Computer exercises 6

Demo exercises

6.1

The file `const.txt` contains monthly data of the number of started construction projects in a neighborhood in the USA between the years 1966 and 1974.

a) Visualize the data. Does the time series look stationary?

b) Use the function `stl` to decompose the time series. In other words, decompose the time series into a trend component, a seasonal component and a random component.

c) Use the following filter to estimate the trend:

$$y_t = \frac{1}{24}\left(x_{t-6} + 2x_{t-5} + 2x_{t-4} + ... + 2x_t + ... + 2x_{t+4} + 2x_{t+5} + x_{t+6}\right).$$

Plot the obtained estimate y_t, the estimate given by the function `stl` and the original time series into a single figure. Are there differences between the estimates?

d) Remove the trend and seasonal component from the time series by using difference operations. Use the function `stl` to decompose the obtained time series.

Solution

a) Read the data and plot the time series. By Figure 1, the time series `const` does not seem stationary.

```
const <- read.table("data/const.txt", header = TRUE, sep = ",", row.names = 1)
const <- ts(const[[1]], start = 1966, frequency = 12)
plot(const)
```

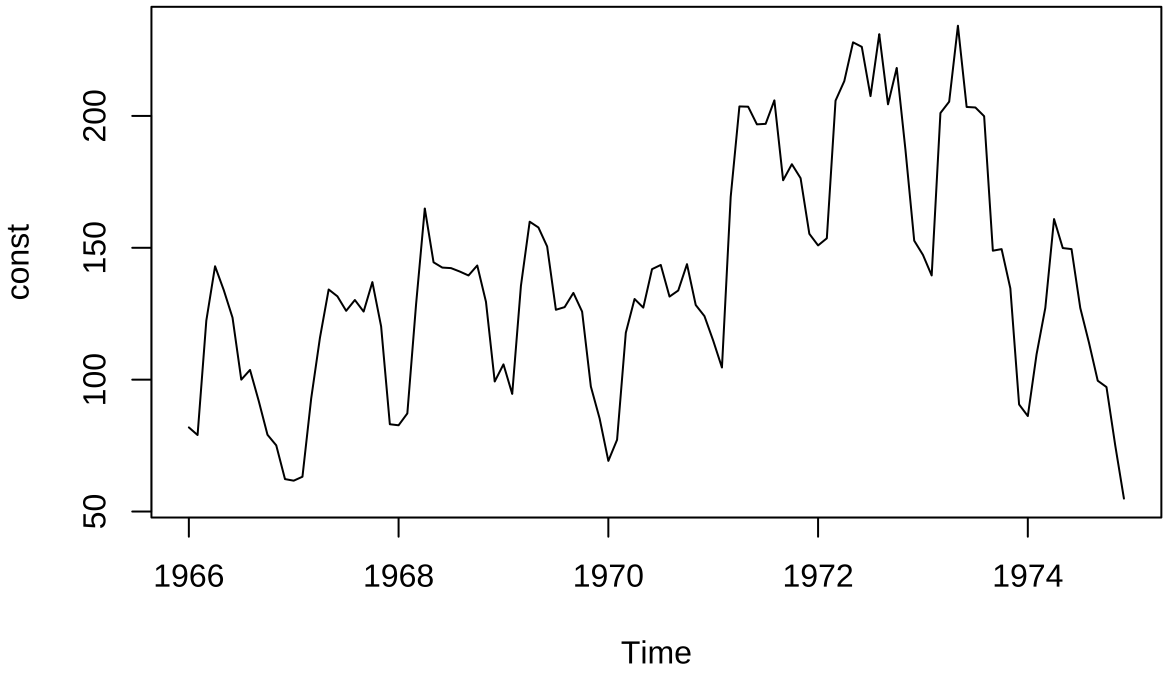


Figure 1: Original time series `const`.

b) Next we decompose time series into trend (m_t), seasonal (s_t) and random (e_t) components,

$$x_t = m_t + s_t + e_t.$$

Decomposition can be done with the function `stl`.

```
decomp <- stl(const, s.window = "periodic")
plot(decomp)
```

