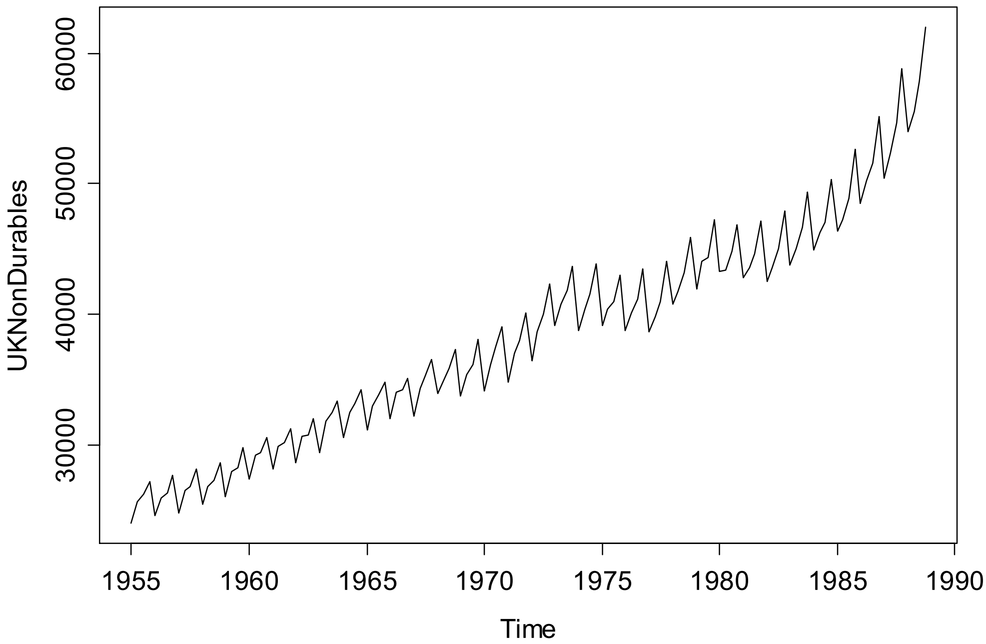
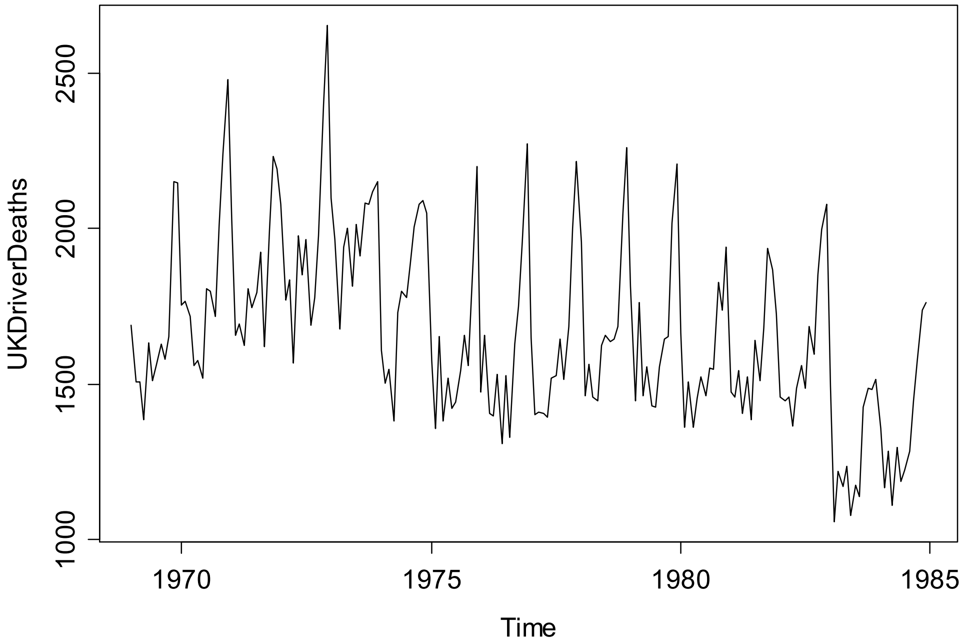
```
# TS
library("AER")
data("UKNonDurables")
# quarterly consumption of non-durables in the United Kingdom
plot(UKNonDurables)
```

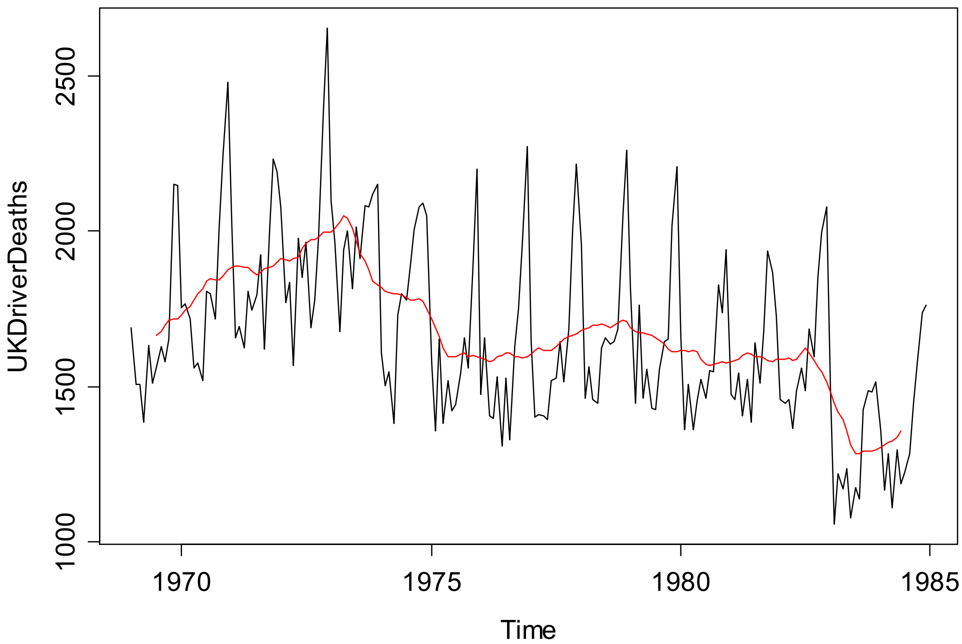



```
tsp(UKNonDurables) #returns the tsp attribute
# working with irregular series (e.g., with many financial time series).
# Consequently, various implementations for irregular time series have emerged
# in contributed R packages, the most flexible of which is “zoo”, provided by
# the zoo package [153]

[1] 1955.00 1988.75    4.00

str(UKNonDurables)

Time-Series [1:136] from 1955 to 1989: 24030 25620 26209 27167 24620 25972 26285
27659 24780 26519 ...
time(UKNonDurables)[1:10]

[1] 1955.00 1955.25 1955.50 1955.75 1956.00 1956.25 1956.50 1956.75 1957.00
1957.25

window(UKNonDurables, end = c(1956, 4)) #podzbiór
      Qtr1  Qtr2  Qtr3  Qtr4
1955 24030 25620 26209 27167
1956 24620 25972 26285 27659

(Linear) filtering [154]
```

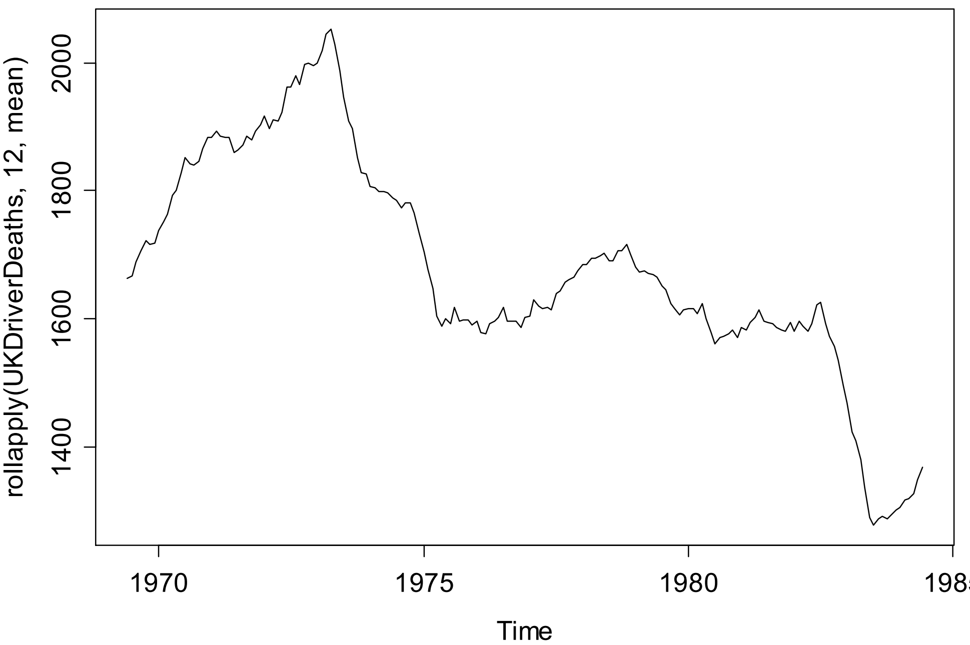
```
data("UKDriverDeaths")
on car drivers
killed or seriously injured in the United Kingdom from 1969(1) through
1984(12). These are also known as the “seatbelt data”, as they were used
by Harvey and Durbin (1986) for evaluating the effectiveness of compulsory
wearing of seatbelts introduced on 1983-01-31
plot(UKDriverDeaths)
```



