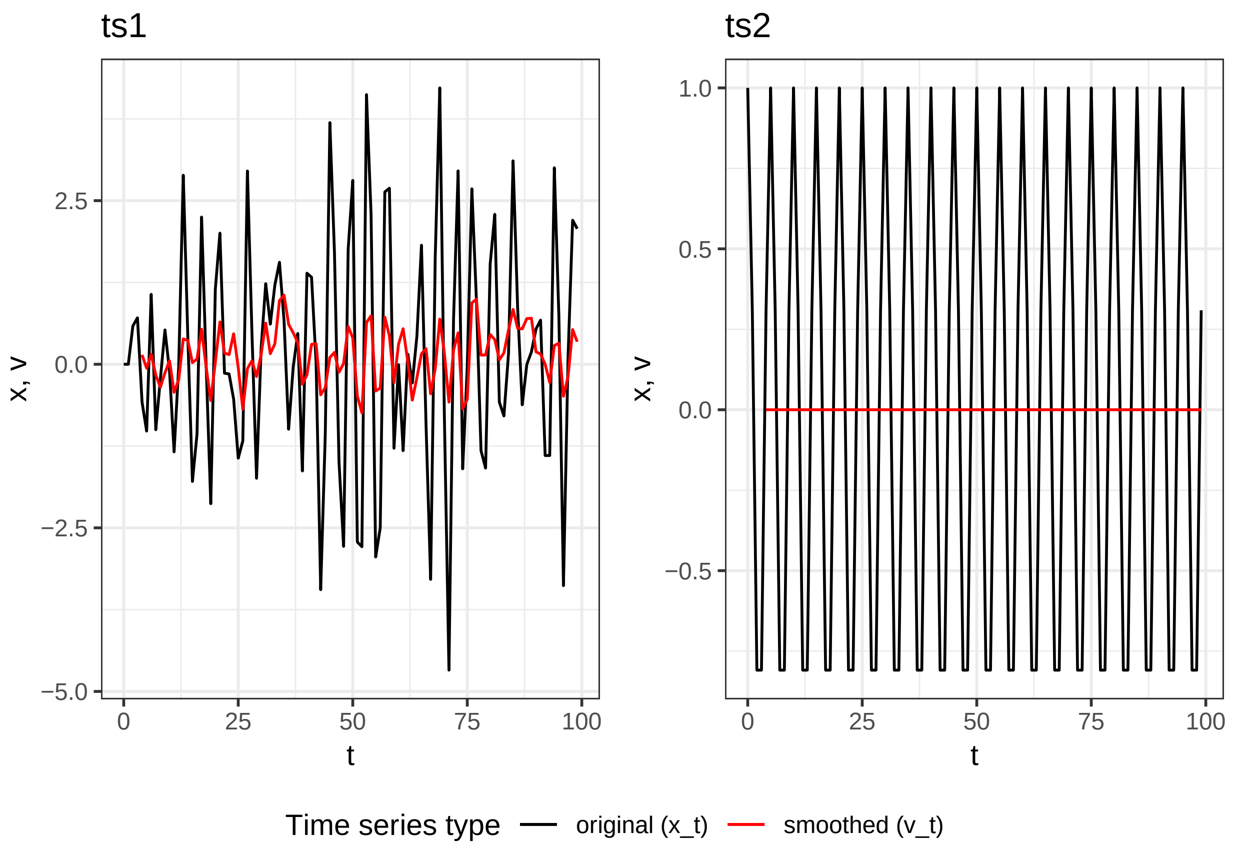
In the following, the affection of the filter will be graphically compared for both time series.

```r
# Implementing function to create plot for time series.
library(ggplot2)
plot_ts = function(ts) {
  ts_plot =   ggplot(data = ts) +
    geom_line(aes(x = t,
                  y = x_t,
                  color = "original (x_t)")) +
    geom_line(aes(x = t,
                  y = v_t,
                  color = "smoothed (v_t)")) +
    scale_color_manual(values = c("original (x_t)" = "black",
                                  "smoothed (v_t)" = "red")) +
    theme_bw() +
    labs(title = as.character(substitute(ts)),
         x = "t",
         y = "x, v",
         colour = "Time series type")
  return(ts_plot)
}

# Plotting both time series next to each other.
library(ggpubr)
```

```
Loading required package: magrittr
```

```r
ggarrange(plot_ts(ts1), plot_ts(ts2), ncol = 2, common.legend = TRUE, legend="bottom")
```

```
Warning: Removed 4 rows containing missing values (geom_path).

Warning: Removed 4 rows containing missing values (geom_path).

Warning: Removed 4 rows containing missing values (geom_path).
```



It can be seen that while for *ts1* the smoothing process has delivered good results (values are smoothed but still vary over time t), for *ts2* the filter led to a nonsatisfying result by creating a new time series i almost exactly 0 for every t so that the value v does not depend on t anymore.

1b. Investigating causality and invertibility.

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

Within the given time series expression, the left part of the equation represents the AR-part. Furthermore, the right part instead reflects the MA-part.

The equation can be transformed as follows:

$$(1 - 4B + 2B^2 + B^5)x_t = (1 + 3B^2 + B^4 - 4B^6)w_t$$

An ARMA-model is causal iff roots $\phi(z') = 0$ are outside unit circle.

```r
# Checking causality.
polyroot(c(1, -4, 2, 0, 0, 1))
```

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

An ARMA-model is invertible iff roots $\theta(z') = 0$ are outside unit circle.

```r
# Checking invertibility.
polyroot(c(1, 0, 3, 0, 1, 0, -4))
```

```
[1]  0.1375513+0.6735351i -0.1375513+0.6735351i -0.1375513-0.6735351i
[4]  0.1375513-0.6735351i  1.0580446+0.0000000i -1.0580446+0.0000000i
```

1c. Simulating from ARMA-process. Computing ACF using built-in R functions.

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.

Simulating from ARMA-process.

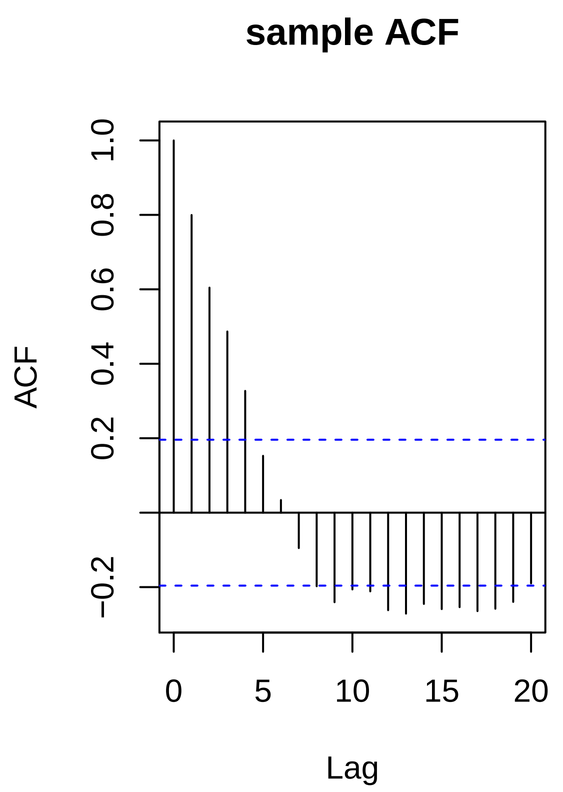
To simulate from the specified ARMA-process, we can use the built-in R function `arima.sim()`.

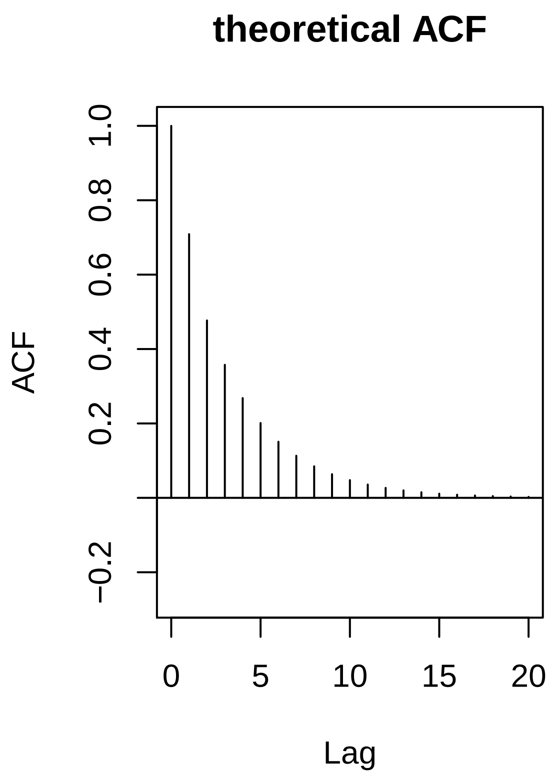
```r
# Setting seed.
set.seed(54321)

# Simulating 100 observations from given ARMA-process.
ts = arima.sim(list(ar = 3/4, ma = c(0, -(1/9))),
               n = 100)
```

Computing and plotting sample ACF and theoretical ACF.

To compute sample ACF, the built-in R function `acf()` is used. To compute the theoretical ACF, the built-in R function `ARMAacf()` is used. To compare both ACF, we will calculate the theoretical autocorrelations up to the maximum lag of the sample ACF. To create a plot for the theoretical ACF, we do not use any built-in R function which directly creates the plot. Instead, we are creating the plot manually using the obtained values in `theoretical_acf`. Finally, we can compare both ACF plots next to each other.

