tsoutliers R Package for Detection of Outliers in Time Series

Javier López-de-Lacalle

https://jalobe.com

DRAFT VERSION: February, 2019

Abstract

Time series data often undergo sudden changes that alter the dynamics of the data transitory or permanently. These changes are typically non-systematic and cannot be captured by standard time series models. That's why they are known as exogenous or outlier effects. Detecting outliers is important because they have an impact on the selection of the model, the estimation of parameters and, consequently, on forecasts. An automatic procedure described in the literature to detect outliers in time series is implemented in the tsoutliers R package.

Key words: outliers, time series, R.

1 News

For the complete history see the NEWS file in the sources or here:

Main changes in version 0.6-7:

- The utilities for structural time series models have been disabled. The package stsm and the function KFKSDS::KF are no longer imported. Those utilities were an experimental version that extended the procedure for detection of outliers in ARIMA models to other models in state space form. If interested in the code, check previous versions or contact the maintainer.
- Fix in outliers.effects: computations were not correct for the case of innovational outliers.
- Tentative approach enabled by argument check.rank in discard.outliers in order to deal with perfect collinearity among the regressor variables.
- Further fixes in calendar.effects to make it independent of the locale.

Main changes in version 0.6-6:

- Fix in function `calendar.effects`. By using weekday as a decimal number instead of a character, the function now works regardless of the the locale in use.
- Fix in `locate.outliers.oloop`. If the series contains NAs, the NAs in the residuals that are passed to `outliers.tstatistics` are now imputed with the mean of the remaining residuals; before, those NAs were propagated throughout the t-statistics preventing the detection of outliers at some points.
- The function `remove.outliers` has been renamed as `discard.outliers`. The arguments `remove.method` and `remove.cval` in function `tso` have been renamed as `discard.method` and `discard.cval`. The old names still work (except in `tso0`), but a warning is given as they will be most likely be ignored in a future version.

Main changes in version 0.6-5:

- The outliers detected at each iteration of the procedure are checked so that outliers identified at consecutive time points are discarded keeping the outlier with the highest t-statistic in absolute value. This is done for all types of outliers, before it was done only for LS.
- The following change in R 3.2.1, *arima(*, xreg = .) (for d >= 1) computes estimated variances based on the number of effective observations as in R version 3.0.1 and earlier*, may lead to discrepancies in the results returned by `tso` compared with R version prior to 3.2.1 and later to 3.0.1.

2 Introduction

Time series data often undergo sudden changes that affect the dynamics of the data transitory or permanently. Sometimes, the practitioner may have some *a priori* information about the existence of these effects. A new regulation, strike periods, exceptional natural phenomena or a change in the methodology used to collect the data are some examples of events that may alter the overall dynamics of the data. In other cases, there is no such clear cut information available. The information may be unknown by the practitioner or it may not be feasible to carry out a review of all the historical events related to the context of the data. Regardless of their known or unknown source, these changes are typically non-systematic and cannot be captured by standard time series models (autoregressive integrated moving average models or structural time series models). A preliminary analysis of the data may reveal the existence of exceptional events that do not respond to patterns that can be captured by time series models, i.e. effects that are exogenous to the model. This analysis can be cumbersome or unfeasible if the data set is large. Moreover, sometimes these effects do not stand out in graphical representations or *ad hoc* auxiliar regressions and remain masked within the overall dynamics of the data. When some *a priori* knowledge about these effects exists, then a regressor variable can be explicitly defined and included in the model. These regressors are known as *intervention* variables. When this exogenous effect is not related to any known event then we say that there is an *outlier* in the time series. We consider five types of outliers: innovational outlier, additive outlier, level shift, temporary change and seasonal level shift. Sometimes when the time and shape of an outlier is detected an *ex post* explanation can be found to it. In that case, the general structure of the outlier regressor should be replaced by the corresponding intervention variable.

Detecting and correcting the effect of outliers is important because they have an impact on the selection of the model, the estimation of parameters and, consequently, on forecasts and other results pursued by the analysis such as seasonal adjustment. In practice, we observe that many time series are affected by outliers and they are more common than intervention variables, especially when a large an heterogeneous set of time series is analysed, as it is the case of the analyses carried out by National Institutes of Statistics and Central Banks.

An automatic procedure described in the literature to detect and adjust the series for outliers is implemented in the package `tsoutliers` of `R`.

The remaining of the paper is organised as follows. ARIMA and structural time series models are briefly introduced in Section 3. The types of outliers considered in `tsoutliers` are described in Section 4. Section 5 described the procedure to detect outliers. The performance of the implementation and a comparison with other specialised software is discussed by means of simulations exercises in Section 6. The usage of the package is illustrated with real examples in Section 7. Section 8 concludes.

3 Time series models

3.1 ARIMA time series models

A stationary ARMA model of orders (p, q) for a series y_t observed at time points $t = 1, 2, \dots, n$ can be defined as follows:

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + a_t + \sum_{i=1}^{q} \theta_i a_{t-i}\,, \quad \text{where } a_t \sim NID(0, \sigma_a^2)\,,$$

or more concisely

$$\phi(B) y_t = \theta(B) a_t\,, \quad \text{where } B \text{ is the lag operator.}$$

All the roots of the autoregressive polynomial, $\phi(B)$, and the moving average polynomial, $\theta(B)$, lie outside the unit circle. The parameters of the model $\alpha_{1,\dots,p}$, $\theta_{1,\dots,q}$ and σ^2_a can be estimated by maximum likelihood (see for instance [1], [2, Chapter 5], and [10, Chapter 22]).

For non-stationary data, the observed series y_t is transformed by means of a differencing filter and then an ARMA model is defined for the differenced data. An ARIMA model for the observed series y_t is defined as follows:

$$\phi(B)\alpha(B)y_t = \theta(B)a_t\,. \tag{1}$$

The differencing filter that renders the data stationary is given by $\alpha(B)$, which is an autoregressive polynomial that contains all its roots on the unit circle.

3.1.1 Automatic ARIMA model selection

The detection of outliers can be conducted along with the model selection procedure implemented in the `forecast` package. For details see [6, 11].

