tfarima: an R package to build customized TFARIMA models

José Luis Gallego
Universidad de Cantabria

Abstract

The R package **tfarima** provides classes and methods to build and use customized Transfer Function and ARIMA models with multiple operators and/or parameter restrictions. Model estimation can be accomplished by exact or conditional maximum likelihood (EML/CML). Procedures for automatic outlier detection, calendar effect estimation, prediction and seasonal adjustment based on this general class of models are also provided. Some classic time series are analyzed to illustrate the usage of the package.

Keywords: ARIMA models, transfer function models, prediction, outliers, R.

1. Introduction

Box and Jenkins (1970) provided a methodology to build a general and flexible class of parsimonious ARIMA and Transfer Function time series models following a three-stages iterative process based on identification, estimation, and diagnostic checking. The fifth edition of the book (Box, Jenkins, Reinsel, and Ljung 2016) includes for first time R code (R Core Team 2018) to replicate many of the applications illustrated along the different chapters of the book. Some of the R packages used in that edition are **stats**, which comes with the base distribution of R, **TSA** (Chan and Ripley 2018), **astsa** (Stoffer 2021) and **MTS** (Tsay, Wood, and Lachmann 2022).

The `arima()` function of the **stats** package enables to fit traditional multiplicative ARIMA $(p, d, q)(P, D, Q)_s$ models to time series with at most two types of serial correlation: regular, dependence between consecutive observations in the series, and seasonal, dependence between observations that are s interval apart. Explanatory and intervention variables such as pulses and steps can be included to predict the output or to handle additive outliers and level shifts. This class of models, sometimes called regARIMA models, is extended by the wrapper `arimax()` function of the **TSA** package to allow for transfer functions (TFs) and innovational outliers. Other R packages with ARIMA modeling capabilities, but oriented to the seasonal adjustment, are the RJDemetra (la Tente, Michalek, Palate, and Baeyens 2021) and seasonal (Sax and Eddelbuettel 2018) packages. TF and ARIMA models can be also estimated in state space form using Kalman filter. Some packages with these capabilities are reviewed by Tusell (2011).

Although the `arima()` and `arimax()` functions are useful to analyze and forecast a great variety of time series, sometimes it is needed to formulate ARIMA models taking into account not only the possibility of multiple dependencies between observations in the series, but

also the type and nature of its different unobserved components that can impose of certain restrictions on the parameters. The `um()` and `tfm` functions of the **tfarima** package here presented extend the `arima()` and `arimax()` functions in these two directions to allow the estimation of univariate models (UMs) and TF models (TFMs) with multiple AR, I and MA operators, as well as with restrictions on the coefficients of these operators. The **tfarima** also provides classes to apply the Box-Jenkins three stages procedures when building TF-ARIMA models, which can be estimated by exact or conditional maximum likelihood. The evaluation of the likelihood function is based on the closed form of the inverse covariance matrix following the algorithm of Ljung and Box (1979). Other capabilities of the package are the implementation of the Box, Pierce, and Newbold (1987) method to decompose seasonal time series using the eventual forecast function, which is compared with the Tramo-Seats method used by RJDemetra, as well as the automatic treatment of outliers and calendar effects following the methods of Chen and Liu (1993) and Bell and Hillmer (1983). A recent, and still in progress, addition to the package is the estimation of basic structural time series models (Durbin and Koopman 2001) following a similar approach to that used for ARIMA models, which allows not only a direct comparison of the results obtained by these two class of models but a productive interaction between both. To this end, functions are provided to obtain the ARIMA reduced form of a basic structural time series model and, vice versa, to obtain the structural form of an ARIMA model.

This manual is organized as follows. Section 2 describes the general class of univariate ARIMA models with multiple lag polynomials and parameter restrictions that can be specified with the `lagpol()` and `um()` functions. Section 3 illustrates the Box-Jenkins methodology to build ARIMA models, as well as the use of these models to forecast and decompose time series. Section 4 shows how to build customized seasonal ARIMA models identical or very similar to three alternative representations: linear regression with deterministic trend and seasonality, the Holt-Winters methods and the Basic Structural Model. Section 5 presents the general class of TF models and how they can be created and estimated with the `tfm()` function. Section 6 concerns with the estimation of intervention models and the automatic or supervised treatment of outliers. Section 7 deals with the calendar effects. Finally, some conclusions and future developments are described in Section 8.

2. Univariate models

The general class of ARIMA(p,d,q) models that can be handled with the **tfarima** package is defined by the equation

$$\varphi_p(B)[\delta_d(B)z_t^{(\lambda)} - \mu] = \vartheta_q(B)a_t, \quad t = 1, 2, \ldots, n; \tag{1}$$

where z_t is a time series of length n; $z_t^{(\lambda)}$ is the power Box-Cox transformation $z_t^{(\lambda)} = (z_t^{\lambda} - 1)/\lambda$; $\varphi_p(B)$, $\delta_d(B)$ and $\vartheta_q(B)$ are the AR, I and MA operators, respectively, which are polynomials in the backshift operator B of degree p, d and q; μ is the mean of the transformed time series $\delta_d(B)z_t^{(\lambda)}$, and a_t is a Gaussian white noise process with variance σ_a^2. Each one of the three operators AR, I and MA can in turn be expressed as the product of several lag polynomials of the form

$$a_d(B^s)^p = (1 - a_1B^s - \cdots - a_dB^{sd})^p, \tag{2}$$

which is a polynomial in B^s of degree d, raised to the integer power p, that is, a polynomial of order (d, s, p). Note that, following the Box-Jenkins notation (Box *et al.* 2016), the polynomial (2) is normalized to $a_d(1) = 1$ and the coefficients $(a_1, \ldots, a_d)$ are preceded by a minus sign. Such coefficients can be expressed as functions of a set of parameters $(b_1, \ldots, b_k)$:

$$a_j = f_j(b_1, \ldots, b_k), \quad j = 1, \ldots, d \text{ and } k \leq d. \tag{3}$$

In this way, the **tarima** package allows us to specify customized ARIMA models with multiple operators and parameter restrictions.

2.1. The `lagpol` class: lag polynomials

Lag polynomials of order (d, s, p) defined by equations (2)-(3) can be created with the `lagpol()` function of the **tfarima** package,

```
lagpol <- function(param = NULL, s = 1, p = 1, lags = NULL, coef = NULL)
```

where `param` is a numeric vector of named parameters, `coef` is a vector of expressions to compute the coefficients of the polynomial as functions of the parameters, `s` is the power of the backshift operator, `p` is the power to the which the polynomial is raised and `lags` is an optional integer vector for unequally spaced polynomials indicating the non-null coefficients.

For example, to create and print the nonseasonal polynomial $(1 - \theta B)^2$ with $\theta = 0.5$, we run the following lines:

```
R> lp1 <- lagpol(param = c(theta = 0.5), p = 2L)
R> lp1

(1 - 0.5B)^2 = 1 - B + 0.25B^2
```

where `param = c(theta = 0.5)` is a single-parameter vector and `p = 2L` indicates that this lag polynomial is raised to the power of 2. Here, the default value for `coef = NULL` is equivalent to `coef = "theta"`. Note that lag polynomials are created following the Box-Jenkins notation. If the notation $(1 + \theta B)^2$ with $\theta = -0.5$ is preferred, the sign of the coefficients must be changed in the `coef` argument,

```
R> lp2 <- lagpol(param = c(theta = -0.5), p = 2L, coef = "-theta")
R> lp2

(1 - 0.5B)^2 = 1 - B + 0.25B^2
```

where `coef = "-theta"` would be the coefficient in $(1 - (-\theta)B)^2$.

The so-called seasonal polynomials are created in a similar way, but using the `s` argument to indicate the seasonal period. For example, the second-order seasonal polynomial $1 - \Theta_1 B^{12} - \Theta_2 B^{24}$ with $\Theta_1 = 1.2$ and $\Theta_2 = -0.9$ is created as follows:

```
R> lp3 <- lagpol(param = c(Theta1 = 1.2, Theta2 = -0.9), s = 12)
R> lp3
```

```
1 - 1.2B^12 + 0.9B^24
```

Customized lag polynomials with restrictions on the coefficients can be created passing to the `coef` argument a vector of strings with such restrictions. For example, to create the lag polynomial with three-coefficients and two-parameters $1 - \theta B - \Theta B^{12} + \theta\Theta B^{13}$, with $\theta = 0.8$ and $\Theta = 0.9$, we run the following sentence:

```
R> lp4 <- lagpol(param = c(theta = 0.8, Theta = 0.9),
+                coef = c("theta", "Theta", "-theta*Theta"),
+                lags = c(1, 12, 13))
R> lp4

1 - 0.8B - 0.9B^12 + 0.72B^13
```

where `lags = c(1, 12, 13)` contains the positions of the three non-null coefficients.

The `lagpol()` function returns an S3 object of class `lagpol`, which is a building block to create ARIMA and TF models. Some useful methods of the `lagpol` class are `roots()` and `inv()`.

```
R> roots(lp4)[c(1:4,10:13),]

       Real Imaginary Modulus Frequency Period Mult.
[1,]  1.009 -1.75e-13    1.01    0.0000    Inf     1
[2,]  1.250  1.10e-13    1.25    0.0000    Inf     1
[3,]  0.874 -5.04e-01    1.01    0.0833   12.0     1
[4,]  0.874  5.04e-01    1.01    0.0833   12.0     1
[5,] -0.504  8.74e-01    1.01    0.3333    3.0     1
[6,] -0.874  5.04e-01    1.01    0.4167    2.4     1
[7,] -0.874 -5.04e-01    1.01    0.4167    2.4     1
[8,] -1.009 -7.36e-16    1.01    0.5000    2.0     1

R> inv(lp4, lag.max = 9)

 [B0]  [B1]  [B2]  [B3]  [B4]  [B5]  [B6]  [B7]  [B8]  [B9]
1.000 0.800 0.640 0.512 0.410 0.328 0.262 0.210 0.168 0.134
```

2.2. The um class: univariate models

Univariate ARIMA models belonging to the class defined by (1) can be created using the `um()` function,

```
um <- function(z = NULL, bc = FALSE, ar = NULL, i = NULL, ma = NULL,
               mu = NULL, sig2 = 1.0,  fit = TRUE)
```

where `z` is an object of class `ts`; `bc` is a logical value that indicates whether or not to take logs; `ar`, `i` and `ma` are lists of objects of class `lagpol`, `character` and/or `numeric` representing

lag polynomials; `mu` is a numeric value for the mean of the stationary series $\delta(B)z_t^{(\lambda)}$; `sig2` is a numeric value for the variance of the error, and `fit` is a logical value indicating whether or not to estimate the model. The function returns an object of class `um`.