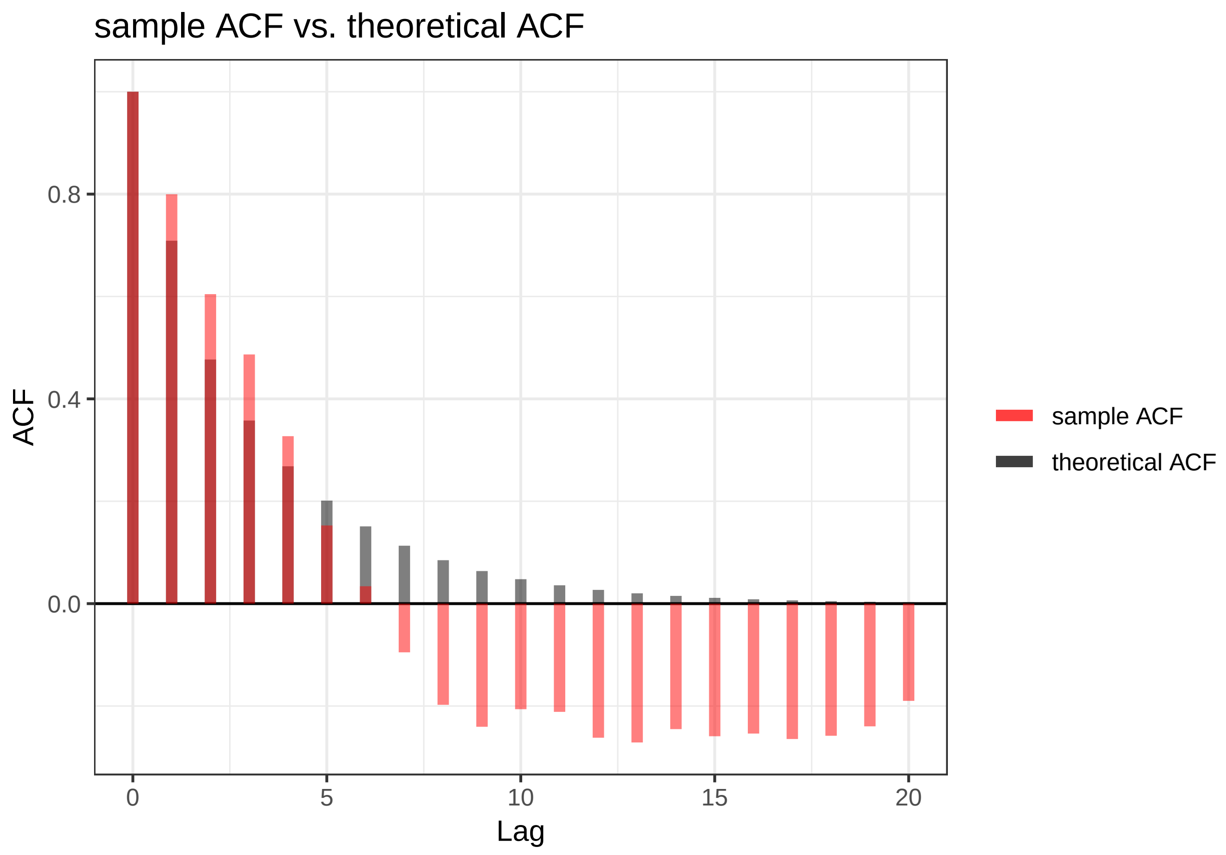
```r
# Plotting theoretical acf plot next to previously created plot sample_acf.
  # Defining following plots to be next to each other.
  par(mfrow = c(1,2))
  # Plotting sample ACF using simulated values.
  sample_acf = acf(x = ts,
                   type = "correlation",
                   plot = TRUE,
                   main = "sample ACF")
  # Storing theoretical acf.
  theoretical_acf = ARMAacf(ar = 3/4,
                            ma = c(0, -(1/9)),
                            lag.max = max(sample_acf$lag))
  # Plotting theoretial ACF.
  theoretical_acf_plot = plot(x = 0:max(sample_acf$lag),
                              y = theoretical_acf,
                              type = "h",
                              ylab = "ACF",
                              xlab = "Lag",
                              main = "theoretical ACF",
                              ylim = c(min(sample_acf$acf), 1))
  abline(h = 0)
```





We can also compare both together in one plot.

```r
ggplot(mapping = aes(x = 0:max(sample_acf$lag))) +
  geom_hline(aes(yintercept = 0)) +
  geom_segment(mapping = aes(y = theoretical_acf,
                             yend = 0,
                             xend = 0:max(sample_acf$lag),
                             color = "theoretical ACF"),
              alpha = 0.5,
              size = 2) +
  geom_segment(mapping = aes(y = as.numeric(sample_acf$acf),
                             yend = 0,
                             xend = 0:max(sample_acf$lag),
                             color = "sample ACF"),
              alpha = 0.5,
              size = 2) +
  scale_color_manual(values = c("theoretical ACF" = "black",
                                "sample ACF" = "red")) +
  theme_bw() +
  labs(title = "sample ACF vs. theoretical ACF",
       x = "Lag",
       y = "ACF",
       colour = "")
```



It becomes obvious that fot the first lags (0 to 5), both ACF are very similar. However, for lags larger than 5, one can observe a certain difference. While the theoretical correlations are continously larger than zero, the sample ACF also recognizes a lot of values smaller than zero.

Assignment 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.

2a. Exploring time series.

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?

Importing data and converting it to time series object.

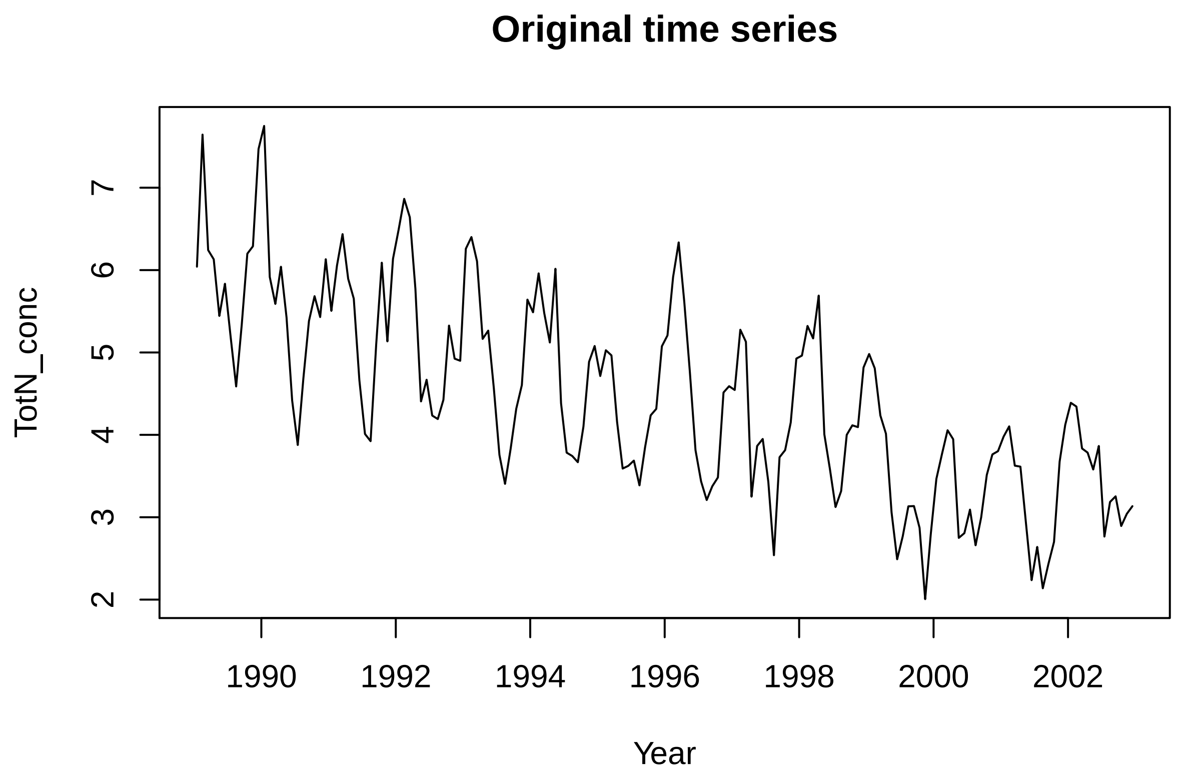
First, the data will be imported and converted.

```r
# Importing data.
data_df = read.csv2("Rhine.csv", sep = ";")

# Converting dataframe to time series object.
data_ts = ts(data_df)
```

Plotting time series.