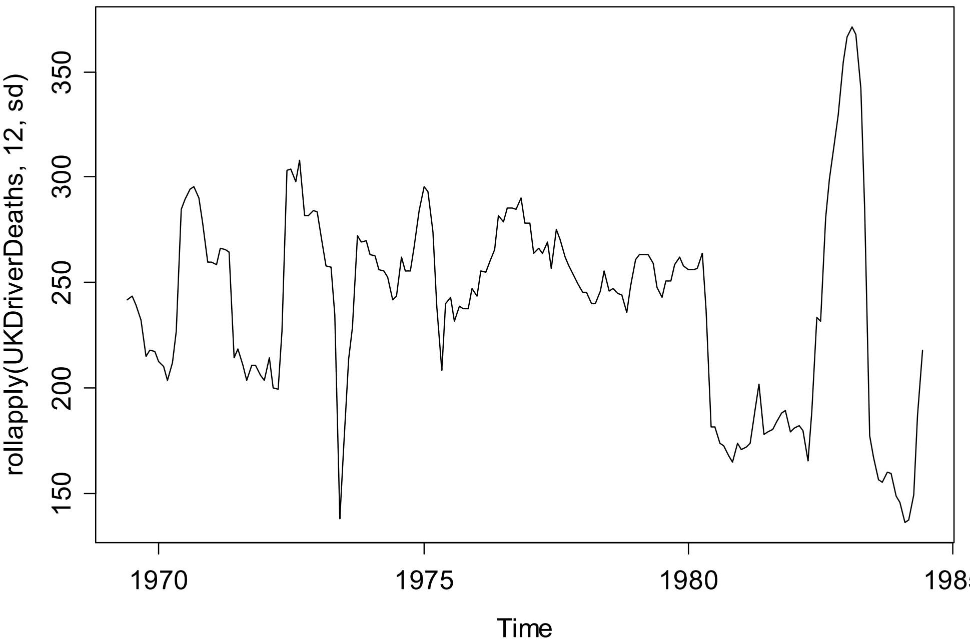

```
lines(filter(UKDriverDeaths, c(1/2, rep(1, 11), 1/2)/12),col = 2)
```



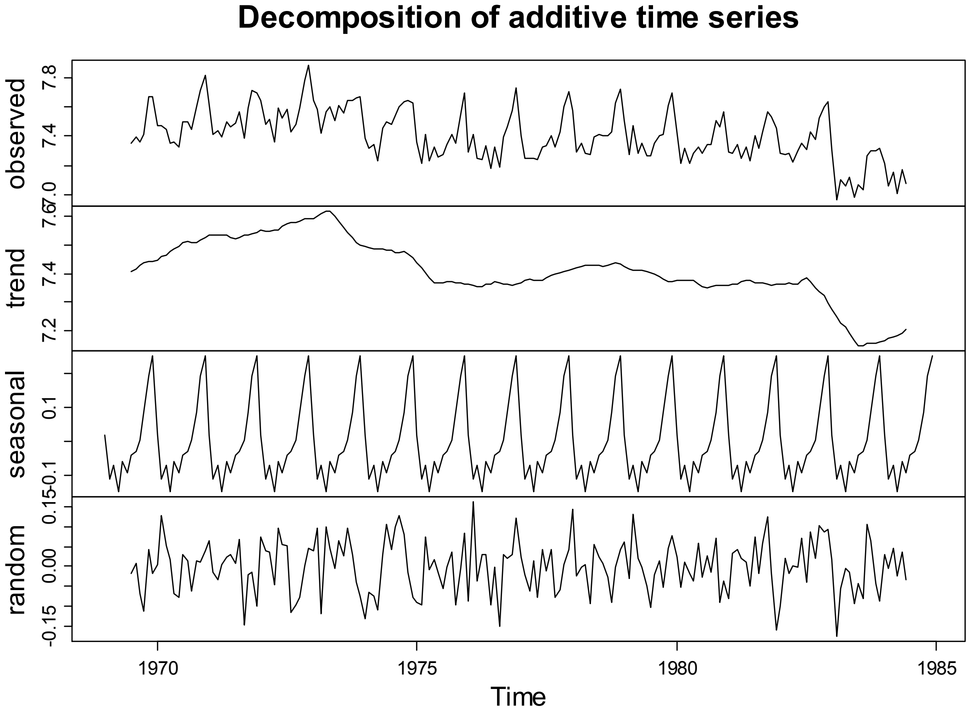
```
plot(rollapply(UKDriverDeaths,12, mean)) # ruchome okno
```



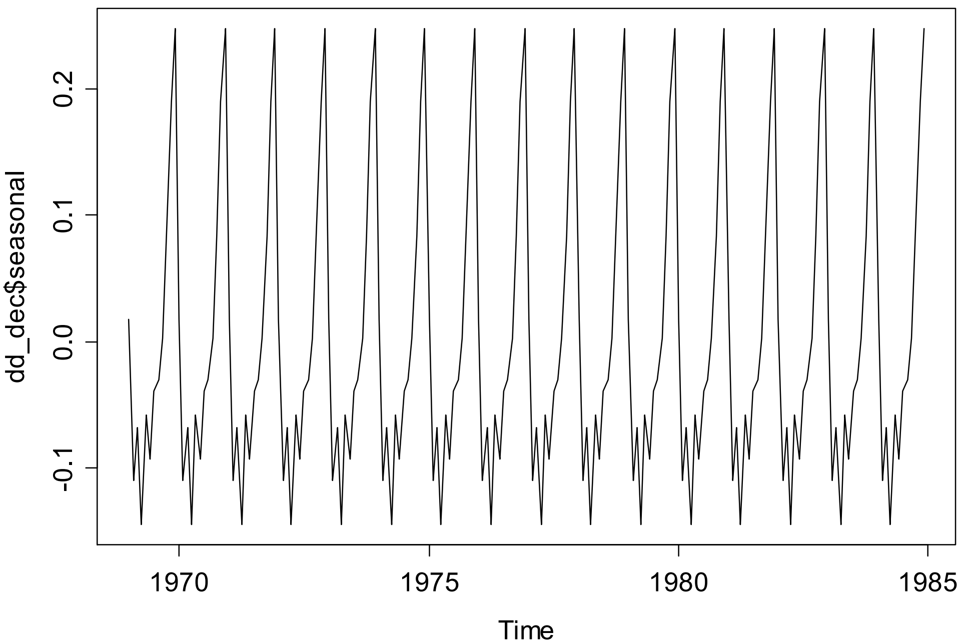

```
plot(rollapply(UKDriverDeaths, 12, sd))
```