To illustrate the usage of this function we create a multiplicative ARIMA model with orders $(0,1,1)(0,1,1)_3(0,1,1)_{12}$,

$$\nabla\nabla_3\nabla_{12}Z_t = (1-\theta_1 B)(1-\theta_2 B^3)(1-\theta_3 B^{12})a_t$$

where $\theta_1 = \theta_2 = \theta_3 = 0.8$ and $\sigma_a^2 = 1$. We can set the arguments `ar`, `i` and `ma` providing three types of objects:

1. Lists of objects of class `lagpol`:

```
R> d1 <- lagpol(coef = 1)
R> d3 <- lagpol(coef = 1, s = 3)
R> d12 <- lagpol(coef = 1, s = 12)
R> ma1 <- lagpol(param = c(th1 = 0.8))
R> ma3 <- lagpol(param = c(th2 = 0.8), s = 3)
R> ma12 <- lagpol(param = c(th3 = 0.8), s = 12)
R> um1 <- um(i = list(d1, d3, d12), ma = list(ma1, ma3, ma12))
```

   In this code chunk, we first create six `lagpol` objects, one for each operator of the model, and then we pass them as lists to the arguments of the `um` function. This approach is recommended to create customized models since it provides a way to specify names, preestimates and restrictions for parameters.

2. Character strings with the equations of the polynomials:

```
R> um2 <- um(i = "(1 - B)(1 - B3)(1 - B12)",
+            ma = "(1 - .8B)(1 - .8B3)(1 - .8B12)")
```

   Here we provide to each argument `i` and `ma` the equations of the operators as character strings. This way of creating a model is more compact than the previous one, but doesn't allow us neither to set the names of the parameters nor create customized models. It is useful to study the statistical properties of common ARMA models (Section 4.1).

3. Lists containing the orders `c(d, s = 1, p = 1)` of the operators:

```
R> um3 <- um(i = list(1, c(1, 3), c(1, 12)),
+            ma = list(1, c(1, 3), c(1, 12)))
```

   In this case, we only provide the orders of each type of operator and the function sets the names and values of the parameters: .1 for AR parameters, .2 for MA parameters and 1 for I coefficients. Hence, `ar = list(1, c(1, 3), c(1, 12))` and `ma = list(1, c(1, 3), c(1, 12))` are equivalent to `ar = "(1 - .1B)(1 - .1B3)(1 - .1B12)"` and `ma = "(1 - .2B)(1 - .2B3)(1 - .2B12)"`, whereas `i = list(1, c(1, 3), c(1, 12))` is equivalent to `i = "(1 - B)(1 - B3)(1 - B12)"`. This approach is convenient to fit conventional models to time series.

We can combine these three types of objects to set the arguments `ar`, `i` and `ma`:

```
R> um4 <- um(i = list(1, "1 - B3", d12), ma = list("1 - .8B", c(1, 3), ma12))
```

Each list of operators AR, I and MA in a `um` object, and the corresponding extended polynomials, are stored into the data members `ar`, `i` and `ma`, `phi`, `nabla` and `theta`.

```
R> printLagpolList(um1$i)

[1] 1 - B   [2] 1 - B^3   [3] 1 - B^12

R> printLagpolList(um1$ma)

[1] 1 - 0.8B   [2] 1 - 0.8B^3   [3] 1 - 0.8B^12

R> nabla(um1)

1 - B - B^3 + B^4 - B^12 + B^13 + B^15 - B^16

R> theta(um1)

1 - 0.8B - 0.8B^3 + 0.64B^4 - 0.8B^12 + 0.64B^13 + 0.64B^15 - 0.51B^16
```

Note that we have created the object `um1` without providing a time series, which is useful, for example, to study the statistical properties of a particular model, to simulate data or to fit the same model to several time series. These and other capabilities for objects of class `um` are explained in the next sections.

3. ARIMA model building

This section illustrates the use of the **tfarima** package to build ARIMA models following the Box-Jenkins three-stage procedure, which requires to be familiar with the characteristics of some representative members of this broad class of stochastic processes.

3.1. Statistical properties

We can create an object of class `um` to study the characteristics of a particular ARMA model in terms of the simple and partial autocorrelation functions (ACF and PACF) and the spectral density function. The following code creates five basic time series models and displays their main functions (see Figure 1):

```
R> ar1p <- um(ar = "(1 - 0.9B)")
R> ar1n <- um(ar = "(1 + 0.9B)")
R> ma1p <- um(ma = "(1 - 0.9B)")
R> ma1n <- um(ma = "(1 + 0.9B)")
R> ar2c <- um(ar = "(1 - 1.52B + 0.8B^2)")
R> display(list(ar1p, ar1n, ma1p, ma1n, ar2c), lag.max = 20)
```

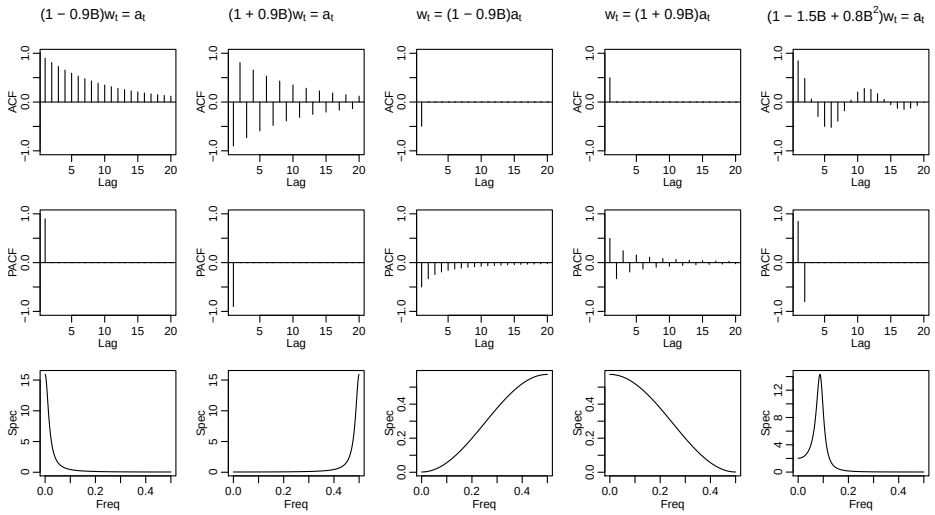

Figure 1: ACF, PACF and spectrum of five UM's.

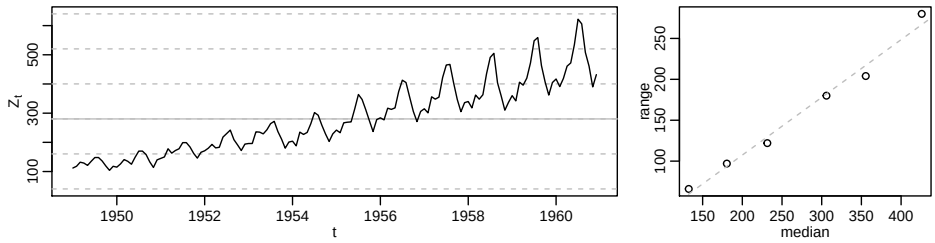

Figure 2: Plot and range-median diagram for the monthly AirPassenger series.

The `display()` function shows the ACF, PACF and/or spectrum of a list of objects of class `um`. Other methods of the `um` class to characterize stochastic processes are `pi.weights()` and `psi.weights()`, which show the AR(∞) and MA(∞) forms; `autocov()` and `autocorr()`, which compute the theoretical autocovariances and (simple/partial) autocorrelations, and `roots()`, which computes the roots of the different polynomials of the model.

3.2. Model identification

Figure 2 shows the plot and the range-median diagram for the totals of international airline passengers listed as Series G in Box *et al.* (2016). These graphs have been generated with the `ide()` function of the **tfarima** package:

```
R> Z <- AirPassengers
R> ide(Z, graphs = c("plot", "rm"))
```

where the option `rm` in the `graphs` argument means range-median.