3.2 Structural time series models

The basic structural model (BSM) is a pure variance structural model commonly used in applications. This model plays a central role in the approach advocated in [5] for time series analysis. A detailed view of the features and theoretical properties of this model can be found, for instance, in [5, Chapter 2], [2, Chapter 8] and [4, § 3.2]. The model is defined as follows:

$$\begin{array}{rll}
\textit{observed series:} & y_t = \mu_t + \gamma_t + \epsilon_t, & \epsilon_t \sim \text{NID}(0,\,\sigma^2_\epsilon);\\
\textit{latent level:} & \mu_t = \mu_{t-1} + \beta_{t-1} + \xi_t, & \xi_t \sim \text{NID}(0,\,\sigma^2_\xi);\\
\textit{latent drift:} & \beta_t = \beta_{t-1} + \zeta_t, & \zeta_t \sim \text{NID}(0,\,\sigma^2_\zeta);\\
\textit{latent seasonal:} & \gamma_t = \sum_{j=1}^{s-1} -\gamma_{t-j} + \omega_t, & \omega_t \sim \text{NID}(0,\,\sigma^2_\omega),
\end{array}$$

for $t = s, \dots, n$, where s is the periodicity of the data.

The BSM encompasses models that are common in applications: the local level model, that consists of a random walk with a deterministic drift, β_0, plus a noise component, ϵ_t; the local trend model, where the drift follows a random walk. Setting $\sigma^2_\omega = 0$ yields a model with deterministic seasonality. Setting also $\gamma_1 = \ldots = \gamma_{s-1} = 0$ removes the seasonal component and gives the local trend model. Adding the restriction $\sigma^2_\zeta = 0$ yields the local level model.

The state space form of the BSM is given by the following representation:

$$\begin{array}{ll}
y_t = Z\alpha_t + \epsilon_t, & \epsilon_t \sim \text{NID}(0, \sigma^2_\epsilon),\\
\alpha_t = T\alpha_{t-1} + R\eta_t, & \eta_t \sim \text{NID}(0, V),\\
\alpha_0 \sim \text{N}(a_0, P_0), &
\end{array}
\quad \text{with } V = \begin{bmatrix} \sigma^2_\xi & 0 & 0 \\ 0 & \sigma^2_\zeta & 0 \\ 0 & 0 & \sigma^2_\omega \end{bmatrix},$$

for $t = 1, \ldots, n$. For $s = 4$, the matrices of this representation are defined as follows:

$$
\begin{aligned}
y_t &= \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \end{bmatrix} \alpha_t + \epsilon_t, \\
\alpha_t \equiv \begin{bmatrix} \mu_t \\ \beta_t \\ \gamma_t \\ \gamma_{t-1} \\ \gamma_{t-2} \end{bmatrix} &= \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} \mu_{t-1} \\ \beta_{t-1} \\ \gamma_{t-1} \\ \gamma_{t-2} \\ \gamma_{t-3} \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} \xi_t \\ \zeta_t \\ \omega_t \end{bmatrix}.
\end{aligned}
$$

The initial state vector, a_0, and its covariance, P_0, can be chosen beforehand or included in the set of parameters to be estimated. Given the variance parameters, the Kalman filter and smoother can be applied to extract an estimate of the the latent components (level, trend and seasonal).

4 Types of outliers

A general representation of an outlier can be given by the following expression: $L(B)I(t_j)$, where $L(B)$ is a polynomial of lag operators and $I(t_j)$ is an indicator variable that takes on the value 1 at time $t = j$ when the outlier springs and the value 0 elsewhere. We consider five types of outliers: innovational outlier (IO), additive outlier (AO), level shift (LS), temporary change (TC) and seasonal level shift (SLS). The polynomial $L(B)$ for each type of outlier is defined as follows:

$$
\begin{aligned}
&\text{IO:} \quad L(B) = \frac{\theta(B)}{\alpha(B)\phi(B)}\,; \quad \text{LS:} \quad L(B) = \frac{1}{(1-B)}\,; \quad \text{SLS:} \quad L(B) = \frac{1}{(1-B^s)}\,; \\
&\text{AO:} \quad L(B) = 1\,; \qquad\qquad \text{TC:} \quad L(B) = \frac{1}{(1-\delta B)}\,.
\end{aligned} \tag{2}
$$

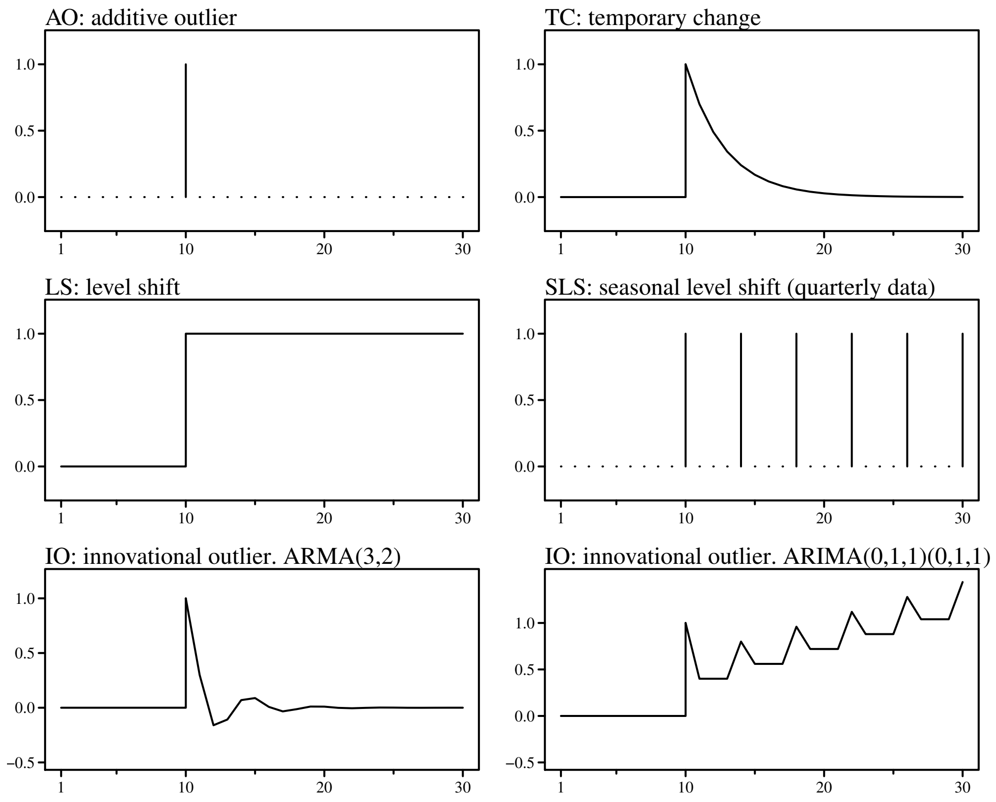
The value of δ is usually set equal to 0.7 and s is the periodicity of the data (e.g. 4 in quarterly data and 12 in monthly data). Each of them has a specific structure and influence on the data. Figure 1 shows a unit impulse for each type of outlier occurring at time point $t = 10$.