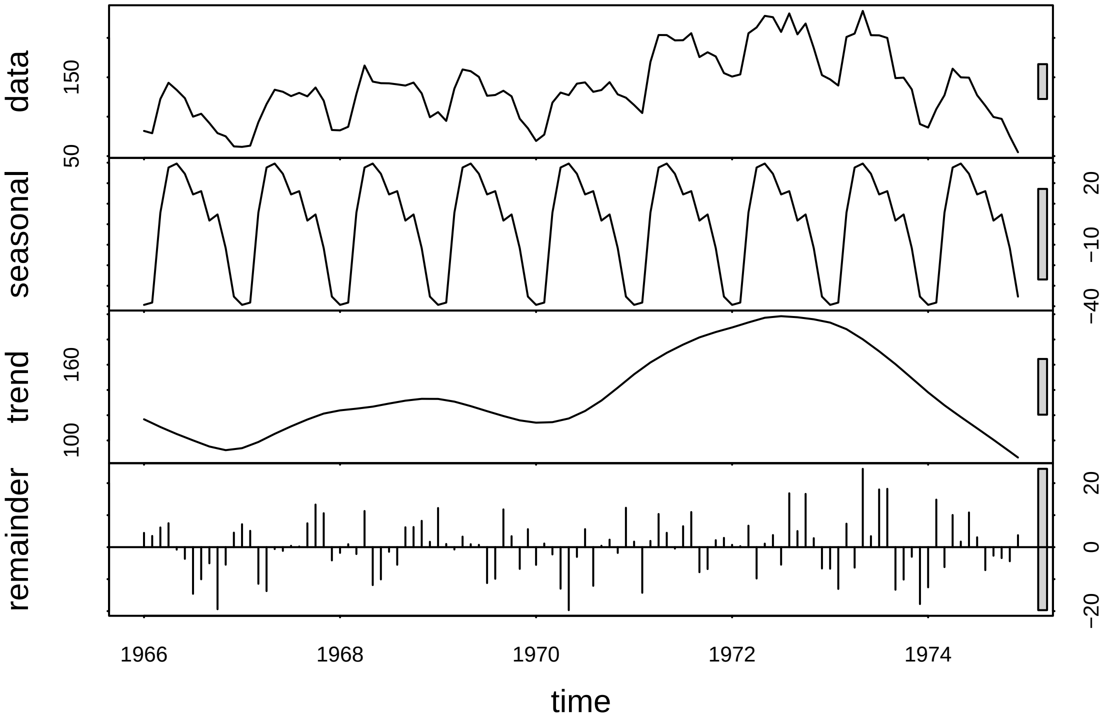


Figure 2: Decomposition of the time series `const`.

c) The filtering can be applied conveniently by utilizing the function `filter`.

```
trend_filter <- filter(const, c(1, rep(2, 11), 1) / 24)
trend_stl <- decomp$time.series[, 2]

plot(const, lty = 3, col = "black")
lines(trend_filter, lty = 2, col = "red")
lines(trend_stl, lty = 1, col = "blue")
legend("topleft", legend = c("Time series", "Filter", "STL"),
       col = c("black", "red", "blue"), lty = c(3, 2, 1))
```

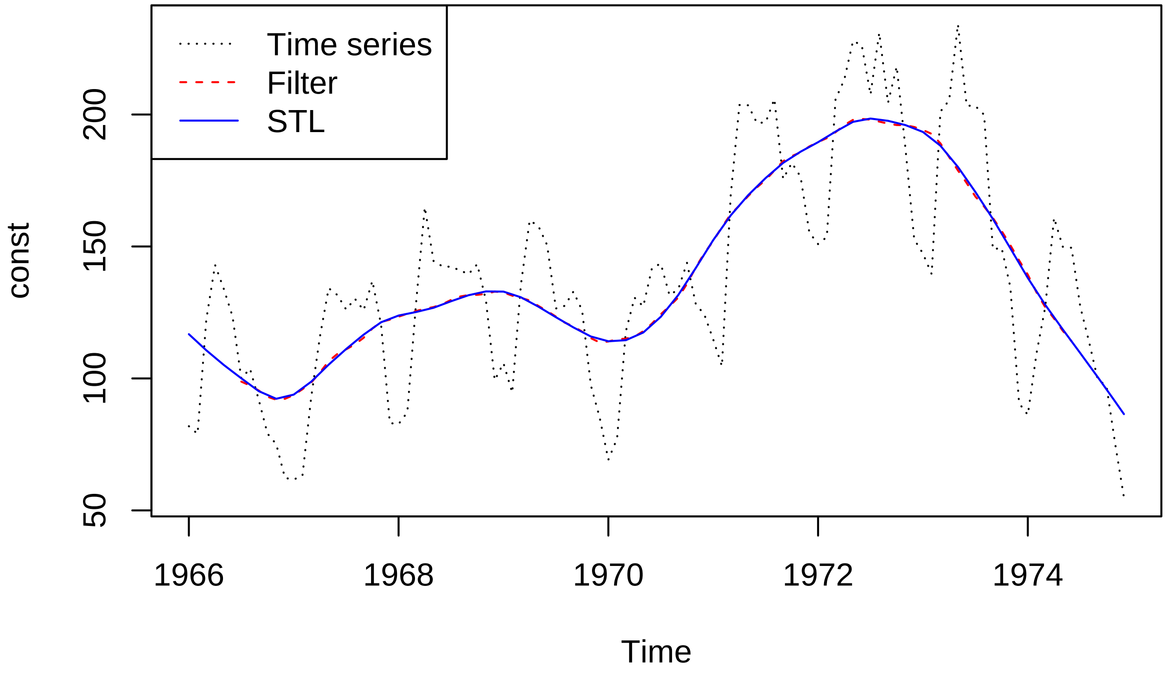


Figure 3: The original time series as gray, the trend given by `stl` as blue and the trend given by `filter` as red.