The `ide()` function can also show several graphs for a list of transformations. Figure 3 displays the plot, ACF and PACF for the transformed series: $\log(Z_t)$, $(1 + B + \cdots + B^{11})\log(Z_t)$ and $\nabla\nabla_{12}\log(Z_t)$. The `transf` argument allows us to set a list of lists of transformations such as

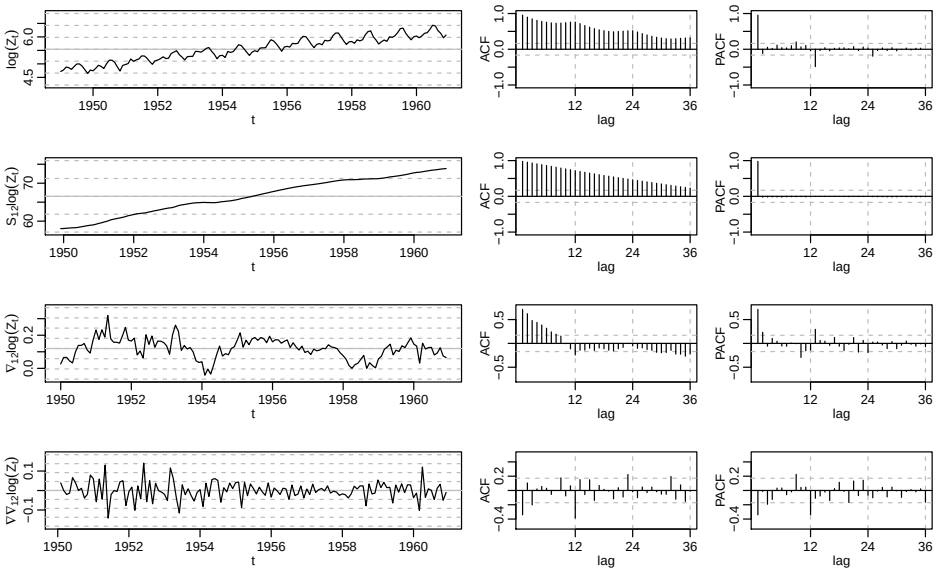

Figure 3: Some identification tools for three data transformations of AirPassengers

the Box-Cox transformation (bc), the number of nonseasonal and seasonal differences (d and D), the annual sum (S) or a list of `lagpol` objects with the integrated operators (i).

```
R> ide(Z, transf = list(list(bc = T), list(bc = T, S = 1),
+                       list(bc = T, D = 1), list(bc = T, D = 1, d = 1)))
```

3.3. Model estimation

ARIMA models created with the `um` function can be estimated by exact or conditional maximum likelihood. We can create and estimate an ARIMA model with the `um` function by providing a `ts` object or we can firstly create a `um` object and then fit to a series with the function `fit`.

```
R> um1 <- um(AirPassengers, bc = TRUE, i = list(1, c(1, 12)),
+            ma = list(1, c(1, 12)))
R> um2 <- um(i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)
R> um2 <- fit(um2, AirPassengers)
R> um1

        Estimate Std. Error
theta1 0.4018464 0.07656434
theta2 0.5570393 0.07401700

log likelihood:  244.6965
Residual standard error:  0.03513935
aic: -3.70529
```

We can check that these estimatates are very similar to that provided by the `arima()` function of the **stats** package

```
R> arima(log(AirPassengers), order = c(0,1,1),
+        seasonal = list(order = c(0,1,1), frequency = 12))

Call:
arima(x = log(AirPassengers), order = c(0, 1, 1), seasonal = list(order = c(0,
    1, 1), frequency = 12))

Coefficients:
          ma1     sma1
      -0.4018  -0.5569
s.e.   0.0896   0.0731

sigma^2 estimated as 0.001348:  log likelihood = 244.7,  aic = -483.4
```

3.4. Model diagnostic checking

Detailed results of the estimation together with diagnostic statistics can be printed with the `summary()` function:

```
R> summary(um2)

Model:
 um2  <-  um(i = list(1, c(1, 12)), ma = list(1, c(1, 12)), bc = TRUE)

Time series:
AirPassengers

Maximum likelihood method:
 exact

Coefficients:
        Estimate  Gradient Std. Error z Value Pr(>|z|)
theta1 4.018e-01 6.544e-06  7.656e-02   5.248 1.53e-07 ***
theta2 5.570e-01 2.775e-05  7.402e-02   7.526 5.24e-14 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Total nobs                  144 Effective nobs             131
log likelihood            244.7 Error variance         0.001348
Mean of residuals    -0.0004231 SD of the residuals     0.03514
z-test for residuals    -0.1445 p-value                 0.8851
Ljung-Box Q(1) st.        10.08 p-value                 0.001495
Ljung-Box Q(32) st.        47.4 p-value                 0.03906
Barlett H(3) stat.        1.138 p-value                   0.566
AIC                      -3.705 BIC                     -3.661
```

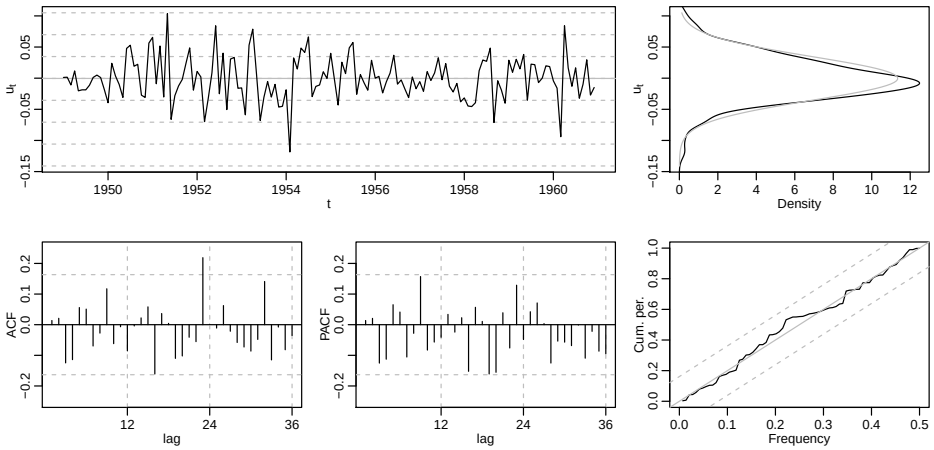

Figure 4: Some diagnostic tools for the residuals of model um1

whose argument is an object of class `um`. Some diagnostic graphs for this object can be shown with the `diagchk()` function (see Figure 4):

```
R> diagchk(um1)
```

Overfitting is a technique that can be used for diagnostic checking. We can extend our model by adding more AR, I and MA polynomials,

```
R> modify(um1, ar = list(2, c(2, 12)))
```

```
          Estimate Std. Error
phi1    0.55477214 0.08502302
phi2    0.24975326 0.08322806
phi3   -0.09466045 0.23810134
phi4   -0.03393216 0.14198889
theta1  0.96372038 0.04100168
theta2  0.48331617 0.23071063

log likelihood:  246.2311
Residual standard error:  0.03447009
aic: -3.66765
```

To remove a lag polynomial we indicate its possition with a negative integer. Here we replace the two MA(1) polynomials with two MA(2) polynomials,

```
R> um3 <- modify(um1, ma = list(-1, -2, 2, c(2, 12)))
R> printLagpolList(um3$ma)
```

```
[1] 1 - 0.41B - 0.037B^2   [2] 1 - 0.6B^12 + 0.067B^24
```

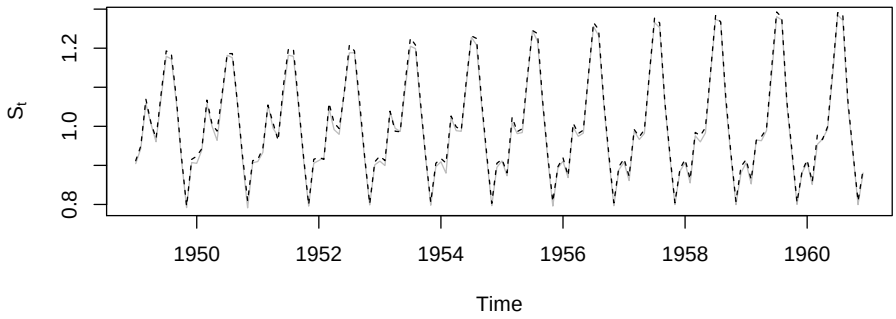

Figure 5: Seasonal components for AirPassengers estimated with **RJDemetra** and **tfarima**.

3.5. Forecasting

Point and interval forecast from a `um` object can be computed with the `predict()` function, whose arguments `n.ahead` and `level` allow to set the lead time and the confidence levels. The object returned by this function can be displayed with the `plot()` function, whose `n.back` argument sets the number of previous observations to be shown.

```
R> p <- predict(um2, n.ahead = 12)
R> #plot(p, n.back = 48)
```

3.6. Unobserved components

The `ucomp()` function computes unobserved components for a time series based on an object of class `um`, which can be graphed with the `plot` command. Figure 5 shows the seasonal component estimated with the **tfarima** and `RJDementra` packages. We can see that, for this time series, the results are practically identical.

```
R> uc1 <- ucomp(um2)
R> #plot(uc1)
R> library(RJDemetra)
R> ts1 <- tramoseats(AirPassengers, spec = "RSA5")
R> plot(ts1$final$series[,4], ylab = expression('S'[t]), col = "gray")
R> lines(exp(uc1$seas), lty = 2)
```

4. Customized seasonal ARIMA models

Three alternative forecasting methods, closely related to ARIMA models, are the linear regression model with deterministic linear trend and seasonal dummies, the Holt-Winters exponential smoothing method and the structural time series models. These three representations

can be reformulated as ARIMA models with restrictions on the MA parameters. This section illustrates the estimation of these restricted ARIMA models.