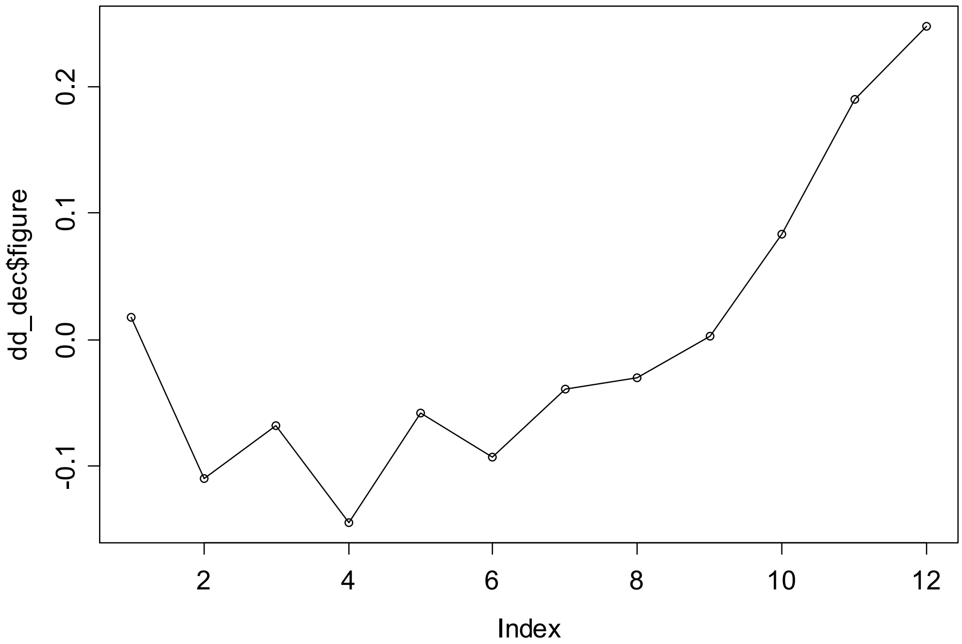
```
# Decomposition [155]
dd_dec <- decompose(log(UKDriverDeaths))
# simple symmetric filter as illustrated above for extracting the trend
# and derive the seasonal component by averaging the trend-adjusted observations
# from corresponding periods
# c("seasonal", "trend", "random",   "figure", "type"),
plot(dd_dec)
```