Figure 3 shows that the trend given by `stl` is almost identical to the one given by filtering. However, the trend given by the filter is a bit rougher than the trend given by `stl`.

d) Next, we calculate the difference DD_{12} and decompose the obtained time series. By the top subfigure of Figure 4, the time series obtained by taking the differences could be stationary.

```
const_d <- diff(diff(const, lag = 12))
const_d_stl <- stl(const_d, s.window = "periodic")

plot(const_d_stl)
```

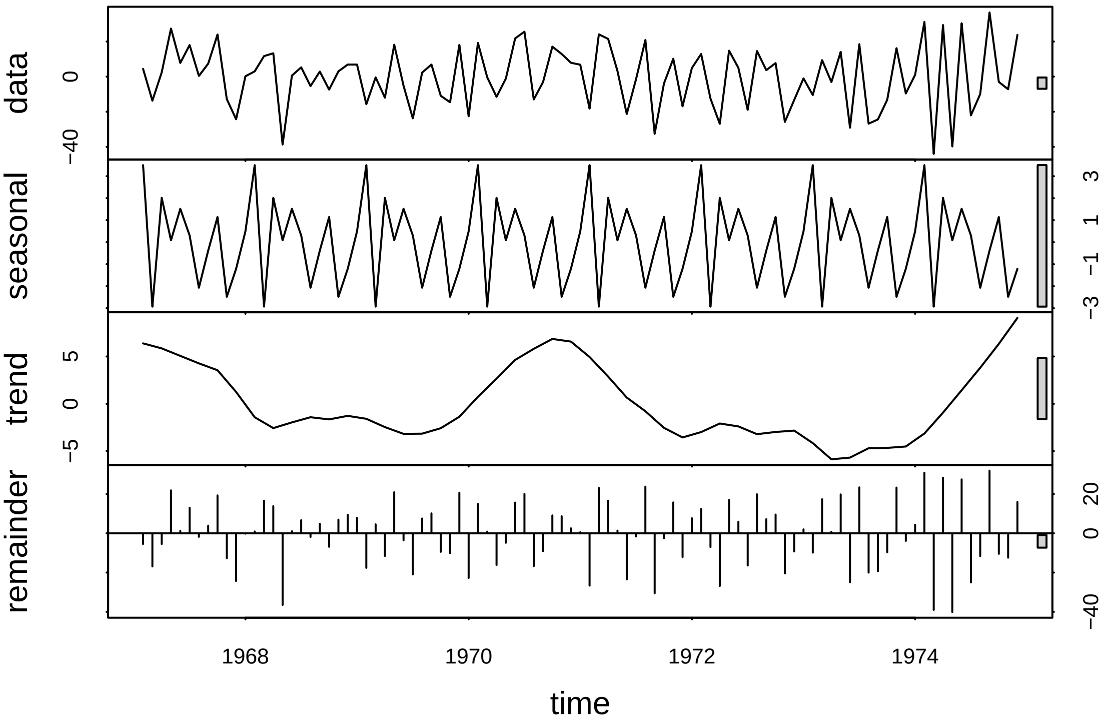


Figure 4: Decomposition of the time series DD_{12}`const`.

6.2

The file `arsimulation.txt` contains realizations at 100 distinct points of time for the following three processes,

$$x_t = \varepsilon_t - \frac{x_{t-1}}{2}, \tag{1}$$

$$y_t = \nu_t - \frac{y_{t-1}}{2}, \tag{2}$$

$$z_t = \eta_t - \frac{z_{t-1}}{2}, \tag{3}$$

where for every $t \in \{2, 3, \dots, 100\}$, ε_t was generated independently from the standard Cauchy distribution, ν_t was generated independently from the Student's t-distribution with 3 degrees of freedom and η_t was generated independently from the Student's t-distribution with 30 degrees of freedom. The starting point for the time series was chosen to be deterministically $x_1 = y_1 = z_1 = 0$.

a) Visualize the three time series.

b) Fit an AR(1) process to each of the three time series by using the function `Arima` from the package `forecast`. Use `Arima` with the arguments `include.mean = FALSE` and `method = "ML"`. Does the AR(1) parameter estimates given by `Arima` match the true parameter values $-1/2$?