4.1. Regression models with deterministic components

It is well known that the linear regression model with deterministic trend and seasonalilty,

$$Y_t = \mu_0 + \mu_1 t + \sum_{j=1}^{s-1} D_{jt} + u_t, \quad t = 1, \ldots, n \tag{4}$$

is equivalent to the non-invertible multiplicative IMA(1,1)(1,1)$_s$ model

$$\nabla\nabla_s Y_t = (1 - B)(1 - B^s)u_t \tag{5}$$

where the seasonal difference ∇_s removes the intercept μ_0 and the set of seasonal dummies D_{jt} and transforms the trend t into a constant $\nabla_s t = s$, which is later removed by the regular difference ∇ (see, e.g., Bell 1987). As a result of this equivalence, both models provide the same residuals. The next code chunk estimates the linear regression model (4) with the `lm()` function and the non-invertible IMA model (5) with the `um()` and `arima()` functions. In Figure 6, we can see that the least-squares residuals of (4) and the exact residuals of (5) computed by the `um` functions are the same series, which differs from the innovations provided by the `arima()` function.

```
R> t <- 0:(length(AirPassengers) - 1)
R> D <- as.factor(cycle(AirPassengers))
R> reg <- lm(log(AirPassengers) ~ t + D)
R> ima1 <- um(AirPassengers, , bc = TRUE, i = list(1, c(1, 12)),
+             ma = "(1-B)(1-B12)", fit = FALSE)
R> ima2 <- arima(log(AirPassengers), order = c(0,1,1),
+         seasonal = list(order = c(0,1,1), frequency = 12),
+         fixed = c(-1, -1), transform.pars = FALSE, method = "ML")
R> res1 <- ts(residuals(reg), start = start(AirPassengers), frequency = 12)
R> res2 <- residuals(ima1)
R> res3 <- residuals(ima2)
```

Note that in the `um()` function we use the argument `fit = FALSE` because all the parameters are fixed and we don't need to estimate the model. If we extend this model with two AR(2) operators, then to fix the non-invertible MA polynomials we have to create two MA polynomials, MA(1) and MA(1)$_{12}$, or a MA(13) polynomial with the `lagpol()` function setting the values of the coefficients in the `coef` argument. Although it is not necessary, two AR(2) polynomials are also created to highlight the way of creating a unrestricted and restricted polynomials. Note that the argument `ar = list(ar2, AR2)` could be replaced by `ar = list(2, c(2, 12))`.

```
R> #i1 <- lagpol(coef = 1); i12 <- lagpol(coef = 12); i <- list(i1, i12)
R> i <- lagpol(lags = c(1, 12, 13), coef = c(1,1,-1))
R> ar2 <- lagpol(param = c(phi1 = .1, phi2 = .1))
R> AR2 <- lagpol(param = c(PHI1 = .1, PHI2 = .1), s = 12)
R> um(AirPassengers, bc = TRUE, ar = list(ar2, AR2), i = i, ma = i)
```

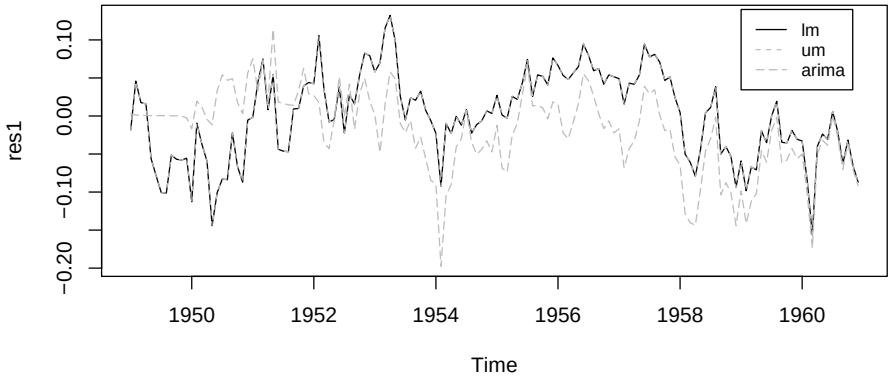

Figure 6: Least-squares residuals, exact ARIMA residuals and innovations.

```
      Estimate Std. Error
phi1 0.5697594 0.07460517
phi2 0.2755773 0.07525051
PHI1 0.4251549 0.07995199
PHI2 0.3131587 0.07636835

log likelihood:  243.8756
Residual standard error:  0.0333997
aic: -3.662224
```

4.2. Exponential smoothing

Roberts (1982) shown that the additive version of Holt-Winters method

$$\begin{aligned} Y_{t+h} =& \mu_t + \beta_t h + S_{t+h-s} + u_{t+h} \\ \mu_t =& \mu_{t-1} + \beta_{t-1} + \alpha_1 u_t \\ \beta_t =& \beta_{t-1} + \alpha_2 u_t \\ S_t =& S_{t-s} + \alpha_3 u_t, \end{aligned}$$

can be written as an ARIMA model

$$\nabla\nabla_s Y_t = (1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_{s+1} B^{s+1}) u_t,$$

with $\theta_1 = 1 - \alpha_1 - \alpha_2, \theta_2 = \cdots = \theta_{s-1} = -\alpha_2, \theta_s = 1 - \alpha_2 - \alpha_3, \theta_{s+2} = -(1 - \alpha_1 - \alpha_3)$. For a monthly time series ($s = 12$), and assigning preestimates to the smoothing parameters, we can create this class of restricted MA(13) polynomials as follows,

```
R> coef <- c("1-a1-a2", rep("-a2", 10), "1-a2-a3", "-(1-a1-a3)")
R> lp <- lagpol(param = c(a1 = 0.2, a2 = 0.1, a3 = 0.1), coef = coef)
R> lp
```

```
1 - 0.7B + 0.1B^2 + 0.1B^3 + 0.1B^4 + 0.1B^5 + 0.1B^6 + 0.1B^7 +
0.1B^8 + 0.1B^9 + 0.1B^10 + 0.1B^11 - 0.8B^12 + 0.7B^13
```

In the following code chunk, the HW-ARIMA model is first estimated with the `um` function and then the smoothing parameters are estimated with the `hw()` function of the **forecast** package, choosing both optimal and simple preestimates of the initial states $\mu_0, \beta_0, S_0, \ldots, S_{-11}$.

```
R> library(forecast)
R> um0 <- um(AirPassengers, bc = TRUE, i = list(1, c(1, 12)), ma = lp)
R> hw1 <- hw(log(AirPassengers))
R> hw2 <- hw(log(AirPassengers), initial = "simple")
```

Given that the values of the likelihood provided by the two packages are not comparable, the estimates of the smoothing parameters obtained with the **forecast** package are used to create the equivalent HW-ARIMA model with the `um()` function and evaluate its log-likelihood.

```
R> a <- unname(hw1$model$par[1:3])
R> lp <- lagpol(param = c(a1 = a[1], a2 = a[2], a3 = a[3]), coef = coef)
R> um1 <- um(log(AirPassengers), i = list(1, c(1, 12)), ma = lp, fit = FALSE)
R> a <- unname(hw2$model$par[1:3])
R> lp <- lagpol(param = c(a1 = a[1], a2 = a[2], a3 = a[3]), coef = coef)
R> um2 <- um(log(AirPassengers), i = list(1, c(1, 12)), ma = lp, fit = FALSE)
R> cbind(hw_um = c(coef(um0), lik = logLik(um0)),
+        hw_simple = c(coef(um2), logLik(um2)),
+        hw_opt = c(coef(um1), logLik(um1)))

            hw_um    hw_simple       hw_opt
a1  3.502908e-01 3.405976e-01 6.974804e-01
a2  9.648322e-04 1.485999e-04 3.058347e-03
a3  4.966692e-01 5.288037e-01 1.133132e-04
lik 2.379866e+02 2.379169e+02 2.292592e+02
```

4.3. Basic structural model

Harvey and Durbin (1986) consider a basic structural model with a trend μ_t, a seasonal S_t and an irregular u_t component for a seasonal time series Y_t with frequency s observations per unit of time,

$$Y_t = \mu_t + S_t + u_t, \quad u_t \sim iidN(0, \sigma_u^2),$$

where the trend component is specified as

$$\begin{aligned} \mu_t =& \mu_{t-1} + \beta_{t-1} + v_t, \quad v_t \sim iidN(0, \sigma_v^2) \\ \beta_t =& \beta_{t-1} + \omega_{0t}, \quad \omega_{0t} \sim iidN(0, \sigma_{w_0}^2) \end{aligned}$$

and the seasonal component $S_t = \sum_{k=1}^{[s/2]}$ is formulated as the sum of $[s/2]$ seasonal cycles with frequencies $2\pi k/s (k = 1 \ldots, [s/2])$ given by

$$\begin{aligned} S_{k,t} =& c_k S_{k,t-1} + s_k S_{k,t-1}^* + \omega_{kt}, \quad \omega_{kt} \sim iidN(0, \sigma_{w_k}^2) \\ S_{k,t}^* =& -s_k S_{k,t-1} + c_k S_{k,t-1}^* + \omega_{kt}^*, \quad \omega_{kt}^* \sim iidN(0, \sigma_{w_k^*}^2) \end{aligned}$$

being [s/2] the integer division, $c_k = \cos(2\pi k/s)$ and $s_k = \sin(2\pi k/s)$ and $S_{kt} = -S_{kt-1} + \omega_{kt}$ for $k = s/2$ when s even. The set of shocks u_t, v_t and w_{kt} $(k = 0, \ldots, [s/2])$ are mutually independent. The BSM reduces to an IMA(13,13) model $\nabla\nabla Y_t = \theta(B)a_t$ where complex restrictions are imposed on the MA parameters.

For comparison purposes, a `bsm()` function has been added to the **tfarima** package to estimate this class of models as special cases of ARIMA models so that the results obtained for both models are directly comparable. The `bsm()` function can also be used to estimate the versions of Harvey and Todd (1983) and Harrison and Stevens (1976) of this model.