5 Automatic detection procedure

The procedure described in [3] for the automatic detection of outliers is implemented in the package `tsoutliers`. Next, we introduce the procedure following the notation of the original reference. At the same time, we give an insight into the implementation in package `tsoutliers` by showing and discussing the main parts of the code.

Let us define y_t^* as the observed series subject to m outliers with weights w. The ARIMA model

Figure 1: Unit impulse for different types of outliers



for the observed series is written as:

$$y_t^* = \sum_{j=1}^{m} \omega_j L_j(B) I_t(t_j) + \frac{\theta(B)}{\phi(B)\alpha(B)} a_t \,. \tag{3}$$

The estimated residuals, which are contaminated with the outliers, are given by:

$$\pi(B) y_t^* \equiv \hat{e}_t = \sum_{j=1}^{m} \omega_j \pi(B) L_j(B) I_t(t_j) + a_t \,, \tag{4}$$

where the coefficients of the power series expansion $\pi(B) = \sum_{i=0}^{\infty} \pi_i B^i$ can be determined from the relation (see for instance [2, Chapter 3]):

$$\pi(B) = \frac{\phi(B)\alpha(B)}{\theta(B)} \,.$$

Since all the roots of $\theta(B)$ lie outside the unit circle, the coefficients π_i satisfy $\sum_{i=0}^{\infty} |\pi_i| < \infty$ and vanish to zero.

Equations (3) and (4) constitute the grounds of the procedure for the detection of outliers, which consists of the following stages:

stage I: Locate outliers. Given an ARIMA model fitted to the data, outliers are detected and located by checking the significance of all types of outliers at all possible time points. This is done by means of the corresponding t-statistics in the regression equation (4).

stage II: Remove outliers. Given a set of potential outliers, an ARIMA model is chosen and fitted according to the regression equation (3). The significance of the outliers is reassessed in the new fitted model. If an ARIMA selection procedure is used a new model may be selected at this stage. Those outliers that are not significant are removed from the set of potential outliers.

stage III: Iterate stages I and II, first for the original series and then for the adjusted series.

5.1 Stage I: locate outliers

Replacing in equation (4) $L(B)$ by the definition of each type of outlier defined above and multiplying by $\pi(B)$, we can see that the equation becomes, respectively for each type of outlier:

$$\begin{aligned}
&\text{IO:} \quad \hat{e}_t = \omega I_t(t_j) + a_t\,, && \text{AO:} \quad \hat{e}_t = \omega \pi(B) I_t(t_j) + a_t\,,\\
&\text{LS:} \quad \hat{e}_t = \omega \frac{\pi(B)}{1-B} I_t(t_j) a_t\,, && \text{TC:} \quad \hat{e}_t = \omega \frac{\pi(B)}{1-\delta B} I_t(t_j) a_t\,.
\end{aligned} \tag{5}$$

[3] adopt the following t-statistics to identify and locate outliers:

$$\hat{\tau}_k = \frac{\hat{\omega}_k(t_j)}{\hat{\sigma}_a \left(\sum_{i=t_j}^{n} x_{k,i}^2 \right)^{1/2}}\,, \quad \text{with} \quad \hat{\omega}_k = \frac{\sum_{i=t_j}^{n} \hat{e}_t x_{k,i}}{\sum_{i=t_j}^{n} x_{k,i}^2}\,,$$

where k stands for each type of outlier, k = IO, AO, LS, TC. [3] derive the structure of each regressor $x_k = \pi(B) L_j(B) I_t(t_j)$ employed in equation (4). The residual standard deviation $\hat{\sigma}_a$ is estimated as the median absolute deviation (MAD) defined as:

$$\hat{\sigma}_a = 1.483 \times \text{median}\left(|\hat{e} - \tilde{e}|\right), \text{with } \tilde{e} = \text{median}(\hat{e})\,.$$

At first glance, the computation of all the τ statistics may seem more cumbersome than it actually is. In principle, the above regression equations should be run for all types of outliers and for all the potential locations of outliers $t_j = 1, 2, \ldots, n$. However, we can take advantage of the fact that, given a type of outlier, the regressors x_k for different time points t_j are shifted versions of x_k defined for $t_j = 1$. As we shall see, we do not need to explicitly build all the regressors and run a regression for each time point t_j and for each type of outlier.

The code below loads the required packages and generates a series from an ARIMA(1,0,1) model with three exogenous shocks (two additive outliers and one level shift). In the sequel, we will use this simulated series to illustrate the procedure.

```
library("tsoutliers")
set.seed(123)
```

```
y <- arima.sim(model = list(ar = 0.7, ma = -0.4), n = 120)
y[15] <- -4
y[45] <- 5
y[80:120] <- y[80:120] + 5
y <- round(y, 2)
```

Next, we start the procedure. First, an ARIMA model is chosen and fitted by means of the function `auto.arima` available in the package `forecast`. No drift is included in the model and the Bayesian information criterion (BIC) is used to select the model. The autoregressive and moving average coefficients as well as the residuals and its length are extracted from the fitted model. The standard deviation of residuals, $\hat{\sigma}_a$, is computed by means of the MAD method defined above.

```
fit <- forecast::auto.arima(x = y, allowdrift = FALSE, ic = "bic")
pars <- coefs2poly(fit)
resid <- residuals(fit)
n <- length(resid)
sigma <- 1.483 * quantile(abs(resid - quantile(resid, probs = 0.5)), probs = 0.5)
```