The time series will be plotted.

```r
# Plotting time series.
plot.ts(x = data_ts[, 3],
        y = data_ts[, 4],
        type = "l",
        xlab = "Year",
        ylab = "TotN_conc",
        main = "Original time series")
```



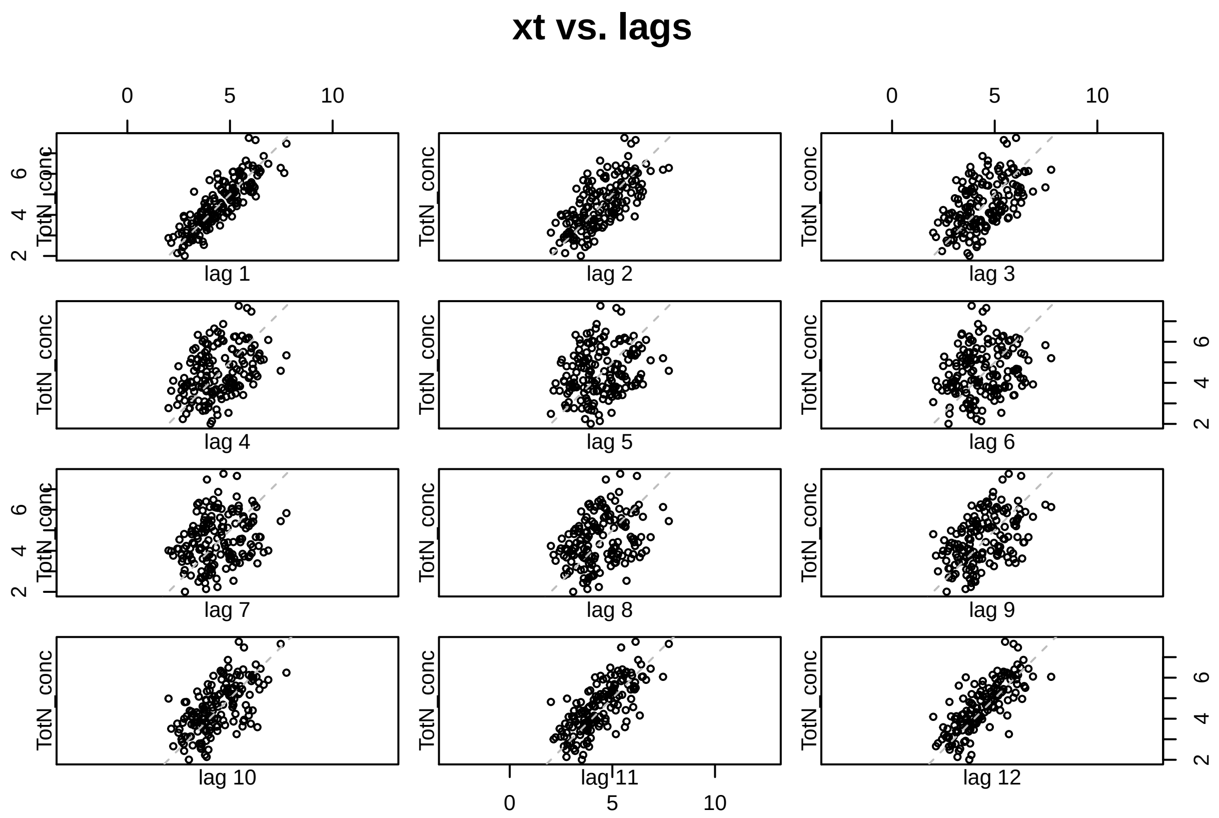
```r
  # Other opportunity: ggplot.
  # ggplot() +
  #     geom_line(aes(x = data_ts[, 3],
  #                   y = data_ts[, 4])) +
  #     theme_bw() +
  #     labs(title = "Original time series",
  #          x = "Year",
  #          y = "TotN_conc")
```

In general, the time series seems to be characterized by a negative linear trend. The mean is decreasing over time. Even if the variance does not stay exactly the same over time, one can say that it the variance of the time series does not increase or decrease too much and stays quite constant over time. Furthermore, it seems to follow a certain seasonality since on a high peak usually a low peak follows and the other way around. Thus, a similar pattern is repeating again and again.

Plotting scatterplots to compare lags.

Also, scatterplots will be created where x_t is compared to the previous lags $x_{t-1}, ..., x_{t-12}$ by usage of the built-in R function `lag.plot()`. This implies that we compare a data point related to the twelve previous data points. It follows that we try to see a connection between a month's record and the records of the previous twelve months.

```r
# Creating scatterplots for x_t vs. x_t-1, ..., x_t-12.
TotN_conc =  data_ts[, 4]
lag.plot(x = TotN_conc,
         lags = 12,
         main = "xt vs. lags")
```



The scatterplot confirms the seasonality. While the linear trend is clearly visible for lags close to zero or close to twelve, this pattern disappears the closer you get to lags like lag 6.

As a result, a value of a month seems to be very similar to the directly previously recorded values and also to the values recorded around a year (twelve months) ago. However, the value does not show that much similarity to values recorded around half a year ago. This implies that there is probably a seasonality over around half a year (e.g. winter-summer-seasonality) within the data.

2b. Fitting linear model. Analysing residuals.

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.

Fitting linear model.

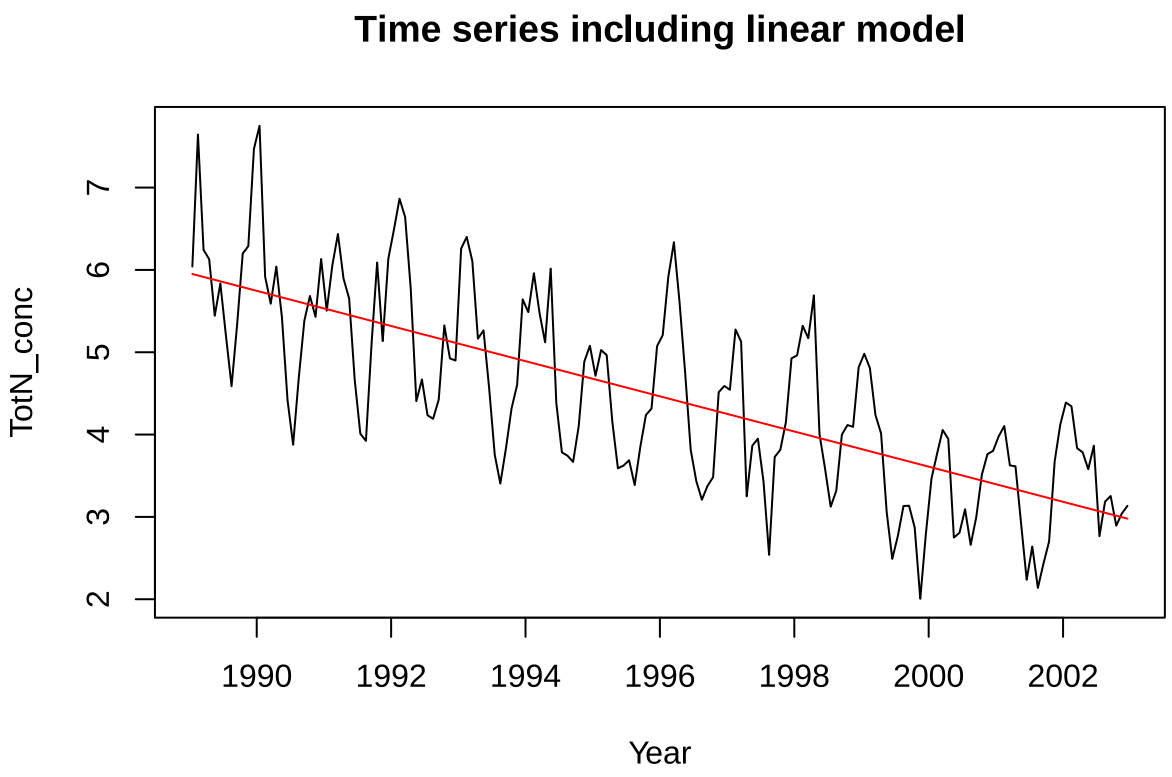
First, we fit a linear model w.r.t. t to the time series.

```r
# Fitting linear mode w.r.t. t to time series.
linear_model = lm(formula = TotN_conc ~ Time,
                  data = data_ts)
# Printing information about coefficients of linear model.
knitr::kable(summary(linear_model)$coefficients[, c(1,4)])
```

	Estimate	Pr(>|t|)
(Intercept)	430.7072475	0
Time	-0.2135483	0

The fitted linear model will be implemented in the plot of the time series.

```r
# Plotting time series with linear model.
plot.ts(x = data_ts[, 3],
        y = data_ts[, 4],
        type = "l",
        xlab = "Year",
        ylab = "TotN_conc",
        main = "Time series including linear model")
lines(x = linear_model$model$Time,
      y = linear_model$fitted.values,
      col = "red")
```

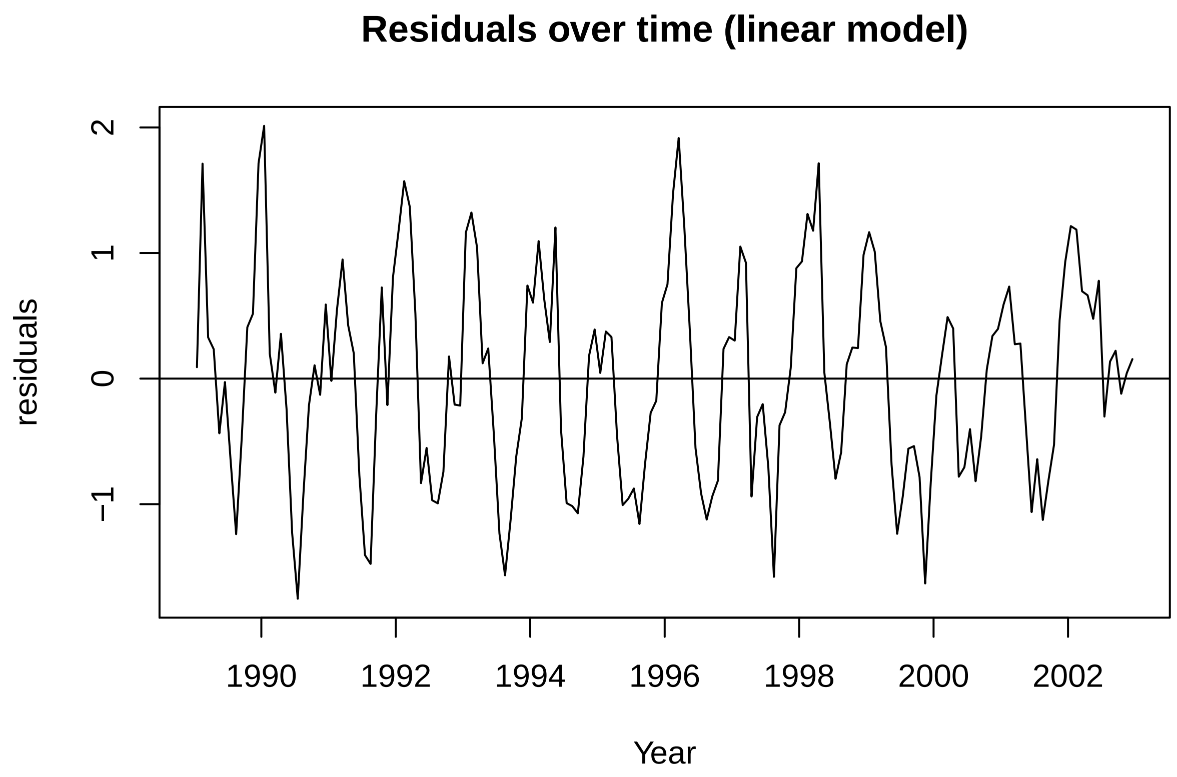


Both the graph and the given coeffients information (p-value for the β of time smaller than 5%) confirm the guess about the significant negative linear trend.