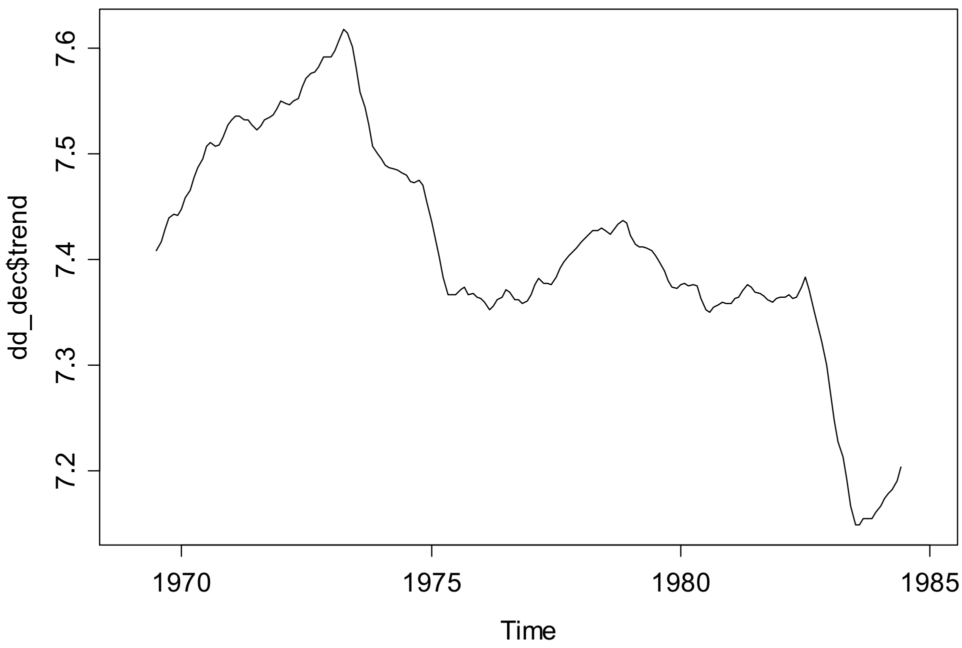

```
plot(dd_dec$seasonal)
```



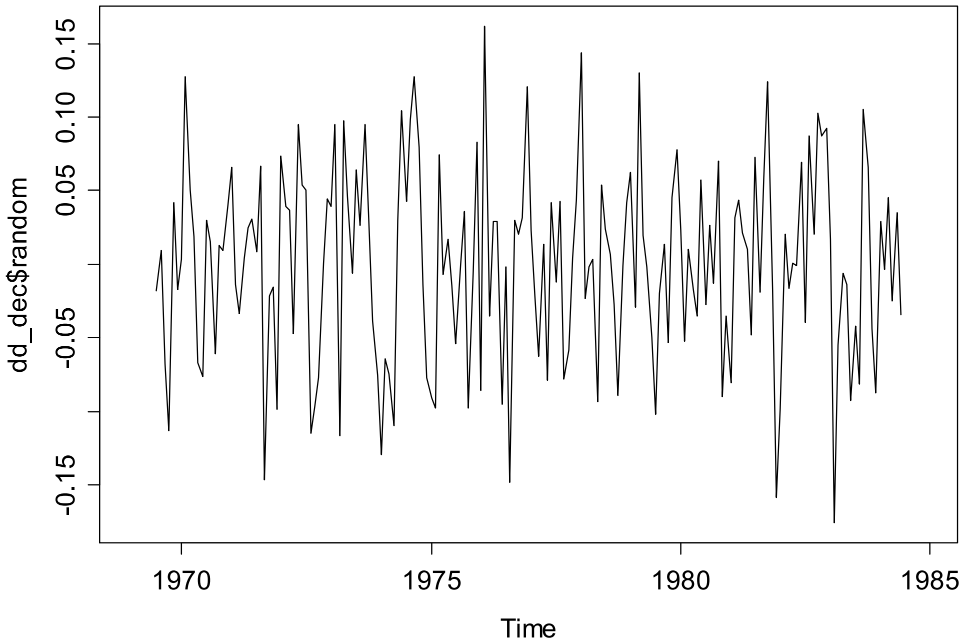
```
plot(lines(dd_dec$figure))
```



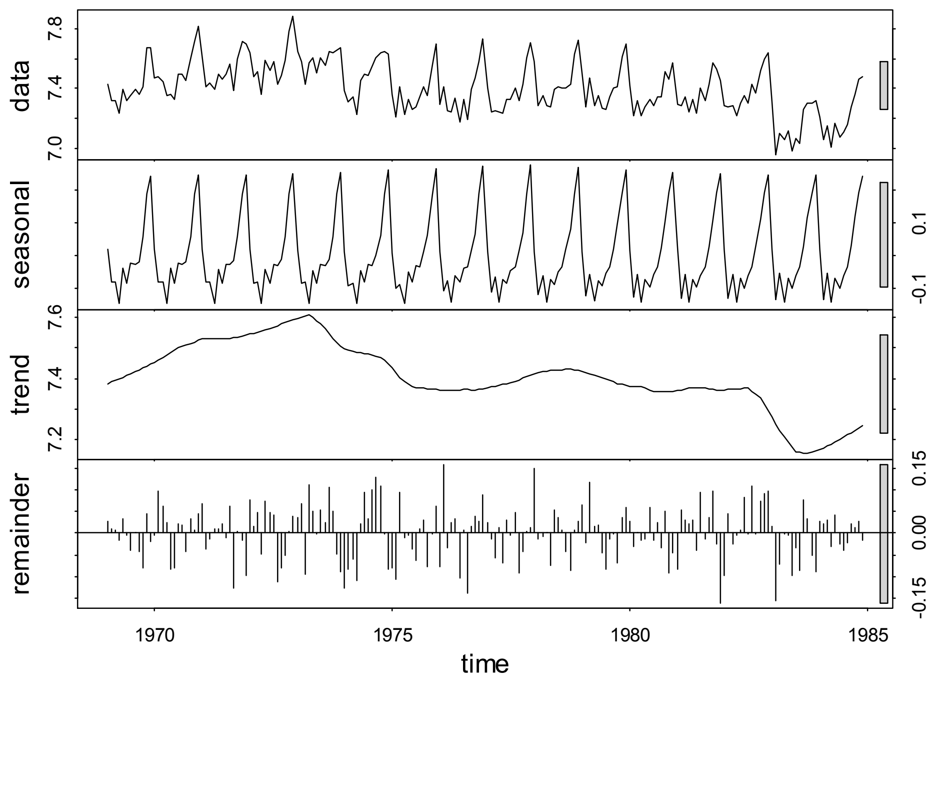

```
plot(dd_dec$trend)
```



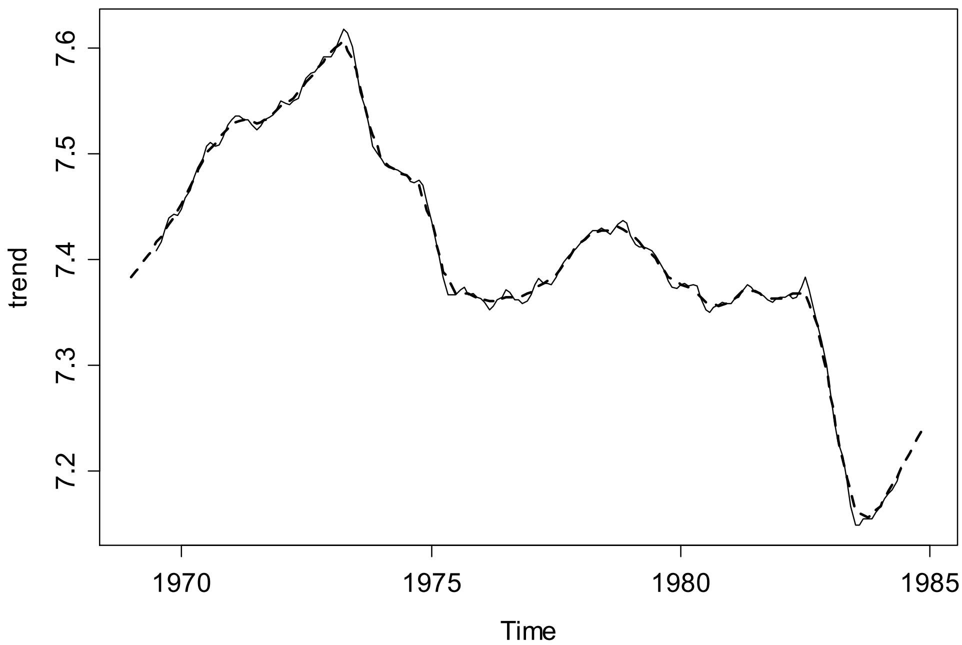
```
plot(dd_dec$random)
```




```
dd_stl <- stl(log(UKDriverDeaths), s.window = 13)
# iteratively finds the seasonal and trend
# components by loess smoothing of the observations in moving data windows
# of a certain size
plot(dd_stl)
```



```
plot(dd_dec$trend, ylab = "trend")
lines(dd_stl$time.series[,"trend"], lty = 2, lwd = 2)
```



stl() yielding a smoother curve.

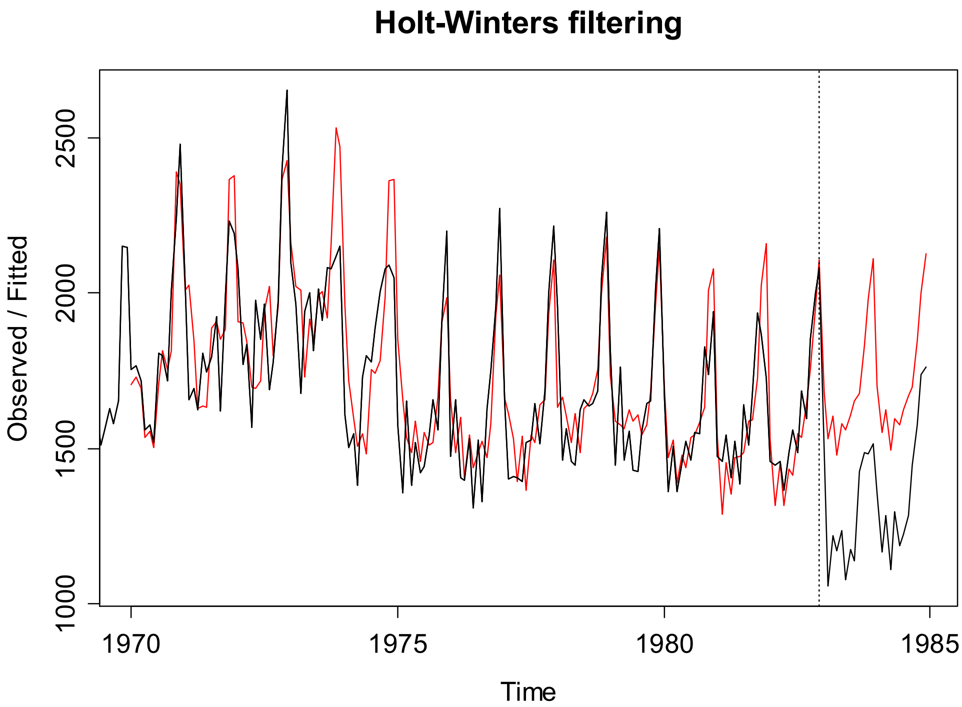
```
# Exponential smoothing
dd_past <- window(UKDriverDeaths, end = c(1982, 12))
dd_hw <- HoltWinters(dd_past)
```

Unknown parameters are determined by minimizing the squared prediction error.

The additive Holt-Winters prediction function (for time series with period length p) is

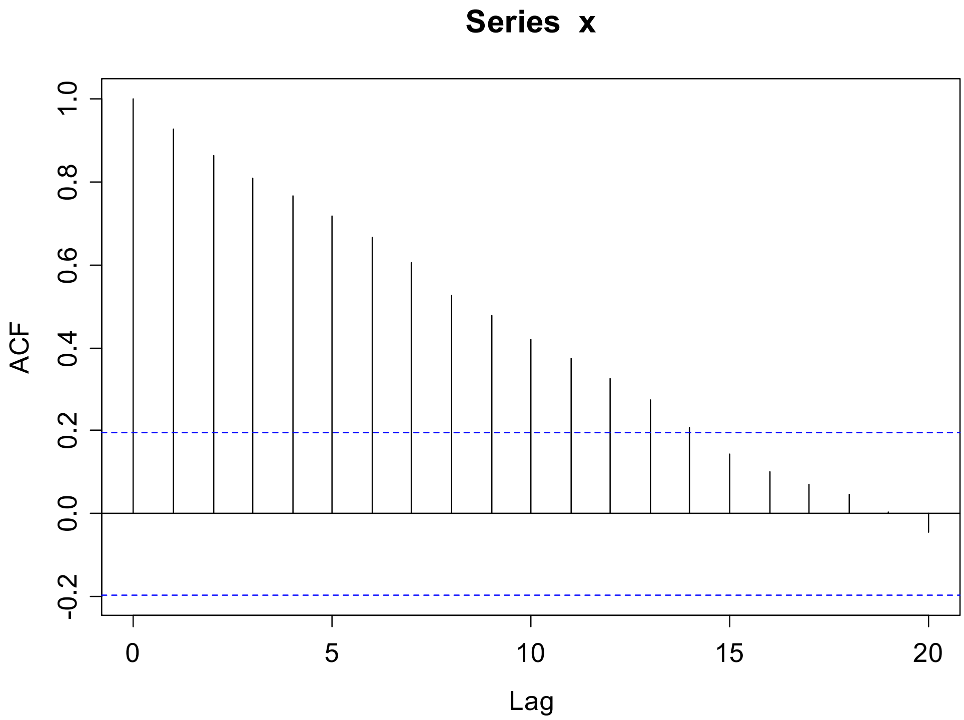
```
Yhat[t+h] = a[t] + h * b[t] + s[t + p + 1 + (h - 1) mod p],
where a[t], b[t] and s[t] are given by
a[t] = α (Y[t] - s[t-p]) + (1-α) (a[t-1] + b[t-1])
b[t] = β (a[t] - a[t-1]) + (1-β) b[t-1]
s[t] = γ (Y[t] - a[t]) + (1-γ) s[t-p]
```

```
dd_pred <- predict(dd_hw, n.ahead = 24)
plot(dd_hw, dd_pred, ylim = range(UKDriverDeaths))
lines(UKDriverDeaths)
```