Upon the information obtained from the fitted model we can compute the τ statistics as follows. The coefficients of $\pi(B)$ are obtained by means of the function `ARMAtoMA` from the `stats` package. Notice that this function computes the coefficients in the relation $\psi(B) = \frac{\theta(B)}{\phi(B)} = \frac{1}{\pi(B)}$. After changing the sign of the parameters and passing the AR coefficients as the MA coefficients and vice versa, this function returns the coefficients of the polynomial $\pi(B)$. Looking at equations (5), we can see that the τ statistics related to the innovational outlier can be easily obtained as the residuals divided by $\hat{\sigma}_a$.

```
picoefs <- c(1, ARMAtoMA(ar = -pars$macoefs, ma = -pars$arcoefs, lag.max = n-1))
tauIO <- resid / sigma
```

The τ statistics for the AO outliers are obtained taking advantage of the fact that the regressors involved in equation (5) for each time point $t = t_j$ are shifted versions of the case $I(t_j = 1)$.

```
padded.resid <- c(resid, rep(0, n-1))
xy <- as.vector(na.omit(filter(x = padded.resid,
  filter = rev(picoefs), method = "conv", sides = 1)))
xx <- rev(cumsum(picoefs^2))
tauAO <- xy / (sigma * sqrt(xx))
```

Similarly, we obtain the τ statistics related to the LS and TC after filtering $\pi(B)$ through $1/(1-B)$ and $1/(1-\delta B)$, respectively.

```
di.picoefs <- diffinv(picoefs)[-1]
xy <- as.vector(na.omit(filter(x = padded.resid,
  filter = rev(di.picoefs), method = "conv", sides = 1)))
```

```
xx <- rev(cumsum(di.picoefs^2))
tauLS <- xy / (sigma * sqrt(xx))
delta <- 0.7
didelta.picoefs <- filter(x = picoefs, filter = delta, method = "rec")
xy <- as.vector(na.omit(filter(x = padded.resid,
  filter = rev(didelta.picoefs), method = "conv", sides = 1)))
xx <- rev(cumsum(didelta.picoefs^2))
tauTC <- xy / (sigma * sqrt(xx))
```

We have seen that the computation of all the τ statistics involves fewer computations than may seem necessary at first glance and it can be done without explicit loops. Now, let's look at some of the values that we obtained for these statistics.

```
tau.stats <- data.frame(IO = tauIO, AO = tauAO,
  LS = tauLS, TC = tauTC, row.names = NULL)
round(tau.stats[c(14:16,44:46,78:82),], 3)
       IO     AO     LS     TC
14  1.119  1.386  0.105 -0.406
15 -4.103 -4.797 -0.930 -2.397
16  2.322  1.613  2.655  2.865
44 -0.535 -1.096  0.786  1.245
45  4.934  5.517  1.605  3.216
46 -2.883 -2.405 -2.518 -2.640
78  1.755 -0.028  4.411  1.595
79  1.215 -0.734  4.432  2.316
80  4.325  2.984  4.981  4.271
81  1.958  1.093  2.751  2.189
82  1.231  0.582  1.934  1.695
```

Those statistics that are in absolute value higher than a threshold identify the time point of a potential outlier. The code below identifies those statistics that are higher than the threshold in absolute value. Here, we use the threshold or critical value `cval` $= 3.5$. The potential outliers are stored in a `data.frame` named `mo` containing the type of outlier, the observation index and the t-statistic.

```
cval <- 3.5
indIO <- which(abs(tauIO) > cval)
indAO <- which(abs(tauAO) > cval)
indLS <- which(abs(tauLS) > cval)
indTC <- which(abs(tauTC) > cval)
mo <- data.frame(type = factor(c(rep("IO", length(indIO)),
  rep("AO", length(indAO)), rep("LS", length(indLS)), rep("TC", length(indTC))),
  levels = c("IO", "AO", "LS", "TC")),
  ind = c(indIO, indAO, indLS, indTC),
  tstat = c(tauIO[indIO], tauAO[indAO], tauLS[indLS], tauTC[indTC]))
cbind(mo[order(mo[,"ind"])[1:5],], " " = "    ", mo[order(mo[,"ind"])[6:10],])
  type ind     tstat      type ind    tstat
```

```
1   IO  15 -4.102540        LS  79 4.432065
4   AO  15 -4.797319        IO  80 4.325142
2   IO  45  4.933745        LS  80 4.980832
5   AO  45  5.517405        TC  80 4.271211
6   LS  78  4.410770      <NA>  NA       NA
```

We can see that, for a given time point, there are more than one type outlier exceeding the threshold (e.g. at $t = 15$, $\tau_{IO} = -4.103$ and $\tau_{AO} = -4.797$). Duplicates are removed keeping the type of outlier with the highest t-statistic in absolute value.

```
for (i in mo[,"ind"][duplicated(mo[,"ind"])])
{
  ind <- which(mo[,"ind"] == i)
  moind <- mo[ind,]
  if ("IO" %in% moind[,"type"]) {
    tmp <- moind[which(moind[,"type"] != "IO"),]
  } else
    tmp <- moind[which.max(abs(moind[,"tstat"])),]
  mo <- mo[-ind,]
  mo <- rbind(mo, tmp)
}
mo[order(mo[,"ind"]),]
  type ind     tstat
4   AO  15 -4.797319
5   AO  45  5.517405
6   LS  78  4.410770
7   LS  79  4.432065
8   LS  80  4.980832
```

The function `locate.outliers` performs the location of outliers in the lines shown above except that it adds one additional rule: when outliers of a given type are identified at consecutive time points, then only the outlier related to the highest τ statistic is kept and the remaining are discarded. Thus, if we run the function, we obtain the result stored above in object `mo` except that the LS at times 78 and 79 are not included since $|4.411| < |4.980|$ and $|4.432| < |4.980|$.

```
locate.outliers(resid, pars, cval, types = c("IO", "AO", "LS", "TC"))
  type ind   coefhat     tstat
4   AO  15 -4.450352 -4.797319
5   AO  45  5.118357  5.517405
6   LS  78  3.057720  4.410770
7   LS  79  3.072482  4.432065
8   LS  80  3.452909  4.980832
```