c) Bootstrap 95% confidence intervals for the AR(1) model parameters. Use `Arima` with the arguments `include.mean = FALSE` and `method = "ML"`.

d) Assume that for all $s \geq 1$, we have that $x_{t-s} \perp\!\!\!\perp \varepsilon_t$, $y_{t-s} \perp\!\!\!\perp \nu_t$, and $z_{t-s} \perp\!\!\!\perp \eta_t$, where $\perp\!\!\!\perp$ is used to denote stochastic independence. In addition, assume that the elements of the set $\{\varepsilon_t\}_{t \in \mathbb{Z}}$ are i.i.d., the elements of $\{\nu_t\}_{t\in\mathbb{Z}}$ are i.i.d. and the elements of $\{\eta_t\}_{t\in\mathbb{Z}}$ are i.i.d. Under these assumptions, which of the theoretical processes (1)–(3) are weakly stationary?

Solution

a) First, we read the data and visualize all three time series.

```
data <- read.table("data/arsimulation.txt", header = TRUE, row.names = 1)
data <- ts(data)
plot(data, yax.flip = TRUE)
```

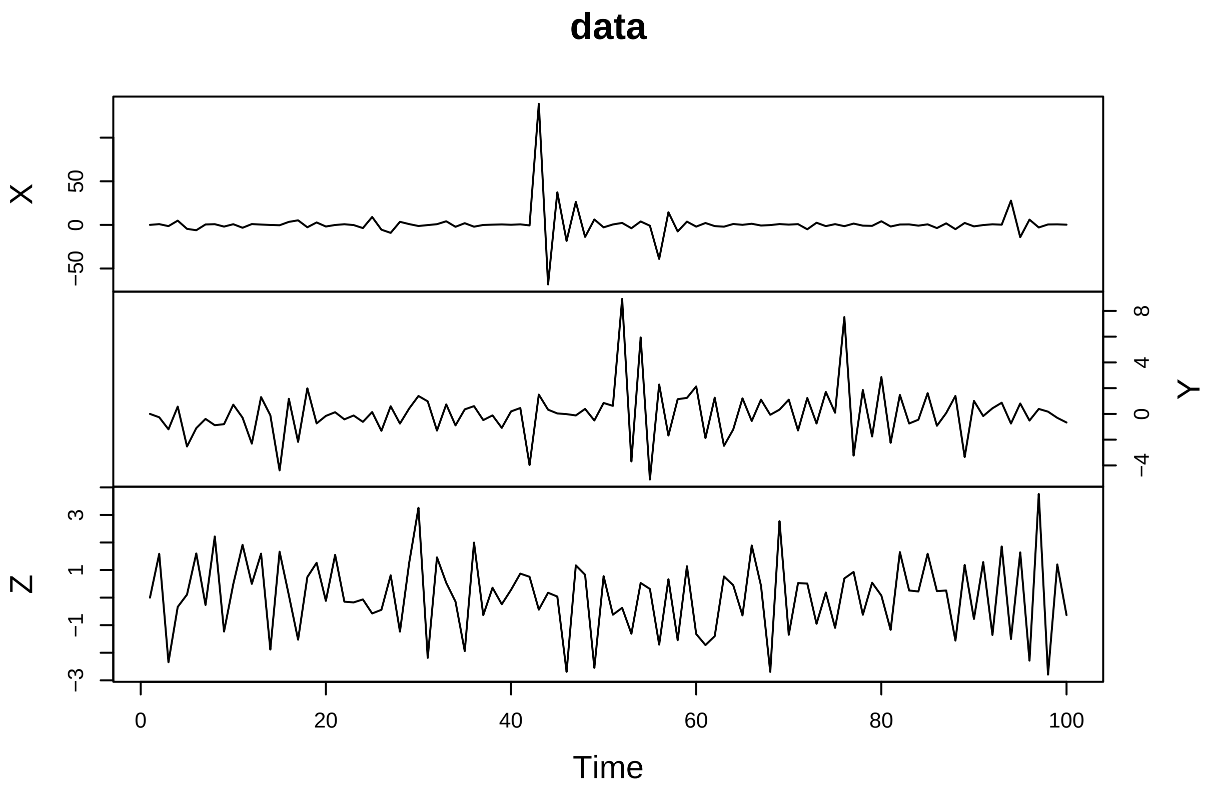


Figure 5: Realizations of all three stochastic processes (1)–(3).