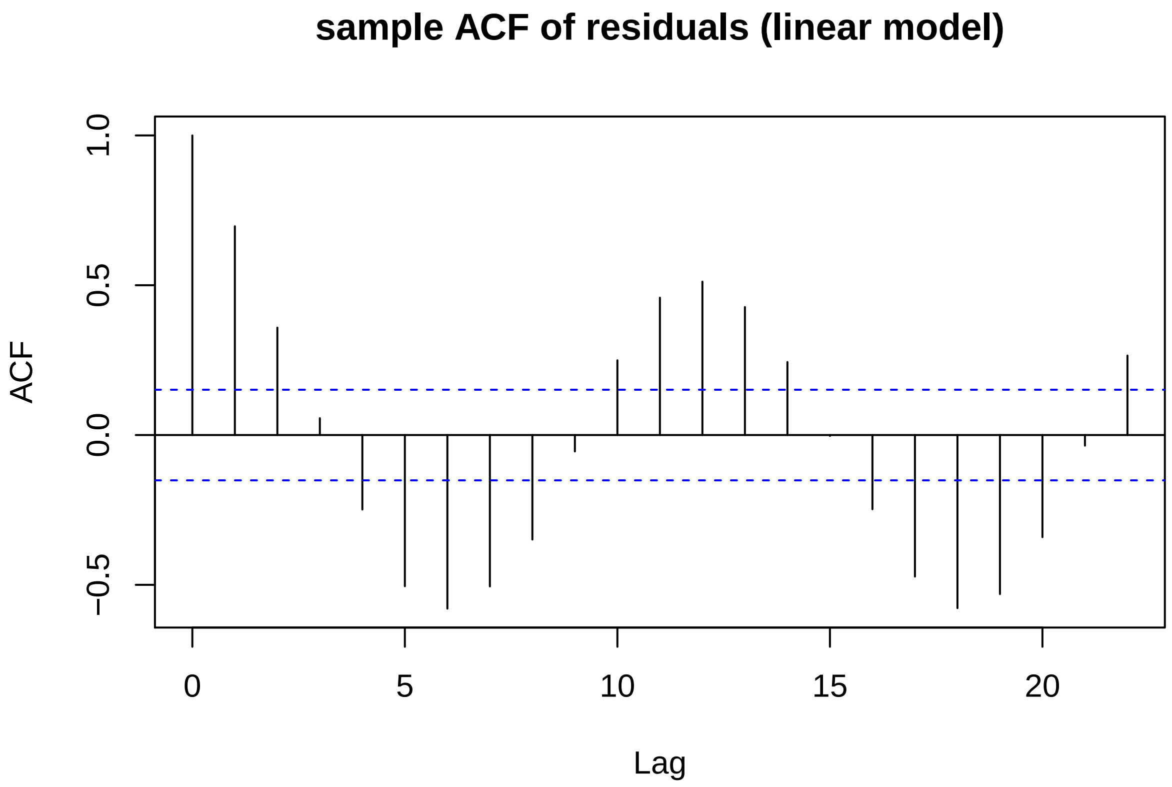
Analyzing residuals.

Comparing the actual time series to the fitted model, we are also able to analyze the residual pattern by plotting the residuals over time and analyzing the sample ACF of the residuals.

```
# Plotting residuals over time.
plot(x = linear_model$model$Time,
     y = linear_model$residuals,
     xlab = "Year",
     ylab = "residuals",
     type = "l",
     main = "Residuals over time (linear model)")
abline(h = 0)
```



```r
# Plotting sample ACF of residuals.
acf(x = linear_model$residuals,
    type = "correlation",
    plot = TRUE,
    main = "sample ACF of residuals (linear model)")
```



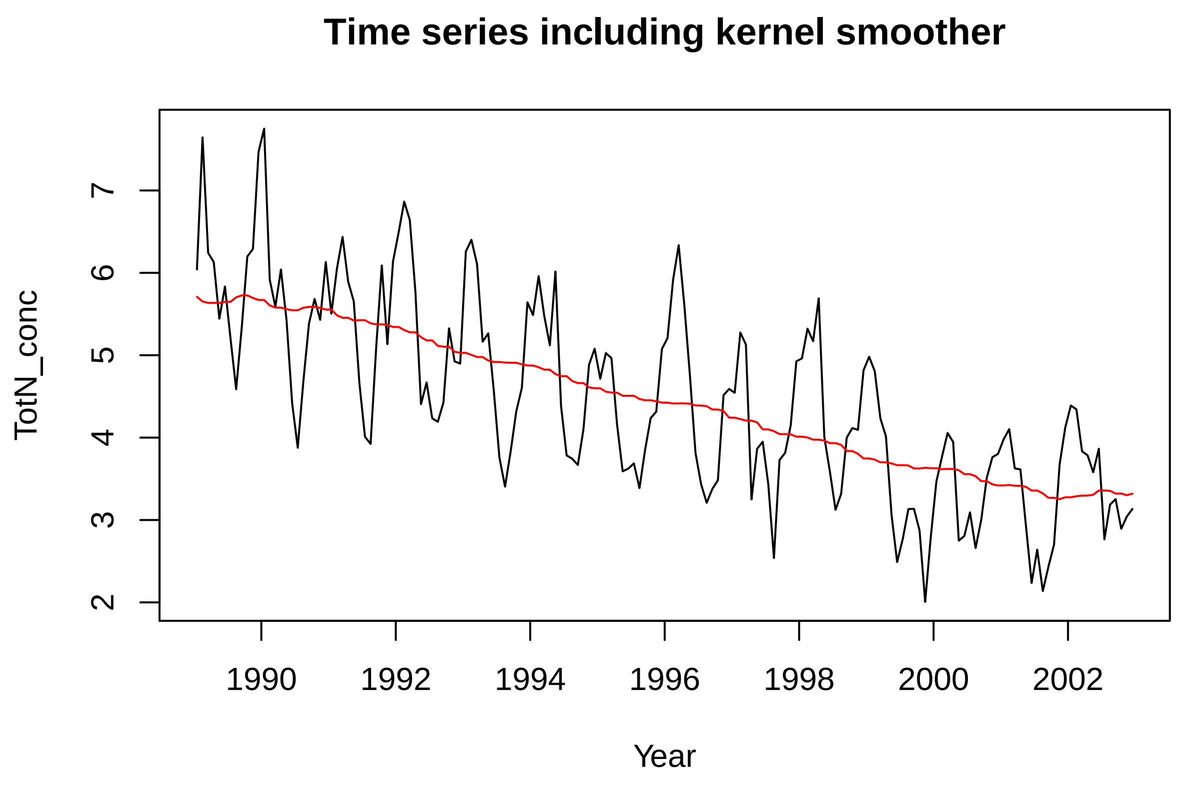
Looking at the plot which shows the residuals over time, a seasonal pattern seem to be visible again. Repetitively, positive peaks are followed by negative peaks and the other way around. Again, the time difference between the peaks is about half a year. The sample ACF plot confirms this pattern. A residual at time x_t seems to have a high correlation to other residuals which are recorded diretly before (x_{t-1} and x_{t-2}). The same correlation can be seen for residuals recorded about twelve months (one year) ago. In addition to that, a negative correlation can be seen for residuals recorded arond half a year ago. As a result, this plot also confirms the guess about the seasonality over around half a year (e.g. winter-summer-seasonality) within the data.

2c. Fitting kernel smoother. Analysing residuals.