The function `locate.outliers` is the baseline of stage I of the procedure. The complete stage consists of an outer loop and an inner loop. The inner loop is implemented in the function `locate.outliers.iloop` and can be sketched as follows:

```
mo.iloop <- NULL
maxit.iloop <- 4
iter <- 0
resid.iloop <- resid
while (iter < maxit.iloop)
{
  mo <- locate.outliers(resid.iloop, pars, cval, types = c("IO", "AO", "LS", "TC"))
  if (nrow(mo) == 0)
    break
  mo.iloop <- rbind(mo.iloop, mo)
  oxreg <- outliers.regressors(pars, mo, n, weights = TRUE)
  resid.iloop <- resid.iloop - rowSums(oxreg)
  iter <- iter + 1
}
print(mo.iloop)
    type ind    coefhat      tstat
4     AO  15  -4.450352  -4.797319
5     AO  45   5.118357   5.517405
6     LS  78   3.057720   4.410770
7     LS  79   3.072482   4.432065
8     LS  80   3.452909   4.980832
2     LS  77  -3.352594  -4.891855
3     LS  78  -4.008420  -5.848787
41    LS  79  -4.692708  -6.847250
51    LS  80  -3.803128  -5.549241
61    LS  81  -3.322974  -4.848636
71    LS  82  -2.428990  -3.544202
13    LS  76   6.276928   8.495680
14    LS  77   8.120745  10.991243
15    LS  78   9.674657  13.094428
16    LS  79  10.008111  13.545751
17    LS  80  10.377375  14.045541
18    LS  81   9.335869  12.635886
19    LS  82   7.862332  10.641488
20    LS  83   5.740041   7.769014
21    LS  84   3.765332   5.096291
22    TC  72  -3.852159  -4.245383
23    TC  73  -5.026003  -5.539050
24    TC  74  -5.777463  -6.367218
25    TC  75  -5.716360  -6.299879
35    LS  76 -23.413614 -32.311913
36    LS  77 -30.170841 -41.637211
37    LS  78 -34.050403 -46.991193
38    LS  79 -36.103541 -49.824623
39    LS  80 -35.508027 -49.002784
40    LS  81 -33.718535 -46.533200
411   LS  82 -30.339489 -41.869954
42    LS  83 -26.181571 -36.131827
43    LS  84 -21.279471 -29.366693
44    LS  85 -15.752958 -21.739839
```

```
45    LS  86 -11.707736 -16.157239
46    LS  87  -8.392412 -11.581932
47    LS  88  -6.377917  -8.801832
48    LS  89  -4.589172  -6.333278
49    LS  90  -2.583970  -3.566003
50    TC  67   3.466848   3.895746
511   TC  68   5.288947   5.943265
52    TC  69   6.938082   7.796421
53    TC  70  10.039979  11.282067
54    TC  71  15.776938  17.728770
55    TC  72  22.560787  25.351877
56    TC  73  26.918399  30.248587
57    TC  74  27.812573  31.253383
58    TC  75  23.253686  26.130497
cat(sprintf("number of iterations: %s\n", iter))
number of iterations: 4
```

First, the following objects are defined: `mo.iloop` stores the potential set of outliers; `maxit.iloop` is the maximum number of allowed iterations; `iter` is the counter; and `resid.iloop` stores the residuals adjusted for the outlier effects. Then the inner loop begins. At each iteration the effect of the outliers on the residuals is removed and a new check for the presence of outliers is carried out. The function `outliers.regressors` builds the regressors x_k from equation (5) weighted for the parameter estimates (column `"coefhat"` in object `mo`). The inner loop stops when no additional outliers are detected or when a maximum number of allowed iterations is reached. The output shows that only one iteration was required to identify outlying observations.

The outer loop is implemented in the function `locate.outliers.oloop`. The main parts of the code are shown below. First, some variables are created: `y.oloop` and `fit.oloop` to avoid overwriting the objects `y` and `fit`; `moall`, the object that will store information about the outliers; `maxit.oloop` maximum number of allowed iterations and the counter `iter`.

```
y.oloop <- y
fit.oloop <- fit
moall <- NULL
maxit.oloop <- 4
iter <- 0
while (iter < maxit.oloop)
{
  pars <- coefs2poly(fit)
  resid.oloop <- residuals(fit.oloop)
  resid.med <- quantile(resid.oloop, probs = 0.5)
  sigma <- 1.483 * quantile(abs(resid.oloop - resid.med), probs = 0.5)
  mo.iloop <- locate.outliers.iloop(resid.oloop, pars, cval,
    types = c("IO", "AO", "LS", "TC"), maxit = maxit.iloop)
  if (nrow(mo.iloop) == 0)
```

```
    break
  moall <- rbind(moall, mo.iloop)
  oeff <- outliers.effects(mo.iloop, n, weights = TRUE, pars = pars)
  y.oloop <- y.oloop - rowSums(oeff)
  fit.oloop <- arima(y.oloop, order = fit$arma[c(1,6,2)],
    seasonal = list(order = fit$arma[c(3,7,4)]))
  iter <- iter + 1
}
print(moall)
  type ind   coefhat     tstat
4   AO  15 -4.450352 -4.797319
5   AO  45  5.118357  5.517405
8   LS  80  3.452909  4.980832
cat(sprintf("number of iterations: %s\n", iter))
number of iterations: 1
```

At each iteration, the required information is extracted from the last fitted model (parameter estimates, residuals and MAD of residuals). The inner loop is run to identify outliers and then the model is fitted again for the series adjusted for the outliers identified at the current iteration. The function `outliers.effects` creates a matrix of outliers effects as shown in Figure 1; `weights` is set to `TRUE` in order to weight each outlier by the estimated coefficient instead of using a unit impulse. The loop stops when no more outliers are detected by the inner loop or when the maximum number of allowed iterations is reached. The final set of outliers is printed. We can see that it is in agreement with the relatively prominent shocks added to the simulated series.