For a vector of disturbances $(v_t, \omega_{0t}, \omega_{1t}, \ldots, \omega_{10t}, \omega_{11t})'$ with diagonal covariance matrix

$$\text{diag}(\Sigma) = (\sigma_v^2, \sigma_0^2, \sigma_\omega^2, \ldots, \sigma_\omega^2, 0.5\sigma_\omega^2),$$

the `bsm()` function provide the following results

```
R> bsm1 <- bsm(AirPassengers, bc = TRUE)
R> bsm1
```

```
              Var        Ratio
lvl  2.984008e-04 1.000000e+00
slp  1.256387e-10 4.210401e-07
seas 3.560024e-06 1.193034e-02
irr  2.345546e-04 7.860387e-01

log likelihood:  242.0884
```

The reduced form of the `bsm1` model can be found wiht the `rfom()` function,

```
R> rf1 <- rform(bsm1)
R> nabla(rf1)
```

```
1 - B + 6.9e-16B^2 - 7.2e-16B^3 - 3.9e-16B^4 + 1.1e-17B^5 + 7e-16B^6
- 9.8e-16B^7 - 7.2e-16B^8 + 1.8e-15B^9 - 3.1e-16B^10 + 8.2e-16B^11 -
B^12 + B^13
```

```
R> theta(rf1)
```

```
1 - 0.52B + 0.024B^2 - 0.047B^3 + 0.025B^4 - 0.041B^5 + 0.021B^6 -
0.039B^7 + 0.012B^8 - 0.046B^9 - 0.012B^10 - 0.08B^11 - 0.46B^12 +
0.17B^13
```

The reduced form `rf1` is an object of class `um` and can be used to compute residuals, as well as to forecast and decompose time series. The BSM can be compared with basic ARIMA model estimated in the following subsection.

4.4. Factored seasonal ARIMA models

As the Holt-Winters method, the ARIMA$(0,1,1)(0,1,1)_s$ model,

$$(1-B)(1-B^s)z_t = (1-\theta B)(1-\Theta B^s)a_t, \tag{6}$$

doesn't not allow to identify neither series with deterministic linear trend and stochastic seasonality nor series with stochastic linear trend and deterministic seasonality. The first type of series would arise with the cancellation of the two regular differences $(1-B)^2$ on both sides of the equation, while the second one would imply the cancellation of the annual sum operator $1+B+\cdots+B^{s-1}$. It is not possible either to identify series with mixed deterministic-stochastic seasonality, which would require the cancellation of some, but not all, factors of the seasonal difference. To overcome these restrictions Gallego and Treadway (1995) considered two generalizations of (6): the basic generalization with three parameters

$$(1-B)^2(1+B+\cdots+B^{s-1})z_t = (1-\theta B)(1-\Theta_0^{1/s}B)(1+\Theta_1^{1/s}B+\cdots+\Theta_1^{(s-1)/s}B^{s-1})a_t \quad (7)$$

and the full generalization with $[s/2]+1$ parameters

$$\begin{aligned}(1-B)^2(1+B)\prod_{k=1}^{[(s-1)/2]}(1-2\cos(2\pi k/s)B+B^2)z_t = \\ (1-\theta B)(1-\Theta_0^{1/s}B)[(1+\Theta_{s/2}^{1/s}B)]\prod_{k=1}^{[(s-1)/2]}(1-2\cos(2\pi k/s)\Theta_k^{1/s}B+\Theta_k^{2/s}B^2)a_t.\end{aligned} \quad (8)$$

Note that model (8) reduces to (7) when $\Theta_k=\Theta_1$ for $k=1,\ldots,[s/2]$, and (7) reduces to (6) when $\Theta_0=\Theta_1$. In both equations (7) and (8), the linear trend is deterministic when $\theta=\Theta_0=1$, and seasonality is deterministic when $\Theta_1=\Theta_k=1$ for $k=1,\ldots,[s/2]$. In (8), mixed deterministic-stochastic seasonality would arise when $\Theta_k=1$ for some $k\in\{1,\ldots,[s/2]\}$. Note also that the paramaters Θ_k are rised to a power multiple of $1/s$ to ease the comparisons with the airline model (6), but such powers could be omited.

The basic generalized airline model with three MA parameters (7) can be created and estimated as follows

```
R> coef <- paste("-Theta1^(", 1:11,"/12)", sep = "")
R> lp3 <- lagpol(param = c(Theta1 = 0.8), coef = coef)
R> lp2 <- lagpol(param = c(Theta0 = 0.8), coef = "Theta0^(1/12)")
R> bam1 <- um(AirPassengers, bc = TRUE, i = list(1, c(1,12)),
+             ma = list(1,lp2,lp3))
R> bam1

        Estimate Std. Error
theta1 0.3789938 0.07891351
Theta0 0.9994439 0.25674367
Theta1 0.5405008 0.07621691

log likelihood:  245.7904
Residual standard error:  0.0346022
aic: -3.706724
```

We can see that the basic factorization of the $\mathrm{MA}(1)_{12}$ polynomial reveals the presence of a unit MA root that implies a linear trend with a deterministic slope, which is coherent with the estimates of the BSM reported in the previous subsection. It is interesting to compare the

roots of the MA polynomial of these models. We can see that the seasonal MA roots of the `bam1` model are exactly at the range of seasonal frequencies $1/12, 2/12, \ldots, 0.5$, while that the corresponding MA roots in the codebsm1 are very close to this range of frequencies. Another difference is that all the seasonal MA roots in the `bam1` model have the same modulus, which doesn't ocurr in the `bsm1` model. Finally, the log-likehood is slightly superior for the `bam1` model.

```
R> rbam1 <- roots(bam1$theta)
R> rbsm1 <- roots(rf1$t)
R> cbind(rbam1[, 3:5], rbsm1[, 3:5])
```

```
       Modulus  Frequency Period  Modulus  Frequency    Period
 [1,] 1.000046 0.00000000    Inf 1.000652 0.00000000       Inf
 [2,] 2.638565 0.00000000    Inf 2.920373 0.00000000       Inf
 [3,] 1.052609 0.08333333   12.0 1.036727 0.08301823 12.045548
 [4,] 1.052609 0.08333333   12.0 1.036727 0.08301823 12.045548
 [5,] 1.052609 0.16666667    6.0 1.058420 0.16642798  6.008605
 [6,] 1.052609 0.16666667    6.0 1.058420 0.16642798  6.008605
 [7,] 1.052609 0.25000000    4.0 1.068622 0.24987214  4.002047
 [8,] 1.052609 0.25000000    4.0 1.068622 0.24987214  4.002047
 [9,] 1.052609 0.33333333    3.0 1.073313 0.33327280  3.000545
[10,] 1.052609 0.33333333    3.0 1.073313 0.33327280  3.000545
[11,] 1.052609 0.41666667    2.4 1.074949 0.41666987  2.399982
[12,] 1.052609 0.41666667    2.4 1.074949 0.41666987  2.399982
[13,] 1.052609 0.50000000    2.0 1.109723 0.50000000  2.000000
```

In the same way that a structural model can be reduced to an ARIMA model, an ARIMA model can be formulated as a structural model, which can be thought of as the solution of the difference equation of the ARIMA model. The `sform()` function returns the structural form of an ARIMA model,

```
R> sf1 <- sform(bam1)
R> sf1$b # vector link
```

```
 [1]  1.000000e+00  1.297573e-15  1.000000e+00  6.390409e-16
 [5]  1.000000e+00  5.979725e-16  1.000000e+00  2.075947e-15
 [9]  1.000000e+00 -6.251810e-16  1.000000e+00  0.000000e+00
[13]  1.000000e+00
```

```
R> sf1$C[3:4, 3:4] # State-transition submatrix
```

```
           [,1]      [,2]
[1,]  0.8660254 0.5000000
[2,] -0.5000000 0.8660254
```

```
R> diag(sf1$Sv)
```

```
 [1] 0.0004262411 0.0003847401 0.0003472818 0.0003134723 0.0002829560
 [6] 0.0002554122 0.0002305511 0.0002081114 0.0001878572 0.0001695756
[11] 0.0001530744 0.0008046841 0.0004266362
```

We can see that the disturbances of the seasonal components for the `bam1` don't have equal variances.

The generalized airline model with seven MA parameters (8) can be created and estimated as follows

```
R> lp0 <- lagpol(param = c(Theta0 = 0.5), coef = "Theta0^(1/12)")
R> lp6 <- lagpol(param = c(Theta6 = 0.5), coef = "-Theta6^(1/12)")
R> lp15 <- lapply(1:5, function(k) {
+    th <- paste("Theta", k, sep = "")
+    param <- 0.8
+    names(param) <- th
+    coef1 <- paste("2*cos(2*pi*", k, "/12)*", th, "^(1/12)", sep = "")
+    coef2 <- paste("-", th, "^(2/12)", sep = "")
+    lagpol(param = param, coef = c(coef1, coef2))
+  })
R> gam <- um(AirPassengers, bc = TRUE, i = list(1, c(1,12)),
+            ma = list(1, lp0,lp15,lp6))
R> gam

        Estimate Std. Error
theta1 0.4493981 0.09018171
Theta0 0.9991544 0.23126557
Theta1 0.4500810 0.14400633
Theta2 0.4432250 0.12780458
Theta3 0.9914946 0.15453472
Theta4 0.5378578 0.15690862
Theta5 0.6285395 0.12799566
Theta6 0.9958152 0.20530270

log likelihood:  249.2741
Residual standard error:  0.03292445
aic: -3.683575
```

5. Transfer function models

Transfer function models describes the dynamic relationship between an output Y_t and one or several inputs X_{jt}. The impulse response function (IRF) $\nu_j(B) = \nu_{j0} + \nu_{j1}B + \nu_{j2}B^2 + \ldots$ for each input X_{jt} can be approximated by the rational transfer function of order (r_j, s_j, b_j)

$$\nu_j(B) = w_{j0}\frac{w_{js_j}(B)}{\delta_{jr_j}(B)}B^{b_j}, \tag{9}$$

where b_j is the delay or dead time, $w_{j0} = \nu_{jb}$ is the first non-null coefficient of the IRF $\nu_j(B)$, and $\delta_{jr_j}(B) = 1 - \delta_{j1}B - \cdots - \delta_{jr_j}B^{r_j}$ and $w_{js_j}(B) = 1 - \omega_{j1}B - \cdots - \omega_{jr_j}B^{r_j}$ are here polynomials of order s_j and r_j that, like the AR and MA operators of (1), can be expressed as the product of multiple normalized polynomials of the class (2)-(3). Note that the $w_{js_j}(B)$ polynomial multiplied by the scalar w_{j0} would correspond to the form of the polynomial $\omega(B) = w_0 - w_1B - \cdots - w_sB^s$ defined by Box *et al.* (2016).