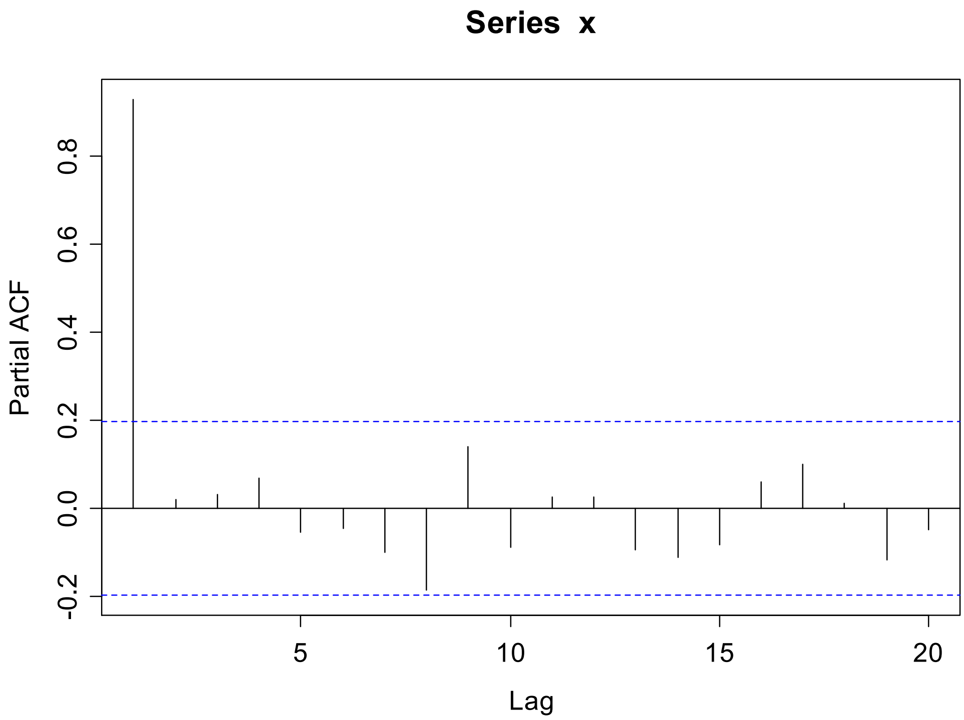


*

```
# Classical Model-Based Analysis
set.seed(1234)
x <- filter(rnorm(100), 0.9, method = "recursive") #AR(1)
acf(x)
```



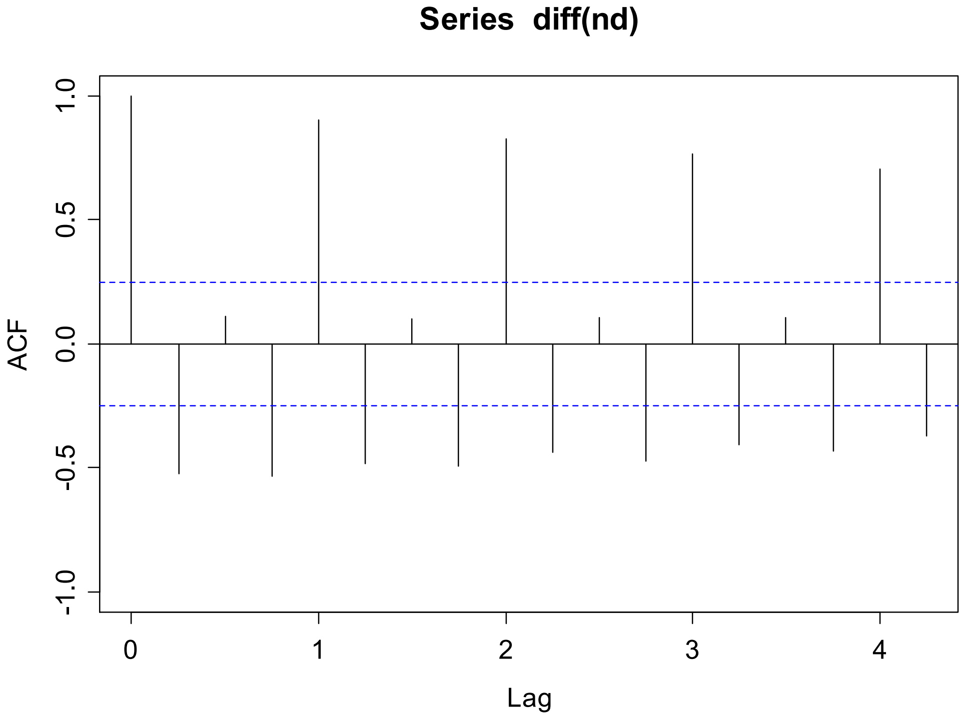

```
pacf(x)
```



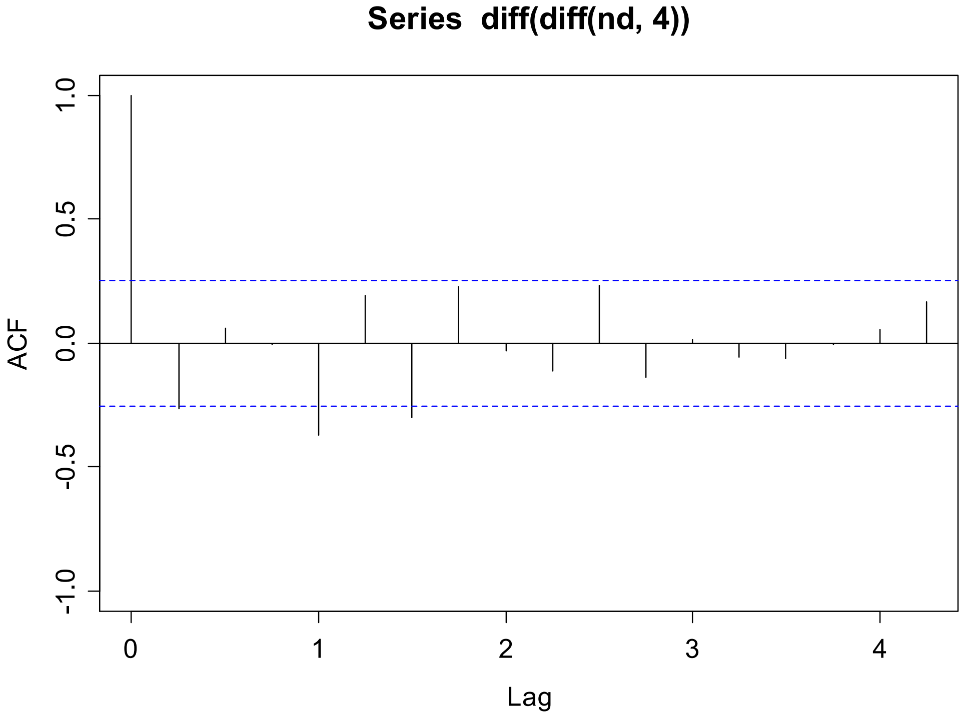

```
ar(x)
Call:
ar(x = x)

Coefficients:
     1
0.9279

Order selected 1  sigma^2 estimated as  1.286
```

By default, ar() fits AR models up to lag p = 10 log (n) and selects the minimum AIC model

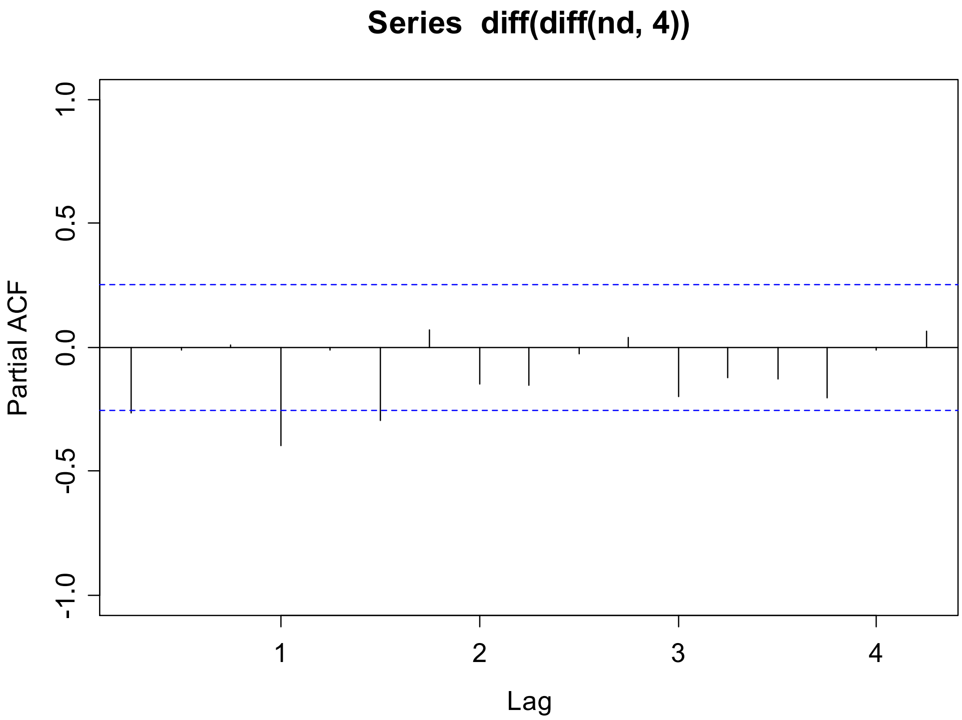
```
ar(x, method="burg")
Call:
ar(x = x, method = "burg")

Coefficients:
     1
0.9457

Order selected 1  sigma^2 estimated as  0.9576

nd <- window(log(UKNonDurables), end = c(1970, 4))
acf(diff(nd), ylim = c(-1, 1))
```




```
acf(diff(diff(nd, 4)), ylim = c(-1, 1))
```



```
pacf(diff(diff(nd, 4)), ylim = c(-1, 1))
```