For completeness, we show how a call to the function `locate.outliers.oloop` would be defined to run the inner and outer loops and obtain the results shown above.

```
stage1 <- locate.outliers.oloop(y, fit, types = c("IO", "AO", "LS", "TC"),
  cval = cval, maxit.iloop = maxit.iloop)
stage1$outliers
  type ind   coefhat     tstat
4   AO  15 -4.450352 -4.797319
5   AO  45  5.118357  5.517405
8   LS  80  3.452909  4.980832
```

5.2 Stage II: remove outliers

Although the detection of outliers is already made in stage I, we must be aware that the initial selection of the ARIMA model may be affected by the presence of outliers (especially the order of integration) and vice versa, that is, the detection of outliers may be affected by the choice of the model and the parameter estimates based on the observed series.

To avoid finding spurious outliers caused by an improper choice or estimation of a model, the equation (3) is fitted for the observed series including the outlier regressors identified in stage I. If any of the outliers turn to be non-significant then they are removed from the set of potential outliers. The function `discard.outliers` performs this operation as follows.

```
moall <- stage1$outliers
xreg <- outliers.effects(moall, n, weights = FALSE, pars = pars)
iter <- 0
while (TRUE)
{
  fit <- forecast::auto.arima(x = y, xreg = xreg, allowdrift = FALSE, ic = "bic")
  xregcoefs <- coef(fit)
  ind <- match(colnames(xreg), names(xregcoefs))
  xregcoefs <- xregcoefs[ind]
  tstats <- xregcoefs / sqrt(diag(fit$var.coef)[ind])
  ref <- which(abs(tstats) < cval)
  if (length(ref) > 0)
  {
    moall <- moall[-ref,]
    xreg <- data.matrix(xreg[,-ref])
  } else
    break
  if (nrow(moall) == 0)
    break
  iter  <- iter + 1
}
```

The `data.frame` `moall` contains information about the set of outliers detected in stage I. The function `outliers.effects` builds the regressors and stores them in the object `xreg`. Notice that here the regressors are not the variables x_k used in stage I but a variable capturing the exogenous effect according to each type of outlier as was defined in equation (2) and depicted in Figure 1. An ARIMA model is selected again for the observed data by means of `forecast::auto.arima` including the object `xreg` as regressor variables. Then the t-statistics are computed and stored in `tstats`. If any of them is below the threshold (in absolute value) then the corresponding outlier is removed from `moall`. The process is iterated until all the outliers are significant in the last model selected for the data. This process to discard outliers corresponds to the option `method = "en-masse"` in function `discard.outliers`. Alternatively, the significance of the outliers can be tested in a botton-up procedure (see the documentation of the function).

A summary output for this stage is printed below.

```
print(fit)
Series:
Regression with ARIMA(1,0,0) errors
```

```
Coefficients:
         ar1     AO15    AO45    LS80
      0.3023  -4.6067  5.4875  4.6667
s.e.  0.0909   0.8736  0.8689  0.1986

sigma^2 estimated as 0.8356:  log likelihood=-157.51
AIC=325.02   AICc=325.55   BIC=338.96
moall[,"coefhat"] <- xregcoefs
moall[,"tstat"] <- tstats
print(moall)
  type ind   coefhat     tstat
4   AO  15 -4.606657 -5.273256
5   AO  45  5.487542  6.315486
8   LS  80  4.666688 23.492144
```

Despite no outliers were removed, a different model is now selected. Although we did not print it, the model chosen at the beginning of stage I in the previous subsection was an ARIMA$(0,1,1)$, now an ARIMA$(1,0,0)$ is chosen. When the level shift is not considered, the data cannot be modelled by a stationary ARMA model. Despite there is a deterministic level shift a stochastic trend is considered and the first order differencing filter is applied. After detecting and including the effect of the level shift, its effect is estimated and a stationary ARMA model is chosen by the automatic model selection procedure.

5.3 Stage III: iterate stages I and II for the adjusted series

Here, the stages I and II are iterated for the adjusted series. If new outliers are found they are added to the set of outliers found in the previous iteration.

The original series can be adjusted for outliers defined in `moall` as follows. The function `outliers.effects` builds the outlier variables according to each type. The total outliers effect is measured and stored in `oeff`. The series adjusted for outliers is stored in `yadj` (the linearized series).

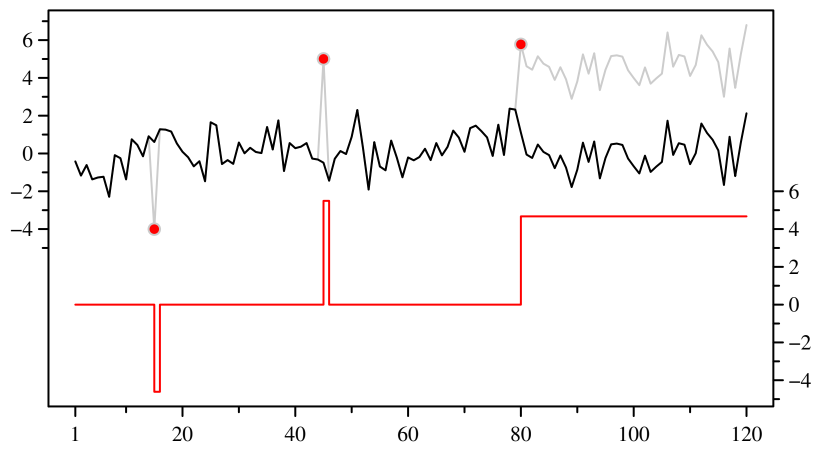
```
tmp <- coefs2poly(fit)
xreg <- outliers.effects(moall, n, weights = FALSE, pars = tmp)
xregcoefs <- coef(fit)
ind <- match(colnames(xreg), names(xregcoefs))
xregcoefs <- xregcoefs[ind]
oeff <- xreg %*% cbind(xregcoefs)
yadj <- y - oeff
plot(y, col = "gray80")
lines(yadj)
```