Combining (1) and (9) we obtain a rich class of TF models for an output Y_t given by

$$\begin{aligned} Y_t &= \sum_{j=1}^{k} w_{j0} \frac{\omega_{js_j}(B)}{\delta_{jr_j}(B)} B^{b_j} X_{jt} + N_t \\ \delta_d(B) N_t &= \mu + \frac{\vartheta_q(B)}{\varphi_p(B)} a_t, \end{aligned} \tag{10}$$

where the noise N_t follows a general ARIMA(p, d, q) model of the class (1). This specification contains as special cases the intervention models used to deal with four common types of outliers: additive outlier (AO), innovative outlier (IO), level shift (LS) and transitory change (TC), see, e.g., Box and Tiao (1975) and Chen and Liu (1993).

5.1. The tfm class: transfer function models

To create TF models defined by equation (10), we have to create previously both the TF for each input and the UM for the noise. The `tf()` function of the **tfarima** package specifies the TF for a particular input,

```
tf <- function(x = NULL, delay = 0, w0 = 0.1, ar = NULL, ma = NULL,
               um = NULL, par.prefix = "")
```

where the input `x` is an object of class `"ts"`, `delay` is an integer value for the input lag, `w0` a numeric value for the parameter w_0 of the TF, `ar` and `ma` are lists of objects of class `lagpol` in the denominator (AR) and numerator (MA) of the TF, `um` is an optional object of class `um` used to back/forecast the input `x`, and `par.prefix` is an optional character to set a prefix for the parameters of the TF. Any of the three ways described to provide lag polynomials in the `um()` function can also be used with the `tf()` function.

Once we have specified a TF for each input, we can create a TF model with the `tfm()` function,

```
tfm <- function(output = NULL, xreg = NULL, inputs = NULL, noise, fit = TRUE)
```

where `output` is an object of class `ts`, `xreg` is a vector or matrix of regressors, `inputs` is a list of objects of class `tf`, `noise` is an object of class `um` with the UM for the noise, `fit` is a logical value indicating whether or not to fit the TF model. Two comments are in order: (1) if the `um` object for the output is set to the `noise` argument, then it is not necessary provide the `output` argument because it is already contained in this `um` object; (2) time series for the inputs and regressors must have at least the same length as the output, but they can be extended with backcasts and forecasts to improve the estimation or to forecast the output (more on that later).

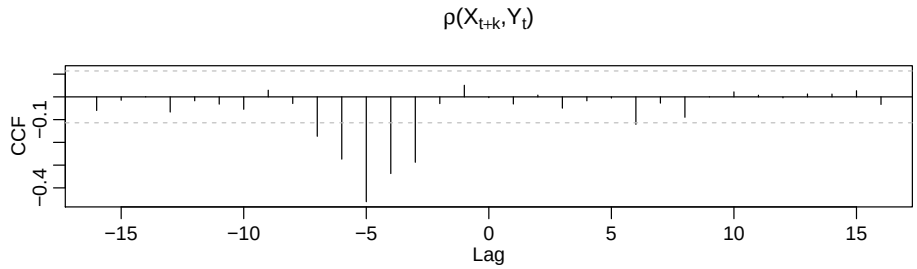

Figure 7: Estimated cross correlation function.

The function returns an object of class `tfm`, whose main data members are `xreg`, `inputs` and `noise`. Some useful methods for these of objects are: `noise()`, `fit()`, `diagchk()`, `calendar()`, `outliers()`, `predict()` and `ucomp()`, which are illustrated in the following subsections.

5.2. Building transfer function models

To illustrate the Box-Jenkins approach to the identification, fitting and checking of TF models, we replicate the analysis of the gas furnace data by Box *et al.* (2016), Series J. They identify and fit an AR(3) model for the input X_t, which can be also fitted to the output Y_t:

```
R> Y <- seriesJ$Y - mean(seriesJ$Y)
R> X <- seriesJ$X - mean(seriesJ$X)
R> umx <- um(X, ar = 3)
R> umy <- fit(umx, Y)
```

This `umx` model is used to prewhiten the input X and the output Y. The `residuals()` function of the **tfarima** package compute the conditional or exact residuals for a time series from an object of class `um`:

```
R> a <- residuals(umx, Y, method = "cond")
R> b <- residuals(umx, X, method = "cond")
```

Now we can use the `ccf()` function of the **stats** package to display the estimated cross correlation function for the gas furnace data after filtering. Alternatively, we can use the `pccf()` function of the **tfarima** defined as

```
R> pccf(X, Y, um.x = umx, um.y = NULL, lag.max = 16)
```

where `x` and `y` are the input and the output, `um.x` and `um.y` are the univariate models used to prewhiten both series, and `lag.max` is the number of correlation in each side. If both `um.x` and `um.y` arguments are equal to `NULL`, the ccf is estimated without prewhitening; if only one of these arguments is equal to `NULL`, the `um` provided is used to prewhiten both series; otherwise, each series is prewhitenned by its own UM.

As in Box *et al.* (2016) we identify a TF with orders $(1, 2, 3)$ or $(2, 2, 3)$, see Figure 7.

Preestimates of the parameters of the TF can be computed with the `tfest()` function

```
R> tfx <- tfest(Y, X, delay = 3, p = 2, q = 2, um.x = umx, um.y = umy)
```

where we provide the output and the input, the delay, the orders of the AR and MA lag polynomials of the TF, and the UMs for the input and the output. The extended lag polynomials of the TF are stored into the data members `theta` and `phi`.

```
R> printLagpol(tfx$theta)

- 0.51 - 0.32B - 0.48B^2

R> printLagpol(tfx$phi)

1 - 0.65B + 0.087B^2
```

Box *et al.* (2016) fit the following TF model:

$$\begin{aligned} Y_t =& \frac{w_0 - w_1 B - w_2 B^2}{1 - \delta_1 B - \delta_w B^2} X_t + N_t, \\ N_t =& \frac{1}{1 - \phi_1 B - \phi_2 B^2} a_t, \end{aligned}$$

which can be estimated as follows:

```
R> tfmy <- tfm(Y, inputs = tfx, noise = um(ar = 2))
R> printLagpol(tfmy$inputs[[1]]$theta)

- 0.53 - 0.37B - 0.51B^2

R> printLagpol(tfmy$inputs[[1]]$phi)

1 - 0.56B + 0.011B^2

R> printLagpol(tfmy$noise$phi)

1 - 1.5B + 0.63B^2
```

where we can see that the estimates are very close to those reported by Box *et al.* (2016): $\hat{\omega}_0 = -0.53$, $\hat{\omega}_1 = 0.33$, $\hat{\omega}_2 = 0.51$, $\hat{\delta}_1 = 0.57$, $\hat{\delta}_2 = 0.02$, $\hat{\phi}_1 = 1.54$, $\hat{\phi}_2 = -0.64$.

It is worth mentioning that the `tfm()` function uses the backcasting method to extend the input using its own UM so that the effect of transients can be minimized. The number of backcasts is set by the `n.back` argument, which by default is n/4. The UM of the input is also used to compute forecasts when forecasting the output.

As a second illustration, Box *et al.* (2016, p. 409) consider a TF model to predict sales data using a leading indicator. Pankratz (1991, p. 202) estimated this model using the SCA system and the first 140 observations,

$$\begin{aligned} \nabla Y_t =& 0.033 + \frac{4.716}{1 - 0.725B} \nabla X_{t-3} + (1 - 0.620B)\hat{a}_t \\ \nabla X_t =& (1 - 0.4483B)\hat{b}_t. \end{aligned}$$

Before comparing these estimates with those provided by the `tfarima` package, we can compute the exact value of the concentrated likelihood function of this TF model. To this end, we first create the model without estimating the parameters and then we call the `logLik` function:

```
R> Y <- window(BJsales, end = 140)
R> X <- window(BJsales.lead, end = 140)
R> umy <- um(Y, i = 1, ma = "1-0.620B", mu = 0.033, fit = FALSE)
R> umx <- um(X, i = 1, ma = "1 - 0.4483B", fit = FALSE)
R> tfx <- tf(X, delay = 3, w0 = 4.716, ar = "1-0.725B", um = umx)
R> tfmy <- tfm(output = Y, inputs = tfx, noise = umy, fit = FALSE)
R> c(logLik(tfmy), logLik(umx))

[1]   7.157349 -23.832730
```

Without the previous information we would estimate the TF model for the sales data with the following code chunk:

```
R> umy <- um(Y, i = 1, ma = 1, mu = mean(diff(Y)))
R> umx <- um(X, i = 1, ma = "1-0.8B")
R> #pccf(Y, X, um.x = umx, lag.max = 20)
R> tfx <- tfest(Y, X, delay = 3, p = 1, q = 0, um.y = umy, um.x = umx)
R> tfmy <- tfm(output = Y, inputs = tfx, noise = umy)
R> c(coef(tfmy), lik = logLik(tfmy))

         X       X.d1         mu     theta1        lik
4.71938012 0.72564183 0.02572325 0.53875615 8.01622163

R> c(coef(umx), lik = logLik(umx))

     theta1         lik
  0.4480407 -23.8327217
```

Following the Box-Jenkins textbook example, forecasts from origin 89 for lead times from 1 to 10 are generated and shown in Figure 8.

```
R> p <- predict(tfmy, n.ahead = 10, ori = 89)
R> plot(p, n.back = 15, ylab = expression("Y"["t"]))
```

6. Intervention analysis and outlier detection

Intervention models suggested by Box and Tiao (1975) are special cases of TF models and can be estimated with the `tfm()` function to take care of outliers. For example, Box *et al.* (2016) identify three innovational outliers (IO) at times 58, 59 and 60 in the Series C, the

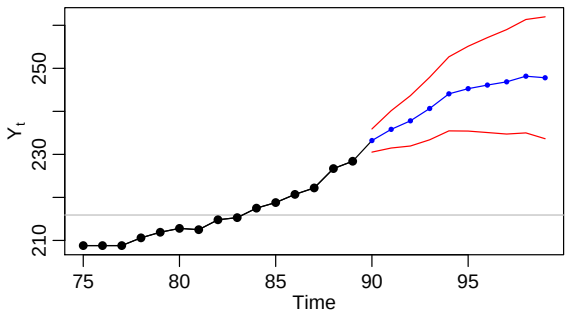

Figure 8: Forecast of sales at origin t = 89.