In Figure 5, there is a visible peak for time series (1) at $t = 43$. Note that, the standard Cauchy distribution has considerably heavier tails when compared to, e.g., normal distribution.

b) Next, we fit AR(1) processes to all three time series.

```
library(forecast)
fit_x <- Arima(data[, 1], order = c(1, 0, 0), include.mean = FALSE,
               method = "ML")
fit_y <- Arima(data[, 2], order = c(1, 0, 0), include.mean = FALSE,
               method = "ML")
fit_z <- Arima(data[, 3], order = c(1, 0, 0), include.mean = FALSE,
               method = "ML")

coefs <- c(fit_x$coef, fit_y$coef, fit_z$coef)
```

```
names(coefs) <- c("x", "y", "z")
round(coefs, 3)
```

```
##      x      y      z
## -0.499 -0.494 -0.492
```

The AR(1) parameters estimated from the first, second and third time series are approximately -0.499, -0.494 and -0.492, respectively. Hereby, the estimates are all relatively close to the true parameter value $-1/2$.

c) In the context of time series we cannot choose bootstrap samples by just simply sampling with replacement, since this would break the dependence structure of the time series. Alternatively, we use a variant of *block bootstrap* where bootsrap samples are constructed by choosing at least w number of consecutive observations. In theory the window size w can be 2 or larger. However, in order to have stable ML-estimation procedure we set minimum window size to 10.

Let us make a function for performing block bootstrap.

```
#' Create block bootstrap confidence interval
#'
#' @param series Univariate time series object of class ts.
#' @param m Number of bootstrap samples.
#' @param w Minimum number of consecutive observations in a bootstrap sample.
#' @param alpha Calculate (1-alpha) level confidence interval.
#'
#' @return A vector of length two, giving lower and upper bounds of the
#'   confidence interval.
bootstrap <- function(series, m = 5000, w = 10, alpha = 0.05) {
  n <- length(series)
  boot <- rep(NA, m)

  for (i in 1:m) {
    # Simulate window start and end points. Window size must be at least w.
    start <- 0
    end <- 0
    while (end - start < w - 1) {
      se <- sample(1:n, 2, replace = FALSE)
      start <- min(se)
      end <- max(se)
    }

    # Estimate parameter from bootstrap sample
    boot[i] <- Arima(series[start:end], order = c(1, 0, 0),
                     include.mean = FALSE, method = "ML")$coef[1]
  }

  # Construct confidence interval
  quantile(boot, c(alpha / 2, 1 - alpha / 2))
}
```

Next, compute confidence intervals.

```
confint_x <- bootstrap(data[, 1])
confint_y <- bootstrap(data[, 2])
confint_z <- bootstrap(data[, 3])
confint_x
```

```
##      2.5%     97.5%
## -0.933286 -0.157096
```

```
confint_y
```

```
##       2.5%      97.5%
## -0.7534900 -0.2361305
```

```
confint_z
```

```
##       2.5%      97.5%
## -0.7822626 -0.2072821
```

The requested confidence intervals estimated from the first, second and third time series are approximately $[-0.93, -0.16]$, $[-0.75, -0.24]$ and $[-0.78, -0.21]$, respectively.

d) The standard Cauchy distribution is exactly the Student's t-distribution with 1 degree of freedom. Student's t-distribution with k degrees of freedom has $k-1$ theoretical moments. As the variance of x_t and the variance of ε_t are undefined, Process (1) is not weakly stationary. Recall that, weak stationarity contains the assumption that the variance is time invariant and *finite*. For more information about ARMA processes with infinite variance, see Chapter 13.3 of (Brockwell and Davis 2009).

Consequently, the AR(1) Processes (2) and (3) have finite variances. Under the assumptions of this exercise and by previous theoretical exercises, we have that Processes (2) and (3) are weakly stationary.

6.3

a) Fit ARMA model to the time series in file `fracsim.txt`.

b) Could there be better fits than ARMA model?