*

ment. As the model space is much more complex than in the AR(p) case, where only the order p has to be chosen, base R does not offer an automatic model selection method for general ARIMA models based on information criteria. Therefore, we use the preliminary results from the exploratory analysis above and R's general tools to set up a model search for an appropriate SARIMA$(p, d, q)(P, D, Q)_4$ model,

$$\Phi(L^4)\phi(L)(1 - L^4)^D(1 - L)^d y_t = \theta(L)\Theta(L^4)\varepsilon_t, \qquad (6.2)$$

which amends the standard ARIMA model (6.1) by additional polynomials operating on the seasonal frequency.

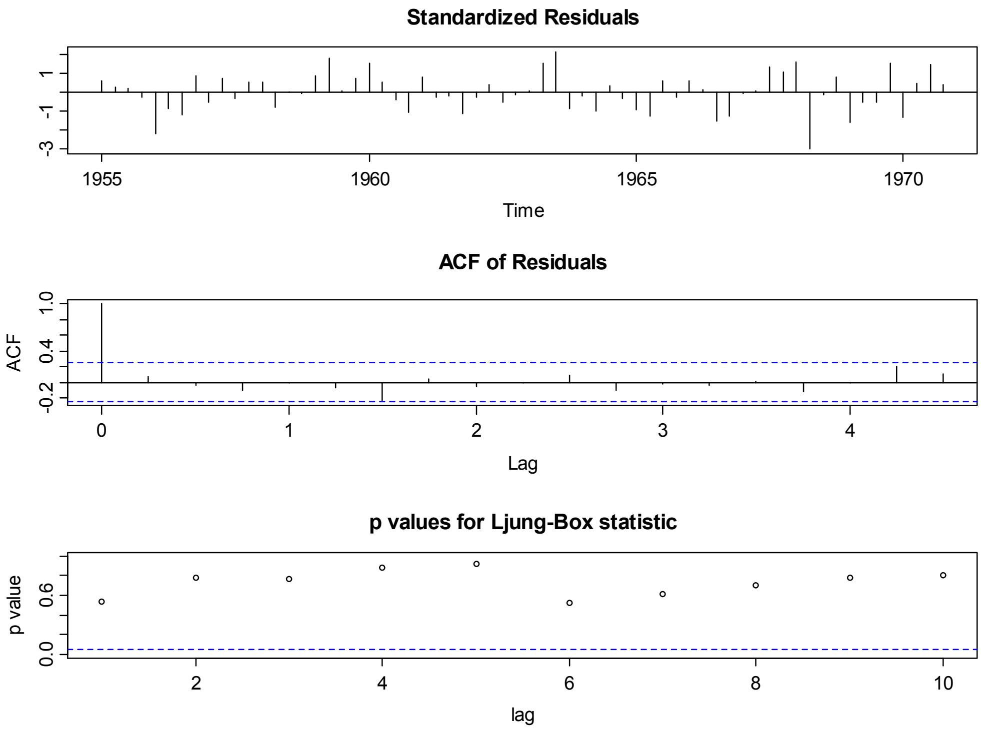
The graphical analysis clearly suggests double differencing of the original series ($d = 1$, $D = 1$), some AR and MA effects (we allow $p = 0, 1, 2$ and $q = 0, 1, 2$), and low-order seasonal AR and MA parts (we use $P = 0, 1$ and $Q = 0, 1$), giving a total of 36 parameter combinations to consider. Of course, higher values for p, q, P, and Q could also be assessed. We refrain from doing so

```
nd_pars <- expand.grid(ar = 0:2, diff = 1, ma = 0:2,
                          sar = 0:1, sdiff = 1, sma = 0:1)
# Create a data frame from all combinations of the supplied vectors or factors.
nd_aic <- rep(0, nrow(nd_pars))
for(i in seq(along = nd_aic)) nd_aic[i] <- AIC(arima(nd,
        unlist(nd_pars[i, 1:3]), unlist(nd_pars[i, 4:6])),
        k = log(length(nd)))
nd_pars[which.min(nd_aic),]
   ar diff ma sar sdiff sma
22  0    1  1   0     1   1
```

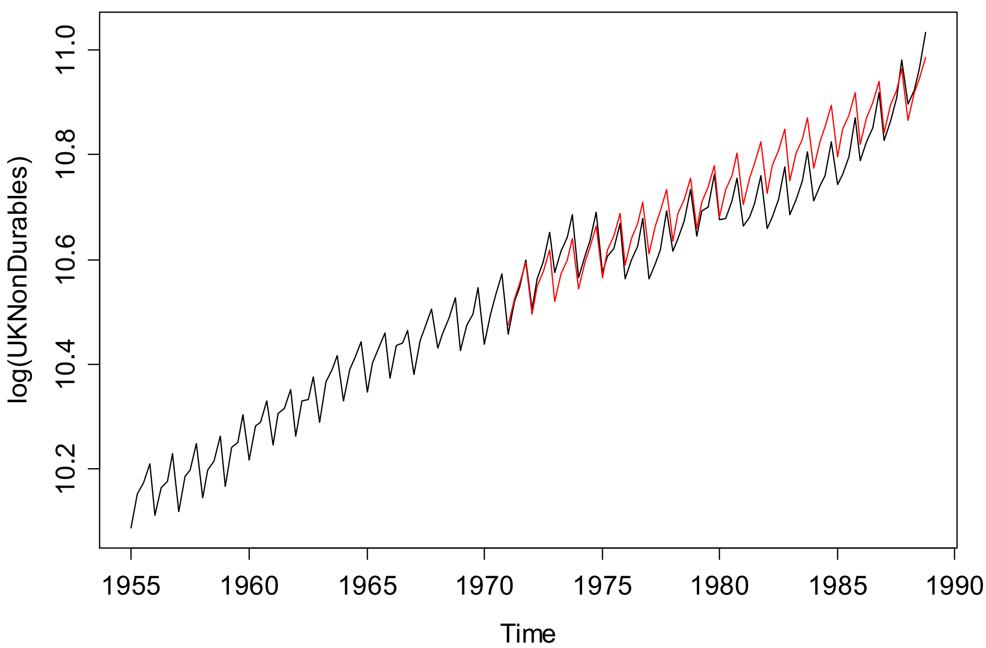
These computations reveal that a SARIMA(0, 1, 1)(0, 1, 1) model is best in terms of BIC, conforming well with the exploratory analysis. This model is also amously known as the airline model due to its application to a series of airline passengers in the classical text by Box and Jenkins (1970). It is refitted to nd via

```
nd_arima <- arima(nd, order = c(0,1,1), seasonal = c(0,1,1))
```