After choosing a model for the adjusted series and calling directly the function `locate.outliers`, we see that no additional outliers are detected and thus the procedure is finished. Figure 2 shows the original data (gray line), the adjusted series (black line), the location of the detected outliers (red points) and their estimated effects (red line).

```
fit <- forecast::auto.arima(x = yadj, allowdrift = FALSE, ic = "bic")
tmp <- coefs2poly(fit)
resid <- residuals(fit)
sigma <- 1.483 * quantile(abs(resid - quantile(resid, probs = 0.5)), probs = 0.5)
locate.outliers(resid, pars, cval, types = c("IO", "AO", "LS", "TC"))
[1] type    ind     coefhat tstat
<0 rows> (or 0-length row.names)
res <- tso(y, cval = cval, types = c("IO", "AO", "LS", "TC"),
  maxit = 1, maxit.iloop = 4)
postscript(file = file.path("figures", "fig-yadj.eps"), horizontal = FALSE,
  family = "Times", paper = "special", width = 4.3, height = 2.6)
par(mar = c(1.7, 1.5, 1, 1.1), las = 1)
#plot(res, args.plot = list(main = NULL), offset = -2.6)
tmp0 <- tmp <- cbind(res$fit$x, res$yadj)
mintmp <- min(tmp)
sdtmp <- sd(tmp)
tmp[,1] <- tmp[,1] / sdtmp
tmp[,2] <- tmp[,2] / sdtmp
xeffects <- res$effects / sdtmp
plot(tmp, plot.type = "single", type = "n", xaxt = "n", yaxt = "n",
  ylim = c(mintmp - abs(min(xeffects)) - 0.0 * sdtmp, max(tmp)))
lines(tmp[,1], col = "gray80")
lines(tmp[,2])
lines(xeffects - abs(mintmp) - 0.0 * sdtmp, col = "red", type = "s")
points(x = res$times, y = tmp[,1][res$outliers[,"ind"]],
  col = "gray80", bg = "red", pch = 21, cex = 0.8)
#ax <- pretty(time(tmp[,1]))
ax <- c(1, seq.int(20, 120 , 20))
axis(side = 1, at = ax, labels = FALSE, tcl = -0.25, lwd = 0, lwd.tick = 1)
axis(side = 1, at = ax, labels = ax, lwd = 0, lwd.tick = 0, line = -0.9, cex.axis = 0.7)
axis(side = 1, at = ax[-1] - (ax[3] - ax[2])/2, labels = FALSE, tcl = -0.15, lwd = 0, lwd.tick = 1
aylabels <- pretty(tmp0[,1])
ay <- round(aylabels / sdtmp)
axis(side = 2, at = ay, labels = FALSE, tcl = -0.25, lwd = 0, lwd.tick = 1)
axis(side = 2, at = ay, labels = aylabels, lwd = 0, lwd.tick = 0, line = -0.6, cex.axis = 0.7)
axis(side = 2, at = ay-(ay[2] - ay[1])/2, labels = FALSE, tcl = -0.15, lwd = 0, lwd.tick = 1)
aylabels <- pretty(res$effects)
ay <- round(aylabels / sdtmp) - abs(mintmp) - 0.0 * sdtmp
axis(side = 4, at = ay, labels = FALSE, tcl = -0.25, lwd = 0, lwd.tick = 1)
axis(side = 4, at = ay, labels = aylabels, lwd = 0, lwd.tick = 0, line = -0.6, cex.axis = 0.7)
axis(side = 4, at = ay-(ay[2] - ay[1])/2, labels = FALSE, tcl = -0.15, lwd = 0, lwd.tick = 1)
invisible(dev.off())
```

In most cases, the user will be interested in running the procedure in one call rather than by stages as we did here for illustration. That's precisely the point of an automatic procedure. The function `tso0` runs stages I and II and the function `tsoutliers` iterates the procedure running `tso0` first for the original series and then for the adjusted series. These functions are relatively flexible and allows the user to control several parameters of the procedure such as the maximum number of iterations in the loops, the threshold and the arguments passed to `forecast::auto.arima` to select the ARIMA model, among other parameters.

Figure 2: Original series, adjusted series and estimated outlier effects



5.4 Polishing rules

In practice, we may encounter some series for which the procedure as described above would not return a reliable result. If the initial model and parameter estimates do not fit well the overall dynamics of the data, the procedure may find too many outliers or the estimate of the outlier effects may not be accurate enough. In this case, the correction for outliers may induce some artificial effects in the adjusted series that may in turn be identified as outliers in the next run of the procedure. Next, we discuss some polishing rules that are included in the code of package `tsoutliers`. Some of them have already been mentioned above.

- By default, the threshold against which the t-statistics are compared is determined based on the sample size, n. `cval` $= 3$ if $n \leq 50$; `cval` $= 4$ if $n \geq 450$; `cval` $= 3 + 0.0025 \times (n - 50)$ otherwise.

- In the adjusted series, which is expected to be cleaner from outlying observations, the threshold or critical value, `cval`, can be reduced by a factor `cval.reduce` so that the threshold `cval` $\times$ (1 $-$ `PC`) is used.

- In a previous version of the package this rule was applied: The outliers AO, LS, and TC have precedence over IO when the t-statistic related to more than one type of outlier exceed the threshold `cval` at a given time point, i.e., the IO is ignored and the other type of outlier is kept. In the current version this rule is not applied but the IO type of outlier is not considered by default.

- If at a given iteration of the inner or outer loop an outlier is found at a time point where an outlier was already detected in a previous iteration, then the type of outlier that was first detected is kept.

- If the t-statistics related to a given type of outlier are beyond the critical value for consecutive time points, only the point with the highest t-statistic is kept.

- The inner and outer loops of stage I are stopped if a predefined maximum number of iterations is reached. This does not seem necessary for the outer loop but a limit is set for safety.

- When regular or seasonal differences are taken, the residuals related to the first observations may be overly erratic. If the maximum value of the first residuals are beyond a threshold then they are set to zero in function `locate.outliers.oloop`. In this case, the first observations are not checked for the presence of outliers. The number of observations that may be omitted is $d + D \times s$, where d is the number of regular differences, D is the number of seasonal differences and s is the periodicity of the data. The threshold is defined as 3.5 times the standard deviation of the remaining residuals.