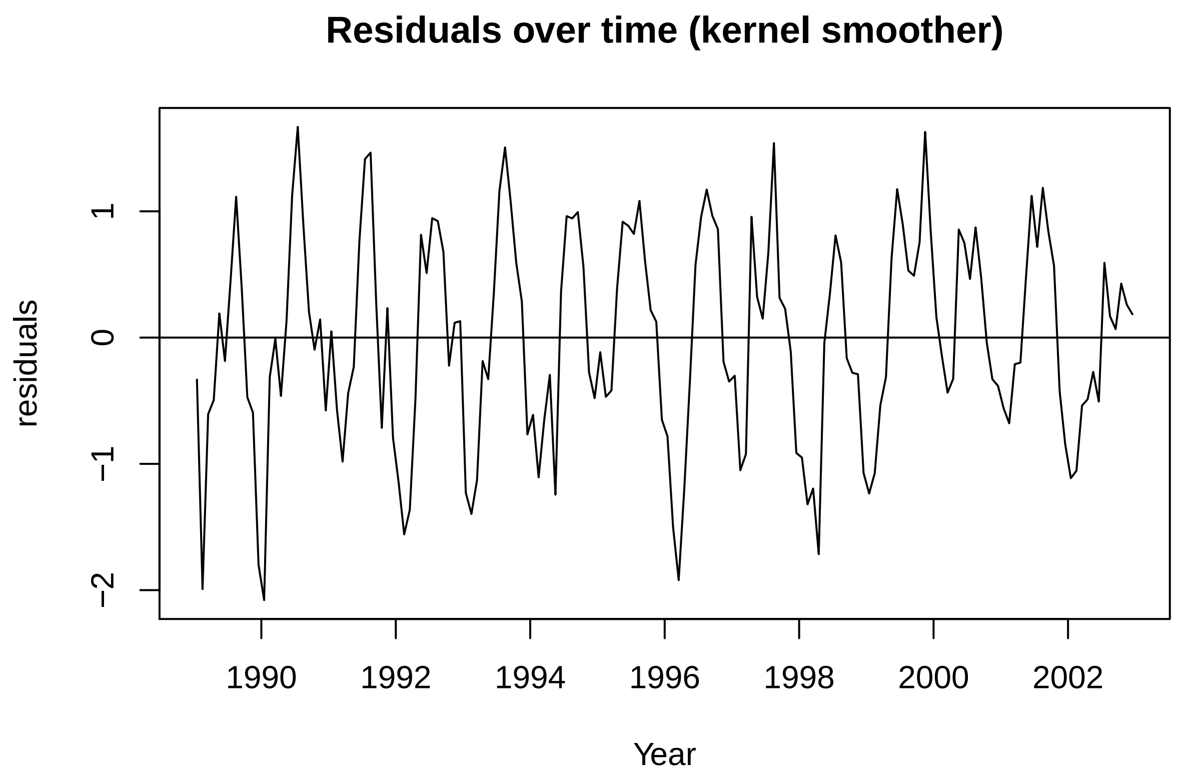
Eliminate the trend by fitting a kernel smoother with respect to t to the time series (choose a reasonable bandwidth yourself so the fit looks reasonable). Analyze the residual pattern and the sample ACF of the residuals and compare it to the ACF from step b). Conclusions? Do residuals seem to represent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

The general procedure from *2b* will be repeated. This time, the trend will not be eliminated by fitting a linear model but by fitting a kernel smoother. To do so, thee built-in R function `ksmooth()` will be used. By testing different values, a `bandwith` = 5 seems to be a reasonable choice.

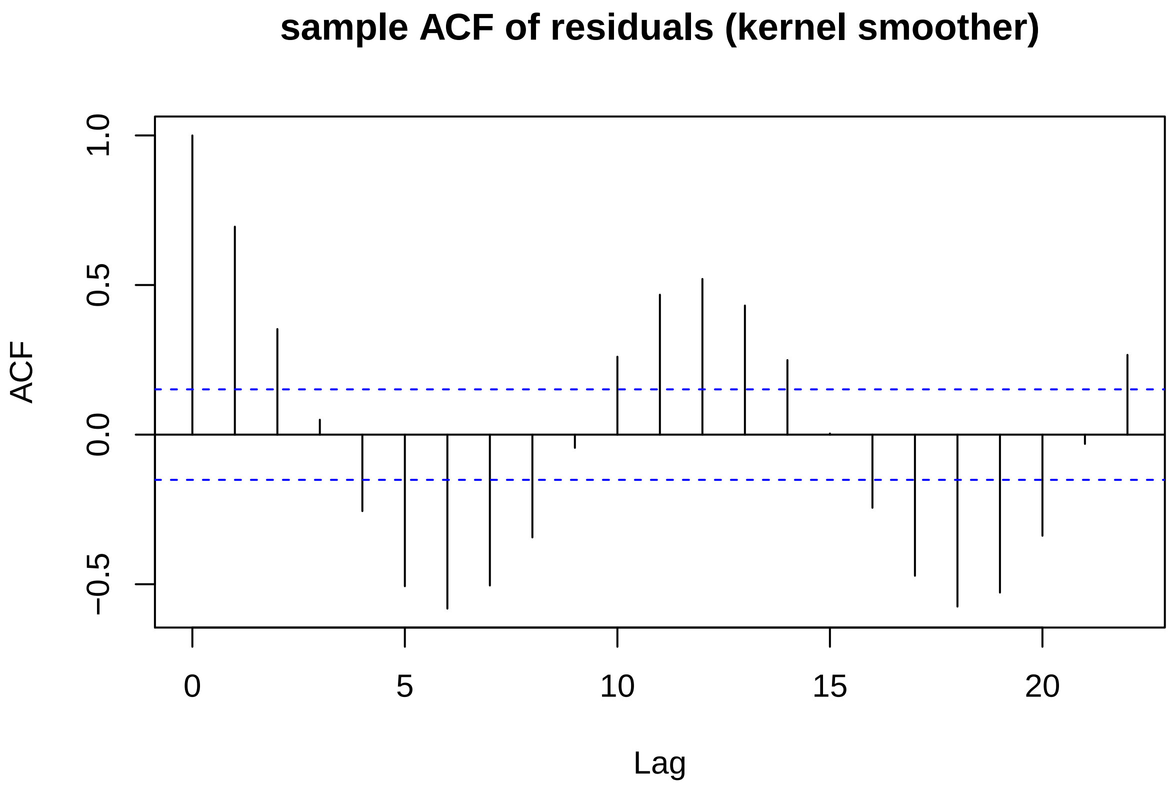
```r
# Fitting kernel smoother w.r.t. t to time series.
kernel_model = ksmooth(x = data_ts[, 3],
                       y = data_ts[, 4],
                       bandwidth = 5)

# Plotting time series with fitted kernel smoother.
plot.ts(x = data_ts[, 3],
        y = data_ts[, 4],
        type = "l",
        xlab = "Year",
        ylab = "TotN_conc",
        main = "Time series including kernel smoother")
lines(x = kernel_model$x,
      y = kernel_model$y,
      col = "red")
```



```r
# Plotting residuals over time.
plot(x = linear_model$model$Time,
     y = kernel_model$y - data_ts[, 4],
     xlab = "Year",
     ylab = "residuals",
     type = "l",
     main = "Residuals over time (kernel smoother)")
abline(h = 0)
```



```r
# Plotting sample ACF of residuals.
acf(x =  kernel_model$y - data_ts[, 4],
    type = "correlation",
    plot = TRUE,
    main = "sample ACF of residuals (kernel smoother)")
```



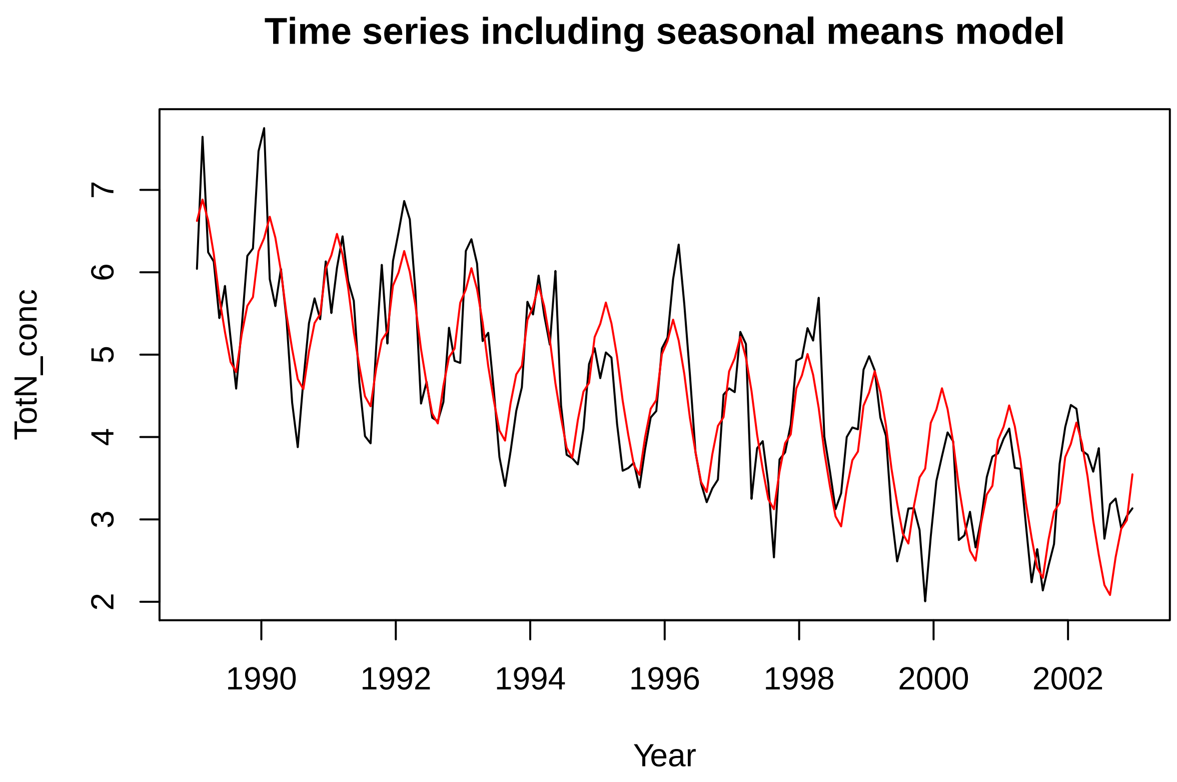
The knowledge we gain from these plots are the the same compared to the usage of the fitted linear model: Clearly, a seasonality is shown. The residuals over time do not represent a stationary time series. Instead, they follow as seasonal pattern.