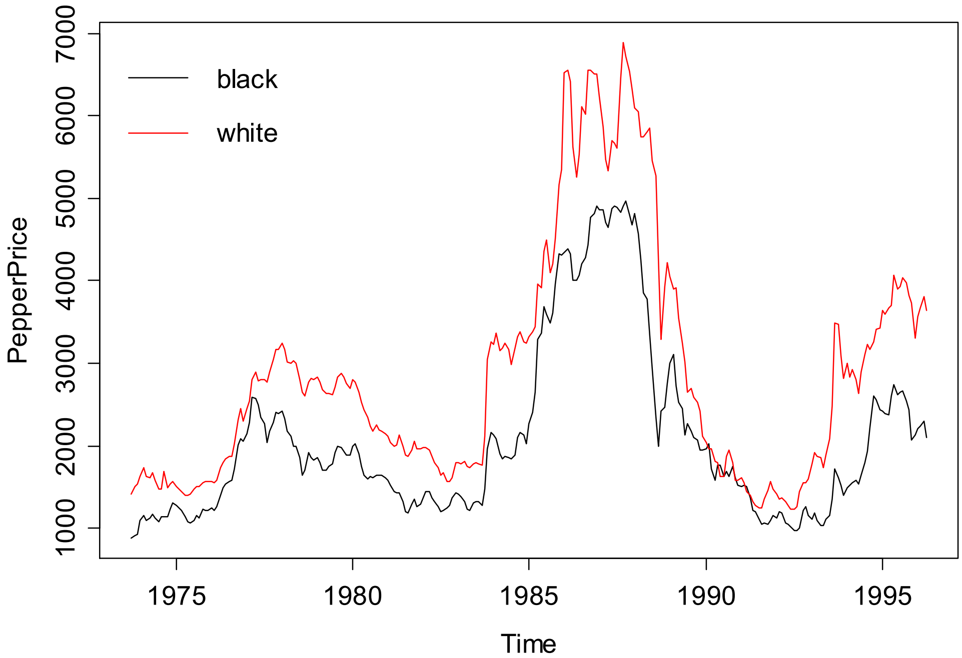
```
nd_arima

Call:
arima(x = nd, order = c(0, 1, 1), seasonal = c(0, 1, 1))

Coefficients:
          ma1     sma1
       -0.353  -0.5828
s.e.    0.143   0.1382

sigma^2 estimated as 9.649e-05:  log likelihood = 188.14,  aic = -370.2
tsdiag(nd_arima)
```



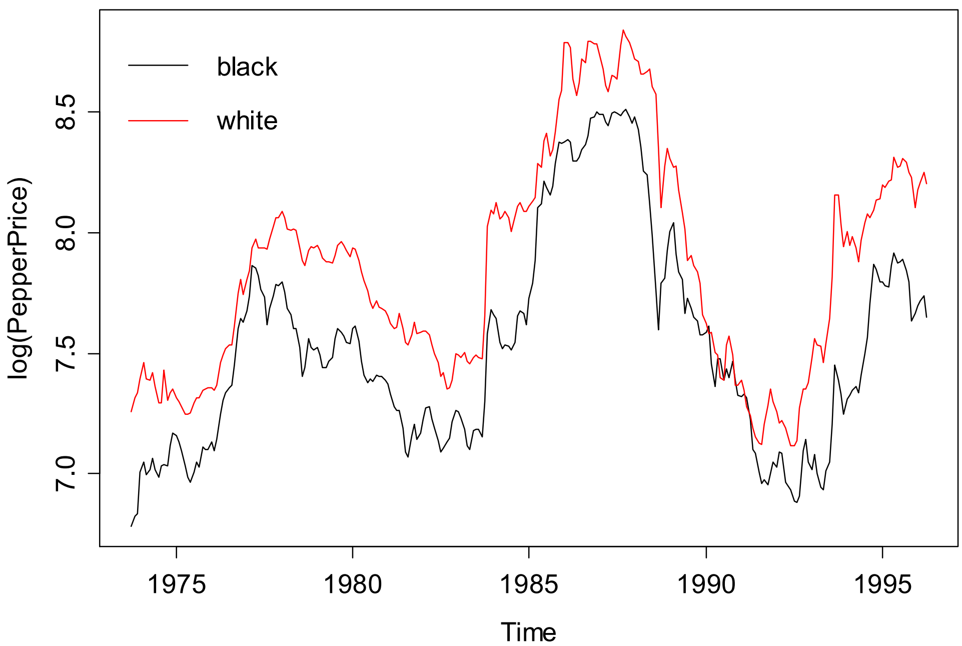

```
nd_pred <- predict(nd_arima, n.ahead = 18 * 4)
plot(log(UKNonDurables))
lines(nd_pred$pred, col = 2)
```



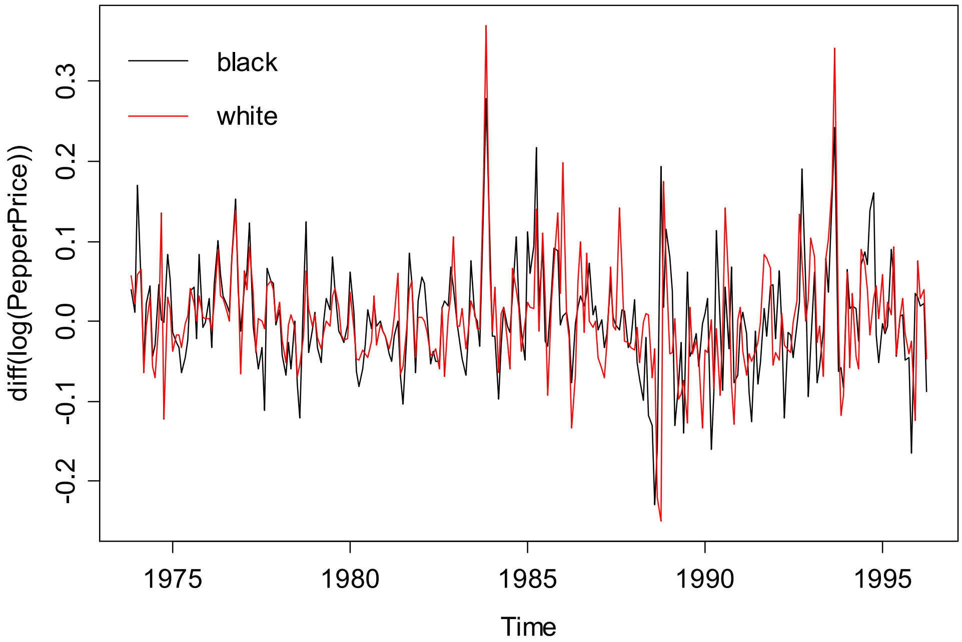
```
# Stationarity, Unit Roots, and Cointegration
data("PepperPrice")
# A monthly multiple time series from 1973(10) to 1996(4) with 2 variables.
# black
# spot price for black pepper,
# white
# spot price for white pepper.
plot(PepperPrice, plot.type = "single", col = 1:2)
legend("topleft", c("black", "white"), bty = "n",
          col = 1:2, lty = rep(1,2))
```




```
plot(log(PepperPrice), plot.type = "single", col = 1:2)
legend("topleft", c("black", "white"), bty = "n",
       col = 1:2, lty = rep(1,2))
```



```
plot(diff(log(PepperPrice)), plot.type = "single", col = 1:2)
legend("topleft", c("black", "white"), bty = "n",
       col = 1:2, lty = rep(1,2))
```




```
library("tseries")
adf.test(log(PepperPrice[, "white"]))

        Augmented Dickey-Fuller Test

data:  log(PepperPrice[, "white"])
Dickey-Fuller = -1.744, Lag order = 6, p-value = 0.6838
alternative hypothesis: stationary
adf.test(diff(log(PepperPrice[, "white"])))
        Augmented Dickey-Fuller Test

data:  diff(log(PepperPrice[, "white"]))
Dickey-Fuller = -5.336, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(diff(log(PepperPrice[, "white"]))) :
  p-value smaller than printed p-value
 Zwroty są stacjonarne
```

Kwiatkowski *et al.* (1992) proceed by testing for the presence of a random walk component r_t in the regression

$$y_t = d_t + r_t + \varepsilon_t,$$

where d_t denotes a deterministic component and ε_t is a stationary—more precisely, $I(0)$—error process. This test is also available in the function `kpss.test()` in the package **tseries**. The deterministic component is either a constant or a linear time trend, the former being the default. Setting the argument `null = "Trend"` yields the second version. Here, we obtain

```
kpss.test(log(PepperPrice[, "white"]))
        KPSS Test for Level Stationarity

data:  log(PepperPrice[, "white"])
KPSS Level = 0.9129, Truncation lag parameter = 3, p-value = 0.01

Warning message:
In kpss.test(log(PepperPrice[, "white"])) :
  p-value smaller than printed p-value

kpss.test(diff(log(PepperPrice[, "white"])))
        KPSS Test for Level Stationarity

data:  diff(log(PepperPrice[, "white"]))
KPSS Level = 0.1336, Truncation lag parameter = 3, p-value = 0.1

Warning message:
In kpss.test(diff(log(PepperPrice[, "white"]))) :
  p-value greater than printed p-value
```

KOINTEGRACJA
A simple method to test for cointegration is the two-step method proposed by Engle and Granger (1987). It regresses one series on the other and performs a unit root test on the residuals. This test, often named after Phillips and Ouliaris (1990), who provided the asymptotic theory, is available in the function po.test() from the package tseries

```
po.test(log(PepperPrice))

        Phillips-Ouliaris Cointegration Test

data:  log(PepperPrice)
Phillips-Ouliaris demeaned = -24.0987, Truncation lag parameter = 2, p-value =
0.02404
po.test(log(PepperPrice[,2:1]))
        Phillips-Ouliaris Cointegration Test

data:  log(PepperPrice[, 2:1])
Phillips-Ouliaris demeaned = -22.6762, Truncation lag parameter = 2, p-value =
0.03354
```

Asymetria (a kointegracja to relacja symetryczna!)