Solution

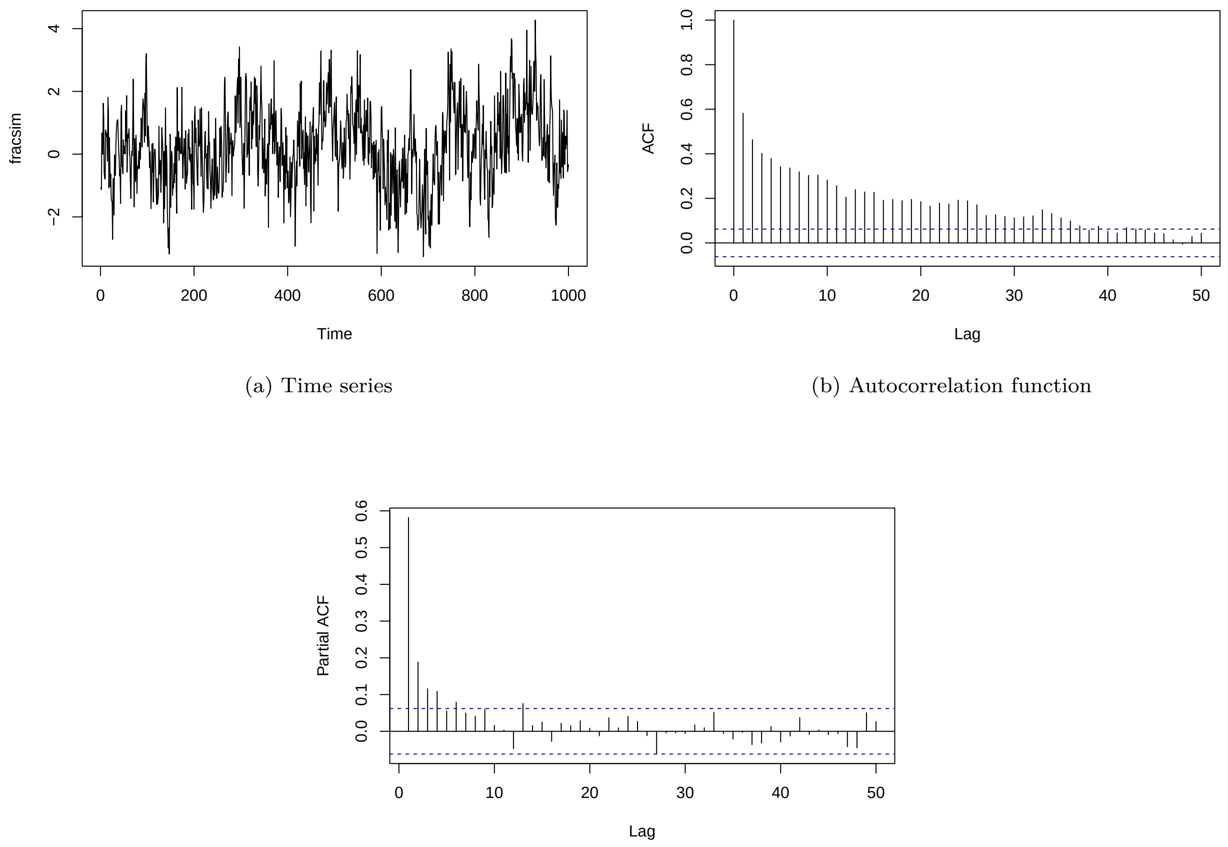
a) First, we read the data. All in all, time series `fracsim` contains 1000 observations.

```
fracsim <- read.csv("data/fracsim.txt")
fracsim <- ts(fracsim[[1]])
length(fracsim)
```

```
## [1] 1000
```

Next, we plot both the time series and ACF/PACF plots.

```
plot(fracsim)
acf(fracsim, lag.max = 50, main = "")
pacf(fracsim, lag.max = 50, main = "")
```



(a) Time series

(b) Autocorrelation function

(c) Partial autocorrelation function

Figure 6: Time series plot, ACF plot and PACF plot of the time series `frac_series`.