2d. Fitting seasonal means model. Analysing residuals.

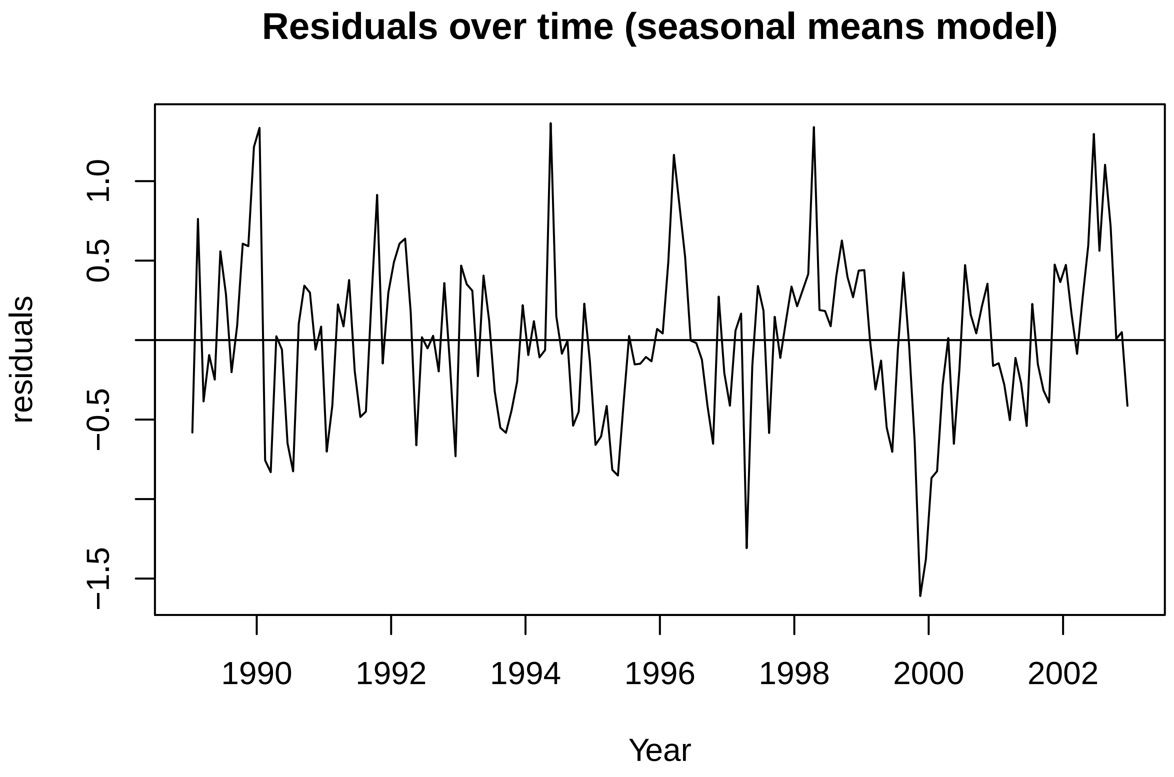
Eliminate the trend by fitting the following so-called seasonal means model: $x_t = \alpha_0 + \alpha_1 t + \beta_1 I(month = 2) + ... + \beta_1 2 I(month = 12) + w_t$, where $I(x) = 1$ if x is true and 0 otherwise. Fitting of this model will require you to augment data with a categorical variable showing the current month, and then fitting a usual linear regression. Analyze the residual pattern and the ACF of residuals.

Again, the general procedure from *2b* and *2c* will be repeated. This time, the trend will be eliminated by fitting a seasonal means model.

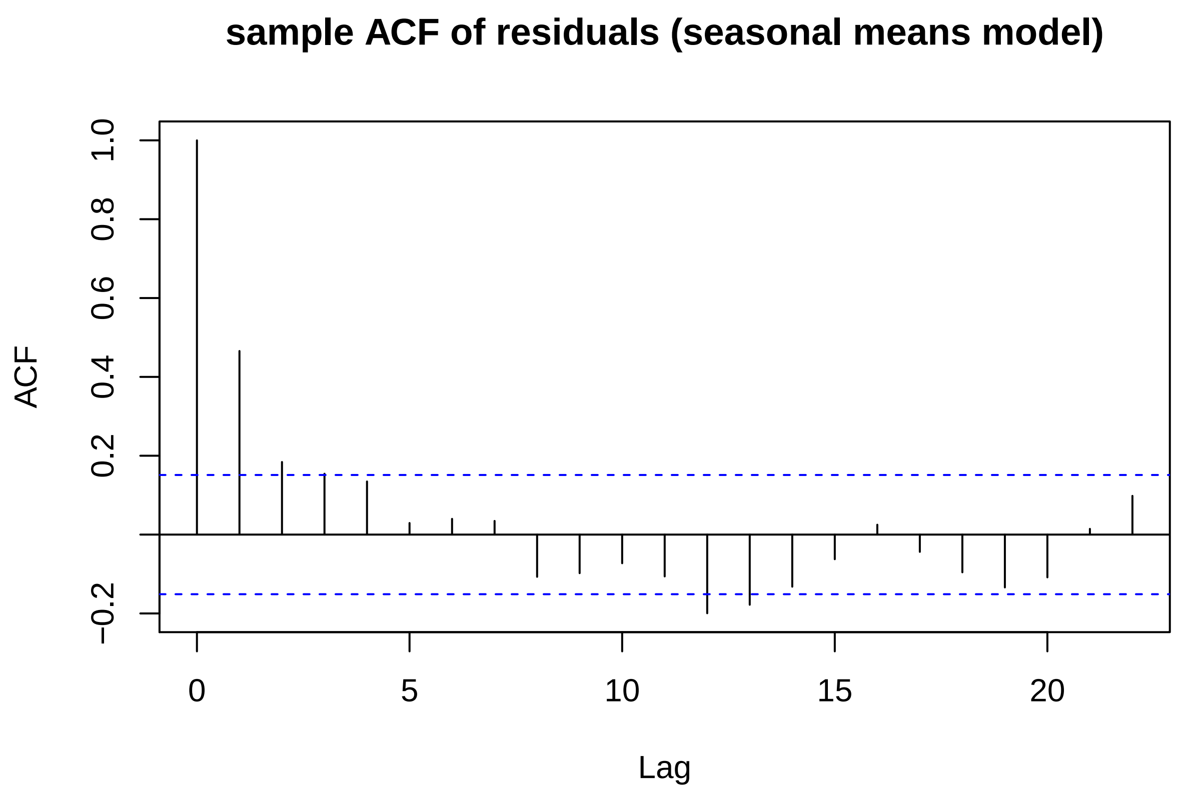
```r
# Fitting seasonal means model.
seasonal_means_model = lm(formula = TotN_conc ~ Time + as.factor(Month),
                          data = data_ts)

# Plotting time series with fitted seasonal means model.
plot.ts(x = data_ts[, 3],
        y = data_ts[, 4],
        type = "l",
        xlab = "Year",
        ylab = "TotN_conc",
        main = "Time series including seasonal means model")
lines(x = seasonal_means_model$model$Time,
      y = seasonal_means_model$fitted.values,
      col = "red")
```



```r
# Plotting residuals over time.
plot(x = seasonal_means_model$model$Time,
     y = seasonal_means_model$residuals,
     xlab = "Year",
     ylab = "residuals",
     type = "l",
     main = "Residuals over time (seasonal means model)")
abline(h = 0)
```



```r
# Plotting sample ACF of residuals.
acf(x =  seasonal_means_model$residuals,
    type = "correlation",
    plot = TRUE,
    main = "sample ACF of residuals (seasonal means model)")
```



It is obvious that in this case, not only the trend but also the seasonality has been removed. By analyzing the plots which show the residuals over time and the ACF of the residuals, one can see that the data looks much more stationary than before using the other models. No seasonal, repetitively pattern is recognizable anymore.

2e. Variable selection.

Perform stepwise variable selection in model from step d). Which model gives you the lowest AIC value? Which variables are left in the model?

To perform variable selection, we use the function `step()` from the `MASS`-package. This function automatically performs stepwise model selection based on a comparison of the AIC.

```
library(MASS)

step(object = seasonal_means_model,
     direction = "both")
```

```
Start:  AIC=-202.02
TotN_conc ~ Time + as.factor(Month)

                   Df Sum of Sq     RSS      AIC
<none>                           43.237 -202.023
- as.factor(Month) 11    68.524 111.761  -64.477
- Time              1   118.387 161.624   17.499


Call:
lm(formula = TotN_conc ~ Time + as.factor(Month), data = data_ts)

Coefficients:
       (Intercept)                Time   as.factor(Month)2
         420.82746            -0.20824             0.27659
 as.factor(Month)3   as.factor(Month)4   as.factor(Month)5
           0.04006            -0.34643            -0.86165
 as.factor(Month)6   as.factor(Month)7   as.factor(Month)8
          -1.26114            -1.60808            -1.71242
 as.factor(Month)9  as.factor(Month)10  as.factor(Month)11
          -1.23669            -0.87446            -0.75127
as.factor(Month)12
          -0.17745
```

Since a lower AIC suggests a better model, the function analyzed that by removing variables from the model, the model only gets worse. That is why all independent variables (`Month`, `Time`) are kept.

Assignment 3. Analysis of oil and gas time series.