`UKDriverDeaths` series: the log-casualties are regressed on their lags 1 and 12, essentially corresponding to the multiplicative SARIMA$(1,0,0)(1,0,0)_{12}$ model

$$y_t = \beta_1 + \beta_2 y_{t-1} + \beta_3 y_{t-12} + \varepsilon_t, \quad t = 13, \ldots, 192.$$

```
dd <- log(UKDriverDeaths)
dd_dat <- ts.intersect(dd, dd1 = lag(dd, k = -1),
                       dd12 = lag(dd, k = -12))
# intersect: tworzy kilka szeregów czasowych
summary(lm(dd ~ dd1 + dd12, data = dd_dat))
Call:
lm(formula = dd ~ dd1 + dd12, data = dd_dat)

Residuals:
     Min       1Q   Median       3Q      Max
-0.32738 -0.07860  0.01414  0.07284  0.18849

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.42055    0.36327   1.158    0.249
dd1          0.43104    0.05327   8.091 9.10e-14 ***
dd12         0.51120    0.05653   9.043 2.65e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.09938 on 177 degrees of freedom
Multiple R-squared: 0.6766,    Adjusted R-squared: 0.673
F-statistic: 185.2 on 2 and 177 DF,  p-value: < 2.2e-16
```

Uwaga! Nie uwzglednia, że trzy szeregi pochodzą od jednego szeregu !

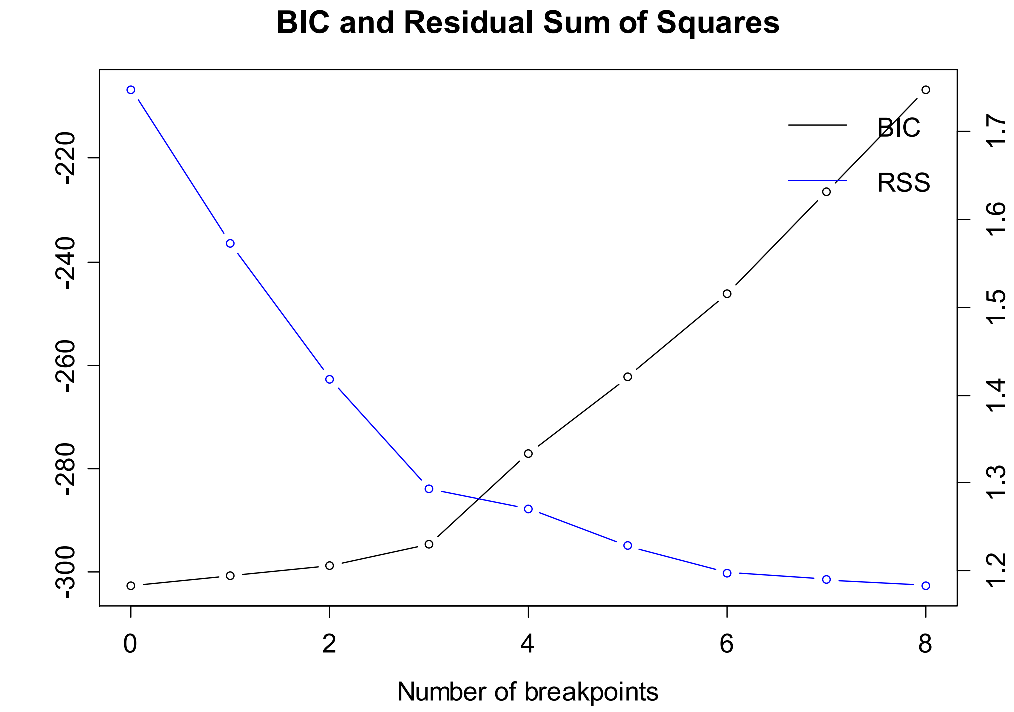
```
library("dynlm")
summary(dynlm(dd ~ L(dd) + L(dd, 12)))
Time series regression with "ts" data:
Start = 1970(1), End = 1984(12)

Call:
dynlm(formula = dd ~ L(dd) + L(dd, 12))

Residuals:
     Min       1Q   Median       3Q      Max
-0.32738 -0.07860  0.01414  0.07284  0.18849

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.42055    0.36327   1.158    0.249
L(dd)        0.43104    0.05327   8.091 9.10e-14 ***
L(dd, 12)    0.51120    0.05653   9.043 2.65e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.09938 on 177 degrees of freedom
Multiple R-squared: 0.6766,    Adjusted R-squared: 0.673
F-statistic: 185.2 on 2 and 177 DF,  p-value: < 2.2e-16
```

on each subset. In the framework of the linear regression model, the setup is

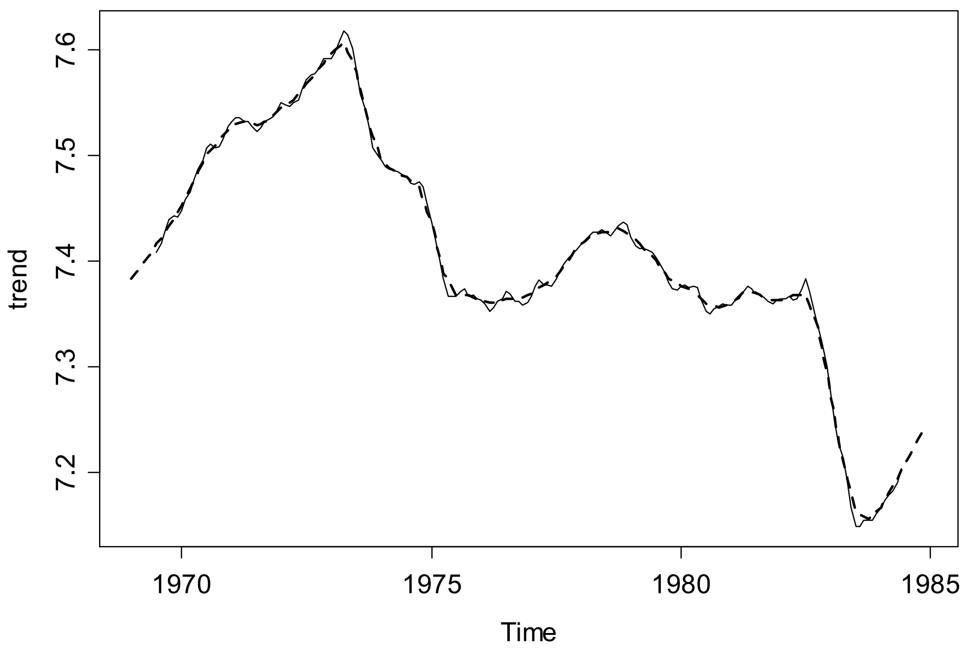
$$y_t = x_t^\top \beta^{(j)} + \varepsilon_t, \qquad t = n_{j-1}+1, \ldots, n_j, \quad j = 1, \ldots, m+1, \qquad (6.4)$$

where $j = 1, \ldots, m$ is the segment index and $\beta^{(j)}$ is the segment-specific set of regression coefficients. The indices $\{n_1, \ldots, n_m\}$ denote the set of unknown breakpoints, and by convention $n_0 = 0$ and $n_{m+1} = n$.

```
library("strucchange")
dd_bp <- breakpoints(dd ~ dd1 + dd12, data = dd_dat, h = 0.1)
plot(dd_bp)
```




```
coef(dd_bp, breaks = 2)
                      (Intercept)       dd1      dd12
1970(1) - 1973(10)       1.457762 0.1173226 0.6944798
1973(11) - 1983(1)       1.534214 0.2182144 0.5723300
1983(2) - 1984(12)       1.686897 0.5486088 0.2141655
Ostatni okres – po wprowadzeniu pasów bezpieczeństwa
```



```
plot(dd)
lines(fitted(dd_bp, breaks = 2), col = 4)
lines(confint(dd_bp, breaks = 2))
```