"uncontrolled" temperature readings every minute in a chemical process. To estimate the effects of these IOs they fit the outlier model:

$$(1-B)Y_t = \frac{1}{1-\phi B}(w_1 P_t^{(58)} + w_2 P_t^{(59)} + w_3 P_t^{(60)} + a_t),$$

where $P_t^{(\tau)}$ is a pulse function at τ:

$$P_t^{(\tau)} = \begin{cases} 0 & t \neq \tau \\ 1 & t = \tau \end{cases}$$

They estimated the model by conditional least squares (CLS) and obtained the following results (s.e. in parenthesis): $\hat{\phi} = 0.851(0.035)$, $\hat{\omega}_1 = 0.745(0.116)$, $\hat{\omega}_2 = -0.551(0.120)$, $\hat{\omega}_1 = -0.455(0.116)$, $\hat{\sigma}_a^2 = 0.0132$. This model can be reformulated as a single input transfer function model:

$$\begin{aligned} Y_t =& w_0 \frac{(1 - w_1' B - w_2' B^2)}{(1-\phi B)(1-B)} P_t^{(58)} + N_t \\ N_t =& \frac{1}{(1-\phi B)(1-B)} a_t, \end{aligned} \tag{11}$$

where the input and the noise share the same ARI operators. The capabilities of the **tfarima** package to estimate models with multiple operators and parameter restrictions allows us to estimate this model by running the following code:

```
R> Y <- as.ts(seriesC)
R> um1 <- um(Y, ar = 1, i = 1, method = "cond")
R> P58 <- InterventionVar(Y, 58)
R> tf58 <- tf(P58, ma = 2, ar = c(um1$ar, um1$i))
R> tfm1 <- tfm(inputs = tf58, noise = um1)
R> tfm1
```

```
        Estimate Std. Error
P58    0.7447334  0.1183491
P58.w1 0.7405461  0.2053821
P58.w2 0.6110603  0.1869052
```

```
phi1   0.8512211  0.0355065

log likelihood:  161.8301
Residual standard error:  0.1180356
aic: -1.402934
```

```
R> printLagpol(tfm1$inputs[[1]]$theta, digits = 3)

0.745 - 0.552B - 0.455B^2
```

Firstly, we load the Series C and fit an ARIMA(1,1,0) model, which is used as noise model. Next, we create a pulse variable at $\tau = 58$ and its transfer function by providing the order of the MA polynomial and the ARI operators of the `um1` model. Finally, we create and fit the transfer function model which is made up of a single input and an ARIMA(1,1,0) noise. We can see that the results of the estimation by conditional likelihood maximum are very similar to those reported by Box *et al.* (2016).

The `outliers()` function implements a version of the Chen and Liu (1993) procedure to detect and correct the effects of four common types of anomalies: additive outliers, innovational outliers, level shifts and transitory changes. We apply this function to model `um1` and use a critical value `c = 3.5` to determine if an observation is anomalous:

```
R> tfm2 <- outliers(um1, c = 3.5)
R> tfm2

       Estimate Std. Error
LS58  0.7037228 0.08971299
IO60 -0.4560648 0.11841531
phi1  0.8535311 0.03524750

log likelihood:  161.6899
Residual standard error:  0.1181278
aic: -1.410577
```

The function detects two outliers at times 58 and 60 of type LS and IO, respectively. Note that model `tfm2` is compatible with model `tfm1` since (11) can be reformulated as

$$Y_t = w_0 \frac{(1 - w_1' B)}{(1 - \phi B)} S_t^{(58)} - w_0 * w_2' P_t^{60} + N_t,$$

where $S_t^{58} = (1-B)^{-1} P_t^{58}$ is a step function at $T = 58$ and $w_1' \simeq \phi$. The `outliers()` function can also be used to identify and estimate the effects of some types of outliers at known times:

```
R> tfm3 <- outliers(um1, dates = c(58, 59, 60), types = c("AO", "LS"))
R> summary(tfm3)$table

       Estimate      Gradient Std. Error   z Value      Pr(>|z|)
LS58 0.90083708  2.764073e-08 0.10556934  8.533132  1.424382e-17
LS59 0.39900251  1.525865e-08 0.11016572  3.621839  2.925162e-04
AO60 0.09983522 -5.708279e-08 0.05530577  1.805150  7.105116e-02
phi1 0.84876909  1.157228e-07 0.03567670 23.790574 4.181158e-125
```

More information on the type of intervention needed at a known time can be obtained with the `intervantion()` function, which shows the estimates for one or several intervention variables

```
R> intervention(um1, type = c("AO", "LS", "TC", "IO"), time = 58)
```

```
                  AO58         LS58      TC58(1)         IO58
Estimate   2.676559e-01 7.044211e-01 7.036062e-01 7.617154e-01
Gradient   2.560198e-07 6.257278e-06 1.757167e-05 3.214669e-05
Std. Error 5.826450e-02 9.317969e-02 1.010188e-01 1.269253e-01
z Value    4.593808e+00 7.559814e+00 6.965101e+00 6.001288e+00
Pr(>|z|)   4.352294e-06 4.036472e-14 3.281681e-12 1.957585e-09
```

where the number between parenthesis next to the name of the TC variable is the estimate of the AR parameter. In this case, the temporary change reduces to a level shift.

7. Calendar effects

Hillmer (1982) used the telephone data of Thompson and Tiao (1971) to illustrate how to forecast time series with trading day variation. This data refers to outward station movements (disconnections) of the Wisconsin telephone company from January 1951 to October 1968. Clearly the series exhibits increasing seasonality and an upward trend requiring logs and both types of differences (regular and seasonal). Besides, it is affected by the number of trading days in a month because there are more disconnections on weekdays than on weekends. Due to these calendar effects, the sample ACF and PACF of $\nabla\nabla \log(Y_t)$ do not have a recognizable pattern. Hence, to identify a tentative ARMA structure for this series, the calendar effects are removed by fitting the regARIMA model

$$\log(Z_t) = TD_t + N_t, \quad \nabla\nabla_{12} N_t = a_t,$$

where

$$TD_t = \sum_{j=1}^{7} \alpha_j X_{jt} + N_t \tag{12}$$

and $X_{jt}, i = 1, ..., 7$, are, respectively, the number of Mondays, Tuesdays, and so on in month t and N_t is the noise. Defining $\beta_0 = \sum_{j=1}^{7} \alpha_j/7$ and $L_t = X_{1t} + \cdots + X_{7t}$ (length of month t), the regression component $TD_t = \sum_{j=1} \alpha_j X_{jt}$ can be reparametrized as

$$TD_t = \beta_0 L_t + \sum_{j=1}^{6} \beta_1 (X_{jt} - X_{7t}) \tag{13}$$

where $\beta_j = \alpha_j - \beta_0$ for $j = 1, \ldots, 6$.

The `calendar()` function enlarges an ARIMA or a TF model by adding calendar variables:

```
calendar <- function(mdl, z = NULL, form = c("td", "td7", "td6", "wd"),
         ref = 0, lom = TRUE, lpyear = TRUE, easter = FALSE, len = 4,
         easter.mon = FALSE, p.value = 1, n.ahead = 0, ...)
```

where `mdl` is an object of class `um` or `tfm` and `form` is a character indicating the type of representation for TD_t: "td", equation (13); "td7", equation (12); "td6", equation (13) without L_t; "wd", a variable for the 5 working days and another for the two weekend days. For representations "td" and "td6", the argument `ref` sets the reference day, by default it is Sunday. Instead of the length-of-month variable L_t, the leap year indicator can be included by setting `lpyear = TRUE` and `lom = FALSE`. Besides, Easter effects can be captured with an Easter variable whose values are the proportion of days of the Easter period ocurring in a particular month, ussually March and/or April. The duration of the Easter period is set with the `len` argument and can include the Easter Monday, `easter.mon = TRUE`. Other arguments are: `p.value`, used to remove the non-significant variables, and `n.ahead`, optional integer to extend the regressors with future observations to forecast the output.

To estimate this regARIMA model, we create an ARIMA(0,1,0)(0,1,0)$_{12}$ model without parameters and call the `calendar()` function, first with `form = "td"` and then with `form = "td7"`. It can be checked that both representations are equivalent.

```
R> Y <- Wtelephone$Y
R> umY <- um(Y, bc = TRUE, i = list(1, c(1,12)))
R> td <- calendar(umY, n.ahead = 13)
R> td7 <- calendar(umY, form = "td7", n.ahead = 13)
R> bTD <- coef(td); bTD7 = coef(td7)
R> cbind(bTD, cbind(c(bTD7[2:7]- mean(bTD7), mean(bTD7))))
```

```
                bTD
Mon_Sun  0.018741626  0.018741626
Tue_Sun  0.026319985  0.026319985
Wed_Sun  0.013239777  0.013239777
Thu_Sun  0.001581267  0.001581267
Fri_Sun  0.011129339  0.011129339
Sat_Sun -0.045970854 -0.045970854
lom     -0.025984496 -0.025984496
```

Alternatively, we could estimate this model generating the calendar variables and creating a `tfm` object:

```
R> X <- CalendarVar(Y, form = "td7", n.ahead = 13)
R> tfmY <- tfm(xreg = X, noise = umY)
```

where the `Y` argument in the `CalendarVar()` function is used to get the sample period.