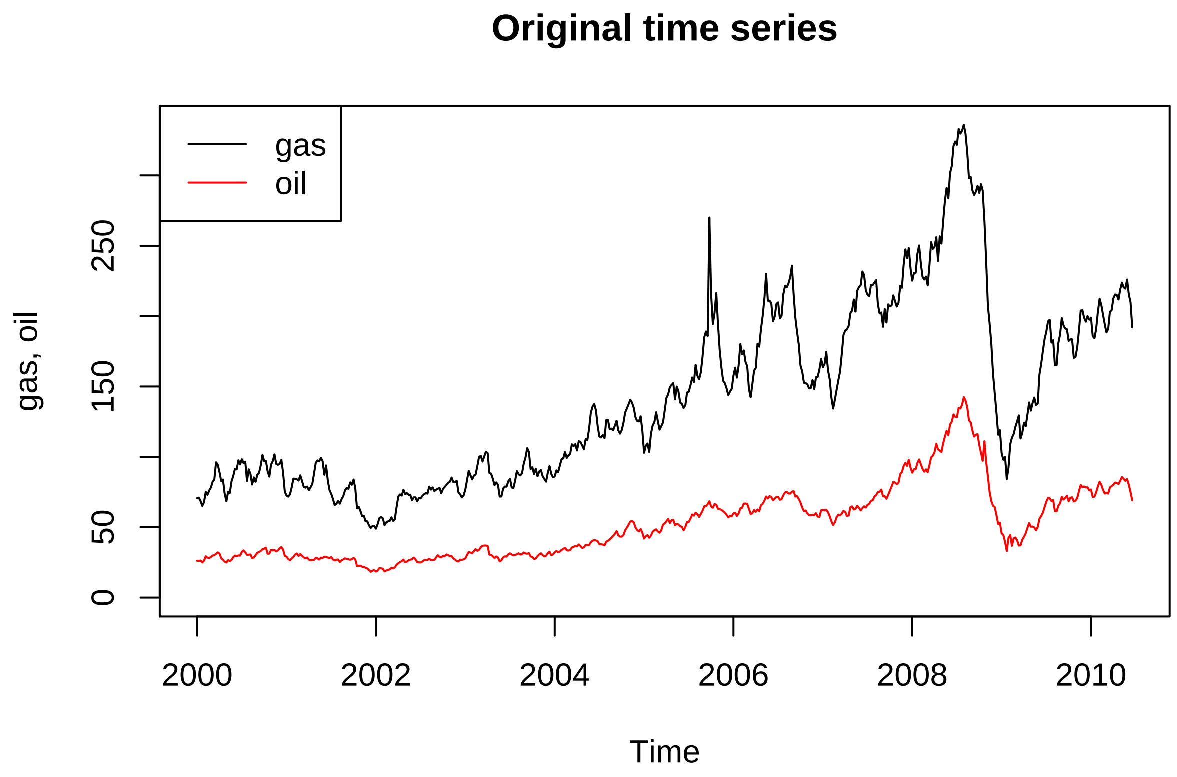
Weekly time series oil and gas present in the package astsa show the oil prices in dollars per barrel and gas prices in cents per dollar.

```r
# Loading data.
library(astsa)
data(oil)
data(gas)
```

3a. Plotting original time series.

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.

```r
# Plotting both time series in one plot.
plot.ts(x = gas,
        type = "l",
        main = "Original time series",
        ylim = c(0, max(gas)),
        ylab = "gas, oil")
lines(x = oil,
      col = "red")
legend("topleft", legend = c("gas", "oil"), lty = c(1,1), col = c("black", "red"))
```

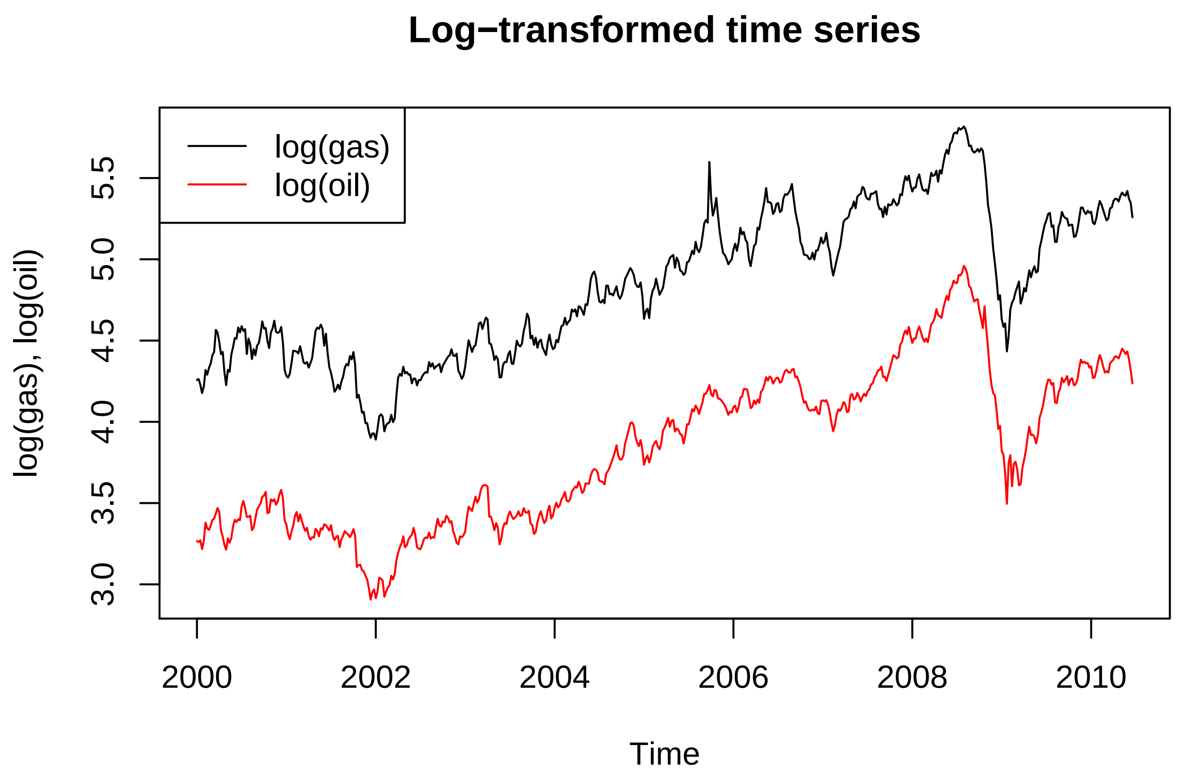


Both time series do not seem to be stationary. Instead, the mean is increasing over time which means that both time series seem to be characterized by a linear positive trend over time. Furthermore, they seem to be related to each other, because after 2008, both prices fall enormously at the same time and in general follow the same pattern.

3b. Plotting log-transformed time series.

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?

```r
# Log-transforming data.
log_oil = log(oil)
log_gas = log(gas)

# Plotting both transformed time series in one plot.
plot.ts(x = log_gas,
        type = "l",
        main = "Log-transformed time series",
        ylim = c(min(log_oil), max(log_gas)),
        ylab = "log(gas), log(oil)")
lines(x = log_oil,
      col = "red")
legend("topleft", legend = c("log(gas)", "log(oil)"), lty = c(1,1), col = c("black", "red"))
```



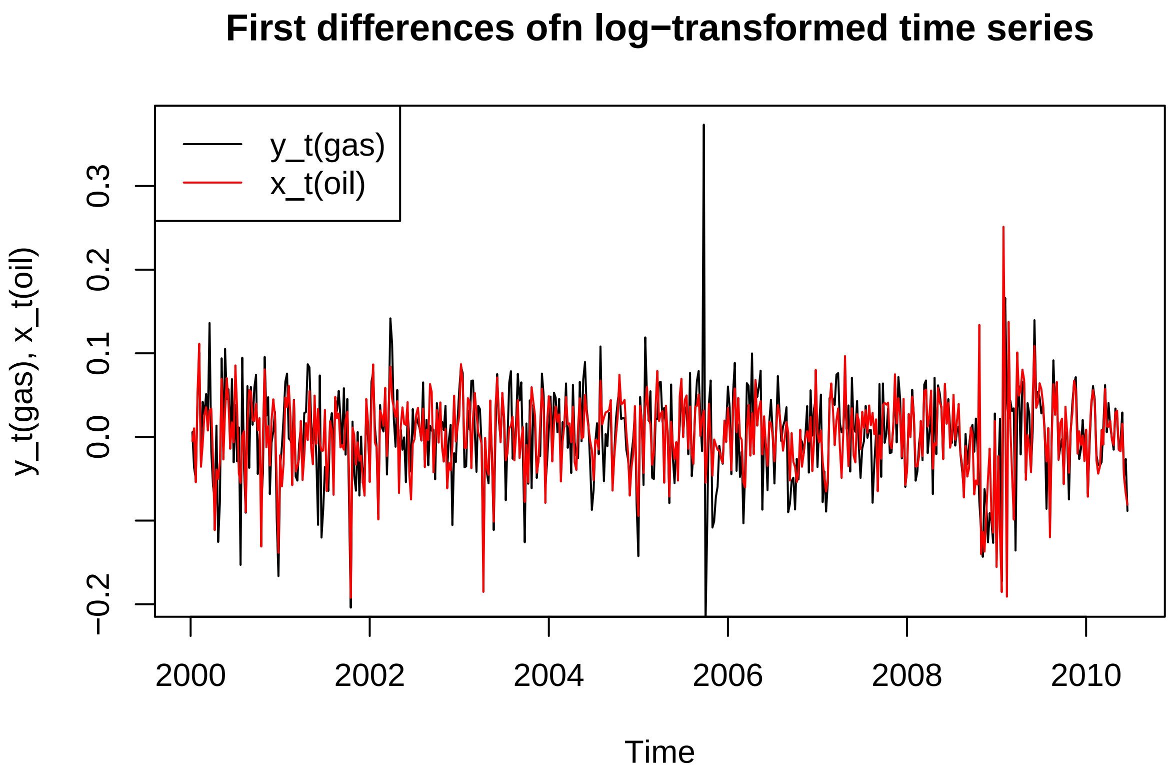
Log-transformations are often used to stabilize the variance. Comparing the origianl time series to the transformed ones, one can see that the variance is not close to constant yet. However, the differences of the variances over time have decreased.