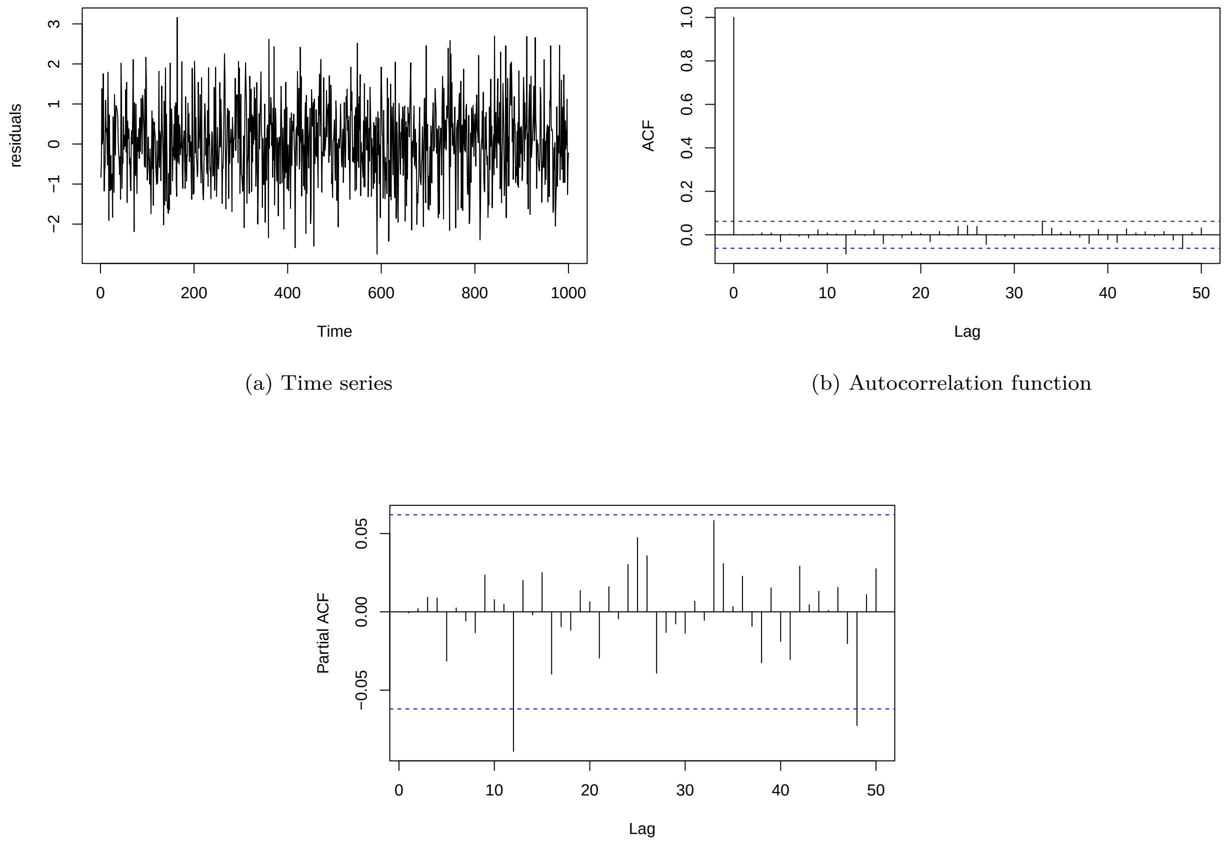
By Figure 6a) time series could be stationary, since there is no trend or seasonality. However, interestingly ACF decays to zero quite slowly. Nevertheless, let us use `auto.arima` to fit an ARMA process to time series `frac_series`.

```
model_auto <- auto.arima(fracsim, d = 0, D = 0, max.p = Inf, max.q = Inf,
                         max.P = 0, max.Q = 0, max.order = Inf)
model_auto
```

```
## Series: fracsim
## ARIMA(1,0,3) with zero mean
##
## Coefficients:
##          ar1      ma1      ma2      ma3
##       0.9537  -0.5357  -0.1237  -0.0669
## s.e.  0.0162   0.0360   0.0368   0.0334
##
## sigma^2 = 0.9822:  log likelihood = -1408.31
## AIC=2826.62   AICc=2826.68   BIC=2851.16
```

Next let us study if residuals are white noise with ACF/PACF plots.

```r
plot(model_auto$residuals, type = "l", ylab = "residuals")
acf(model_auto$residuals, lag.max = 50, main = "")
pacf(model_auto$residuals, lag.max = 50, main = "")
```



(a) Time series

(b) Autocorrelation function

(c) Partial autocorrelation function

Figure 7: Time series plot, ACF plot and PACF plot of the residuals of the ARMA$(1, 3)$.

Figure 7 shows that at least at lag 12 there is a significant spike in ACF and PACF. Thus it seems that residuals of the model ARMA$(1, 3)$ are not (quite) white noise.

b) Actually, simulated observations in file `fracsim.txt` are realizations of *fractionally integrated noise* process x_t,

$$D^{0.45}x_t = \varepsilon_t, \quad \text{where } \varepsilon_t \sim \mathrm{WN}(0, \sigma^2).$$

In this case, we chose $d = 0.45$, but generally, one can choose $d \in (-0.5, 0.5)$. Here the fractional difference operator for $d \in (-0.5, 0.5)$ is defined as

$$\mathrm{D}^d = (1 - \mathrm{L})^d = \sum_{j=0}^{\infty} \pi_j \mathrm{L}^j,$$

where

$$\pi_j = \prod_{k=1}^{j} \frac{k - 1 - d}{k}, \quad j \in \{1, 2, ...\}.$$

Generally, one can define fractionally integrated ARMA (ARFIMA) process for $d \in (-0.5, 0.5)$ as

$$\phi_p(L) D^d x_t = \theta_q(L) \varepsilon_t.$$

It turns out that autocorrelation function of an ARFIMA process decays a lot slower than autocorrelation of an ARIMA process. Thus, ARFIMA processes are often called *long memory* processes and, on the other hand, ARIMA processes are *short memory* processes. For a more rigorous treatment of long memory processes, see Chapter 13.2 of (Brockwell and Davis 2009).