We can recover the noise N_t or its transformations $\exp(N_t)$ and $\nabla\nabla_{12}N_t$ with the function

```
noise <- function(tfmY, diff = TRUE, exp = FALSE)
```

where `diff` is a logical value indicating if the noise must be differenced and `exp` is a logical value indicating if the antilog must be applied to the non-differenced noise. Note that these transformations are determined by the noise model.

Now we can identify a pattern in the ACF and PACF of the corrected series compatible with an ARIMA(0,1,1)(0,1,1)$_{12}$:

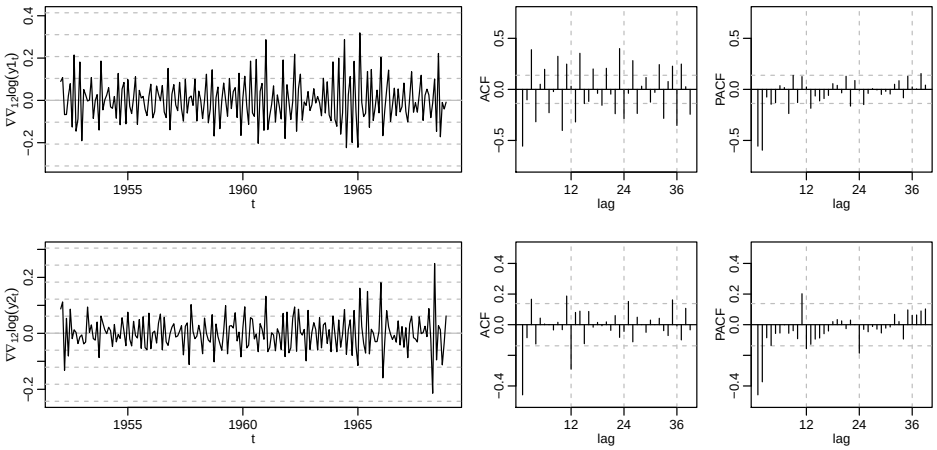

Figure 9: Identification tools for original and corrected telephone data.

```
R> y <- noise(tfmY, diff = FALSE, exp = TRUE)
R> ide(list(Y, y), transf = list(bc = T, d = 1, D = 1))
```

The two MA operators can be added to the `tfmY` object with the `modify()` function:

```
R> tfmY <- modify(tfmY, ma = list(1, c(1,12)))
R> tfmY
```

```
           Estimate Std. Error
Sun    -0.042666358 0.01809336
Mon     0.006352052 0.01706942
Tue     0.005981991 0.01810221
Wed     0.002218116 0.01728891
Thu    -0.019814682 0.01773134
Fri    -0.003301995 0.01755550
Sat    -0.065388393 0.01736859
theta1  0.742658333 0.04579664
theta2  0.456092445 0.06809570

log likelihood:  338.1755
Residual standard error:  0.04367511
aic: -3.259164
```

Finally, we could predict future values:

```
R> p <- predict(tfmY, n.ahead = 13)
R> print(p, rows = c(1, 13))
```

```
         Forecast       RMSE   95% LB   95% UB
dic 1968 18482.13 0.04396946 16956.07 20145.54
dic 1969 20113.80 0.06769674 17614.51 22967.71
```

For the sake of completeness, we estimate the ARIMA model developed by Thompson and Tiao (1971):

$$(1 - \phi_1 B^3)(1 - \phi_2 B^{12}) \log(Y_t) = (1 - \theta_9 B^9 - \theta_{12} B^{12} - \theta_{13} B^{13}) a_t$$

```
R> ma <- lagpol(c(theta9 = .2, theta12 = .2, theta13 = .2),
+               lags = c(9, 12, 13))
R> TT <- um(Y, bc = TRUE, ar = list(c(1, 3), c(1, 12)), ma = ma)
R> TT

           Estimate  Std. Error
phi1     0.94158802 0.030531610
phi2     0.99557584 0.004359192
theta9  -0.23538081 0.064989826
theta12  0.31958805 0.067679083
theta13  0.03334713 0.062176433

log likelihood:  235.9692
Residual standard error:  0.06568544
aic: -2.148551
```

8. Summary

We have presented some of the capabilities offered by the **tfarima** package to build customized TF and ARIMA models, which can include multiple conventional and user-defined lag polynomials. The package provides a full set of functions to apply the three stages of the Box-Jenkins methodology: identification, estimation and diagnosis of TF-ARIMA models, which can be used to forecast and decompose time series. Currently the package is been improved to include tests for a diversity of MA unit roots. A multivariate version of the package is also being developed.

References

Bell W (1987). "A Note on Overdifferencing and the Equivalence of Seasonal Time Series Models with Monthly Means and Models with (0,1,1)12 Seasonal Parts When $\theta = 1$." *Journal of Business & Economic Statistics*, **5**(3), 383–387. ISSN 07350015. URL http://www.jstor.org/stable/1391613.

Bell WR, Hillmer SC (1983). "Modeling Time Series With Calendar Variation." *Journal of the American Statistical Association*, **78**(383), 526–534. ISSN 01621459. URL http://www.jstor.org/stable/2288114.

Box GEP, Jenkins GM (1970). *Time Series Analysis: Forecasting and Control*. 2nd edition. Holden-Day, San Francisco, CA.

Box GEP, Jenkins GM, Reinsel GC, Ljung GM (2016). *Time Series Analysis: Forecasting and Control*. 5th edition. John Wiley and Sons Inc., Hoboken, New Jersey. ISBN 978-1-118-67502-1.

Box GEP, Pierce DA, Newbold P (1987). "Estimating Trend and Growth Rates in Seasonal Time Series." *Journal of the American Statistical Association*, **82**(397), 276–282.

Box GEP, Tiao GC (1975). "Intervention Analysis with Applications to Economic and Environmental Problems." *Journal of the American Statistical Association*, **70**(349), 70–79. URL https://www.tandfonline.com/doi/abs/10.1080/01621459.1975.10480264.

Chan KS, Ripley B (2018). *TSA: Time Series Analysis*. R package version 1.2, URL https://CRAN.R-project.org/package=TSA.

Chen C, Liu LM (1993). "Joint Estimation of Model Parameters and Outlier Effects in Time Series." *Journal of the American Statistical Association*, **88**(421), 284–297. URL https://doi.org/10.1080/01621459.1993.10594321.

Durbin J, Koopman S (2001). *Time series analysis by state space methods*. Oxford University Press, Oxford; New York.

Gallego J, Treadway A (1995). "The general family of seasonal stochastic procresses." Departamento de Economía, Universidad de Cantabria.

Harrison PJ, Stevens CF (1976). "Bayesian Forecasting." *Journal of the Royal Statistical Society. Series B (Methodological)*, **38**(3), 205–247. ISSN 00359246. URL http://www.jstor.org/stable/2984970.

Harvey AC, Durbin J (1986). "The effects of seat belt legislation on Brithish road casualties: a case study in structural time series modelling." *JRSSA*, **149**(4), 187–227.

Harvey AC, Todd P (1983). "Forecasting Economic Time Series with Structural and Box-Jenkins Models: A Case Study." *JBES*, **1**(4), 299–307.

Hillmer SC (1982). "Forecasting time series with trading day variation." *Journal of Forecasting*, **4**(1), 385–95.

la Tente AQ, Michalek A, Palate J, Baeyens R (2021). *RJDemetra: Interface to JDemetra+ Seasonal Adjustment Software*. Package Version 0.1.7, URL https://cran.r-project.org/web/packages/RJDemetra/index.html.

Ljung GM, Box GEP (1979). "The likelihood function of stationary autoregressive-moving average models." *Biometrika*, **66**(2), 265–270. URL https://doi.org/10.1093/biomet/66.2.265.

Pankratz A (1991). *Forecasting with Dynamic Regression Models*. John Wiley and Sons, New York.

R Core Team (2018). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Roberts SA (1982). "A General Class of Holt-Winters Type Forecasting Models." *Management Science*, **28**(7), 808–820. ISSN 00251909, 15265501. URL http://www.jstor.org/stable/2631264.

Sax C, Eddelbuettel D (2018). "Seasonal Adjustment by X-13ARIMA-SEATS in R." *Journal of Statistical Software*, **87**(11), 1–17. doi:10.18637/jss.v087.i11. URL https://www.jstatsoft.org/index.php/jss/article/view/v087i11.

Stoffer D (2021). *Applied Statistical Time Series Analysis*. R package version 1.14, URL https://CRAN.R-project.org/package=TSA.

Thompson H, Tiao GC (1971). "Analysis of Telephone Data: A Case Study of Forecasting Seasonal Time Series." **2**, 515–541.

Tsay RS, Wood D, Lachmann J (2022). *All-Purpose Toolkit for Analyzing Multivariate Time Series (MTS) and Estimating Multivariate Volatility Models*. R package version 1.1.1, URL https://CRAN.R-project.org/package=TSA.

Tusell F (2011). "Kalman Filtering in R." *Journal of Statistical Software*, **39**(2), 1–27. doi:10.18637/jss.v039.i02. URL https://www.jstatsoft.org/index.php/jss/article/view/v039i02.

Affiliation:

José Luis Gallego
Department of Economics
Faculty of Economics and Buisness
Universidad de Cantabria
Avda. de los Castros s/n
39005 Santander, Spain
E-mail: jose.gallego@unican.es