3c. Computing first differences of log-transformed data. Plotting.

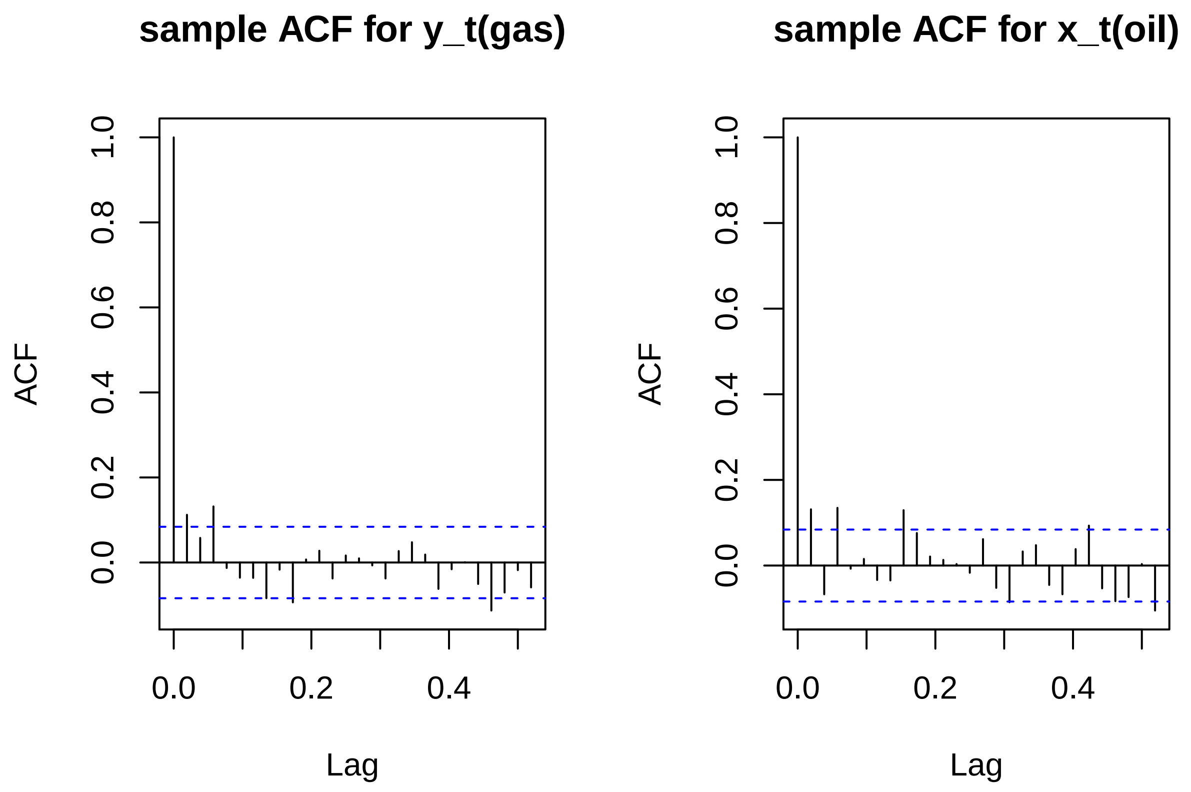
To eliminate trend, compute the first difference of the transformed data, plot the detrended series, check their ACFs and analyze the obtained plots. Denote the data obtained here as x_t(oil) and y_t (gas).

The first difference is given by $\nabla x_t = x_t - x_{t-1}$. To compute the differences x_t(oil) and y_t(gas), we use the built-in R function `diff()`.

```r
# Computing first difference of the transformed data.
log_gas_diff = diff(x = log_gas, lag = 1)
log_oil_diff = diff(x = log_oil, lag = 1)

# Plotting both transformed time series differences in one plot.
plot.ts(x = log_gas_diff,
        type = "l",
        main = "First differences ofn log-transformed time series",
        ylim = c(min(log_oil_diff), max(log_gas_diff)),
        ylab = "y_t(gas), x_t(oil)")
lines(x = log_oil_diff,
      col = "red")
legend("topleft", legend = c("y_t(gas)", "x_t(oil)"), lty = c(1,1), col = c("black", "red"))
```



```r
# Plotting sample ACF.
par(mfrow = c(1,2))
  # Plotting sample ACF for log_gas_diff.
  sample_acf_gas = acf(x = log_gas_diff,
                       type = "correlation",
                       plot = TRUE,
                       main = "sample ACF for y_t(gas)")
  # Plotting sample ACF for log_oil_diff
  sample_acf_gas = acf(x = log_oil_diff,
                       type = "correlation",
                       plot = TRUE,
                       main = "sample ACF for x_t(oil)")
```

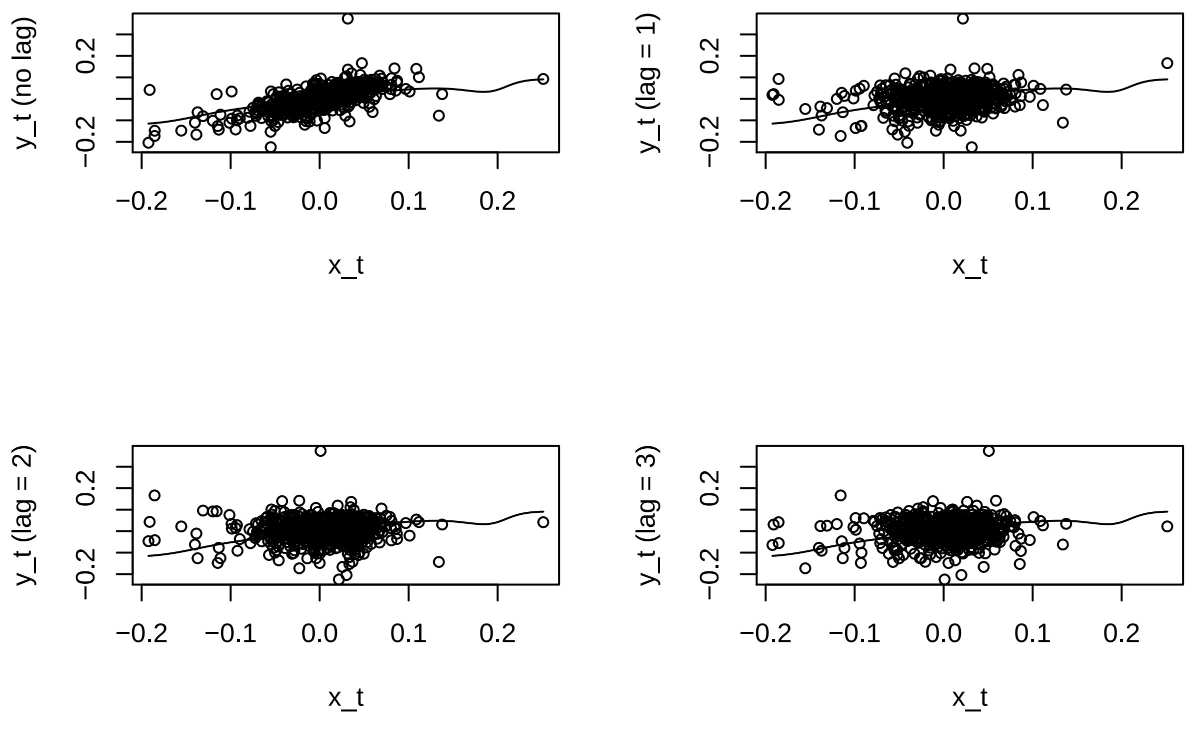


In general, excluding a few outliers, both time series seem to be stationary. The mean is very constant over time. Excluding the mentioned outliers, same could be said for the variance. Furthermore, looking at the ACF plots, it becomes obvious that both differenced time series are not characterized by a significant autocorrelation anymore.

3d. Analyzing scatterplots for different lags.

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
par(mfrow = c(2,2))

plot(x = log_oil_diff, y = log_gas_diff, xlab = "x_t", ylab = "y_t (no lag)")
lines(ksmooth(x = log_oil_diff, y = log_gas_diff, bandwidth = 0.1, kernel = "normal"))

plot(x = log_oil_diff, y = lag(log_gas_diff, 1), xlab = "x_t", ylab = "y_t (lag = 1)")
lines(ksmooth(x = log_oil_diff, y = lag(log_gas_diff, 1), bandwidth = 0.1, kernel = "normal"))

plot(x = log_oil_diff, y = lag(log_gas_diff, 2), xlab = "x_t", ylab = "y_t (lag = 2)")
lines(ksmooth(x = log_oil_diff, y = lag(log_gas_diff, 2), bandwidth = 0.1, kernel = "normal"))

plot(x = log_oil_diff, y = lag(log_gas_diff, 3), xlab = "x_t", ylab = "y_t (lag = 3)")
lines(ksmooth(x = log_oil_diff, y = lag(log_gas_diff, 3), bandwidth = 0.1, kernel = "normal"))
```



3e.

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.