In theory, one can fit an ARIMA model to a long memory process but this often leads to large orders p and q of lag polynomials $\phi_p(L)$ and $\theta_q(L)$, respectively. More parsimonious model is often achieved by fitting a ARFIMA model to long memory processes.

Package `fracdiff` provides tools for simulating and fitting ARFIMA processes. For example we can fit a integrated noise process to simulated realizations `frac_series` with the function `fracdiff`.

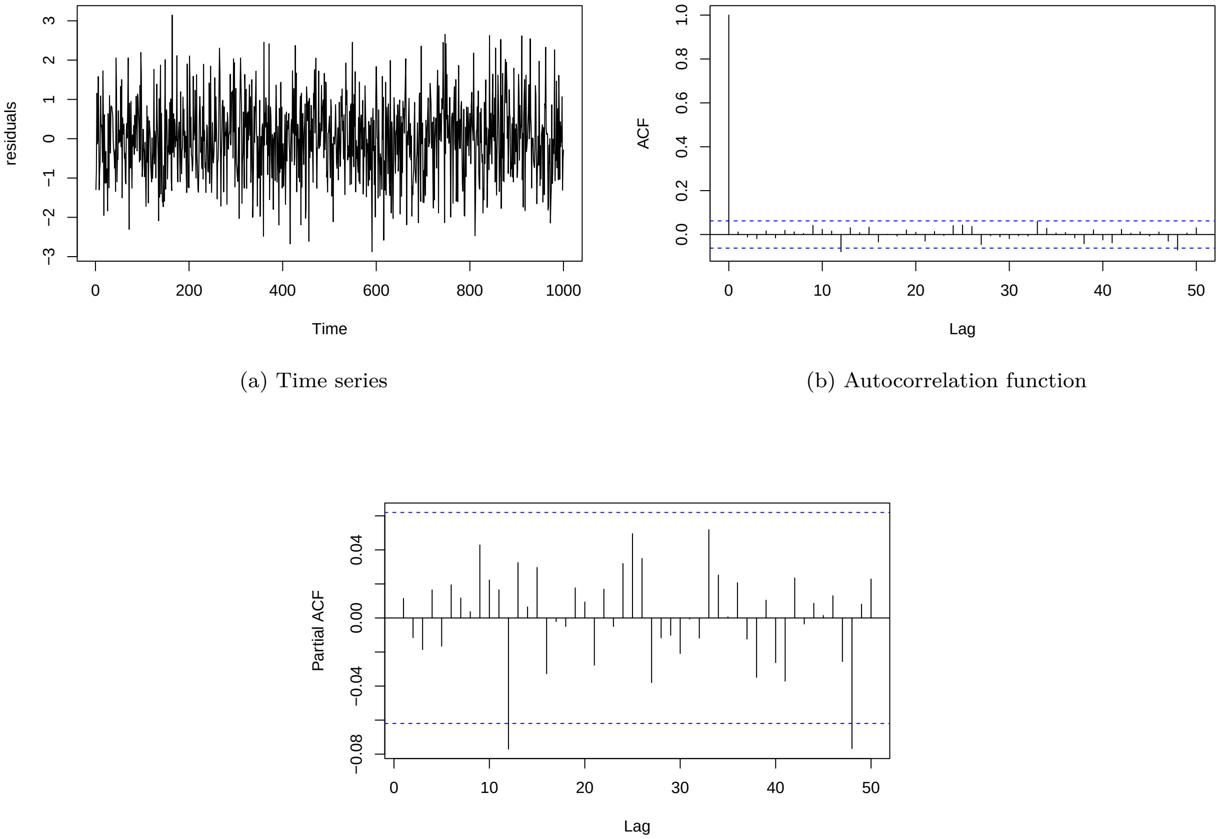
```
library(fracdiff)
model_frac <- fracdiff(fracsim, nar = 0, nma = 0)
model_frac$d
```

```
## [1] 0.409231
```

Estimate $\hat{d} \approx 0.41$ is quite close to the true value of the order of differencing $d = 0.45$.

By Figure 8, residuals of the ARFIMA$(0, 1, 0)$ model behave similarly as the residuals of the ARMA$(1, 3)$ model.

```
plot(model_frac$residuals, type = "l", ylab = "residuals")
acf(model_frac$residuals, lag.max = 50, main = "")
pacf(model_frac$residuals, lag.max = 50, main = "")
```



(a) Time series

(b) Autocorrelation function

(c) Partial autocorrelation function

Figure 8: Time series plot, ACF plot and PACF plot of the residuals of the ARFIMA$(0, 0.45, 0)$ model.

References

Brockwell, Peter J, and Richard A Davis. 2009. *Time Series: Theory and Methods.* Springer science & business media.