5.5 Adaptation of the procedure to structural time series models

The procedure to detect outliers can be adapted to the framework of the basic structural model introduced in § 3.2.

The t-statistics related to IO are computed as in the ARIMA model since the regressor is a unit impulse variable.

```
tauIO.stsm <- resid / sigma
```

The object `resid` is the residuals from structural model and `sigma` is the MAD of the residuals that was defined before.

The statistics for AO are computed similarly as in the ARIMA model. Here, the Kalman filter is run taking as input the indicator variable $I(1)$. To avoid the problems inherent to the initialization of the Kalman filter, `n.start` additional zeros are included in the indicator variable. They act as warming observations and are removed afterwards from the final vector of coefficients `f`. The computations for the remaining time points take advantage of the fact that the regressors are shifted versions of the regressor obtained for $I(1)$, as we did in the ARIMA case.

```
n.start <- 50
I <- ts(rep(0, length(resid) + n.start - 1),
  start = start(resid), frequency = frequency(resid))
I[n.start] <- 1
tmp <- KFKSDS::KF(I, ss)
f <- tmp$v[-seq.int(n.start-1)]
ao.xy <- as.vector(na.omit(
  filter(x = c(resid, rep(0, n-1)), filter = rev(f), method = "conv", sides = 1)))
ao.xx <- rev(cumsum(f^2))
tauAO.stsm <- ao.xy / (sigma * sqrt(ao.xx))
```

The object `ss` is a list containing the matrices of the state space representation of the structural model. If the package `stsm` [9] was used to fit the model, the function `stsm.class::char2numeric` can be used to easily define this object.

The statistics for the LS reuse the values obtained above and stored in `ao.xy`. The inverse of the difference filter is applied on it and then the statistics are computed as usual.

```
ls.xy <- rev(diffinv(rev(ao.xy))[-1])
ls.xx <- rev(cumsum(diffinv(f)[-1]^2))
tauLS.stsm <- ls.xy / (sigma * sqrt(ls.xx))
```

The operations for the TC are the same as for LS except that instead of the inverse of the differencing filter $(1 - L)$, the inverse of $(1 - \delta L)$ is applied.

```
tc.xy <- rev(filter(x = rev(ao.xy), filter = delta, method = "rec"))
dinvf <- filter(x = f, filter = delta, method = "rec")
tc.xx <- rev(cumsum(dinvf^2))
tauTC.stsm <- tc.xy / (sigma * sqrt(tc.xx))
```

Given the t-statistics, the automatic detection procedure can be applied as described in the previous section, except that the structural model is fitted instead of an ARIMA model. Notice also that since the components of the structural model are defined beforehand, there is no need no reassess the choice of the model but only to refit the parameters of the model.

6 Simulation exercises

The results have changed with respect to version 0.6-4, the main reason is apparently this change in R 3.2.1: with `arima(*, xreg = .)  (for d >= 1)`, the number of effective observations used to compute the standard errors of parameter estimates has been modified as reported in PR#16278.

[Table 1 about here.]

[Table 2 about here.]

7 Applications to real data

7.1 Two popular time series

Time series methods described in the literature are often applied to two popular time series: 1) the Nile time series, measurements of the annual flow of the river Nile at Ashwan in the period 1871-1970 and 2) the classic Box & Jenkins airline data, monthly totals of international airline passengers in the period 1949 to 1960. These data are available in the `datasets` package that comes with the base R system.

The Nile time series is a common example of the the local level model, the simplest structural time series model. Next, we run `tso` for the local level model. As of version 0.6-7 the experimental version for structural time series model is not available. Check previous versions of the package or contact the maintainer for details. For illustration, these are the results that were obtained in previous versions for the local level model.

```
resNile1 <- tso(y = Nile, types = c("AO", "LS", "TC"),
  tsmethod = "stsm", args.tsmodel = list(model = "local-level"))
resNile1$fit$call$xreg<-NULL
resNile1

Call:
structure(list(method = "L-BFGS-B"), .Names = "method")


Parameter estimates:
               LS29   var1  var2
Estimate    -247.78  16136     0
Std. error    11.71   1163   NaN


Log-likelihood: -633.0286
Convergence: 0
```

```
Number of iterations: 46 46

Variance-covariance matrix: optimHessian


Outliers:

  type ind time coefhat  tstat

1   LS  29 1899  -247.8 -21.16

Warning messages:

1: In sqrt(diag(solve(res$hessian))) : NaNs produced

2: In sqrt(diag(solve(res$hessian))) : NaNs produced

3: In sqrt(diag(solve(res$hessian))) : NaNs produced

4: In sqrt(diag(solve(res$hessian))) : NaNs produced

5: In sqrt(diag(solve(res$hessian))) : NaNs produced
```

We can see that a level shift is detected as observation 29 (year 1899). More importantly, the variance of the level component is estimated to be *zero* when this level shift is included in the model. Thus, as it has been suggested by others, the Nile time series is well described by a level shift plus a white noise disturbance. No other structure seems to be present in the data. Similar conclusions are found in the ARIMA framework:

```
resNile2 <- tso(y = Nile, types = c("AO", "LS", "TC"),
  maxit = 1, discard.method = "bottom-up", tsmethod = "auto.arima",
  args.tsmethod = list(allowdrift = FALSE, ic = "bic"))
resNile2
Series: Nile
Regression with ARIMA(0,0,0) errors

Coefficients:
      intercept       LS29       AO43
      1097.7500  -242.2289  -399.5211
s.e.    22.6783    26.7793   120.8446

sigma^2 estimated as 14846:  log likelihood=-620.65
AIC=1249.29   AICc=1249.71   BIC=1259.71

Outliers:
  type ind time coefhat  tstat
1   LS  29 1899  -242.2 -9.045
2   AO  43 1913  -399.5 -3.306
```

A strong level shift is detected at observation 29 (year 1899), which is also suggested by the graphical representation of the series (not shown here). Curiously enough, the model selection procedure did not find any ARIMA structure when the level shift is detected and included in the model.

The model ARIMA(0,1,1)(0,1,1) has been found to fit well to many economic time series. This model is often called the airlines model since its application to the airlines passengers data is a popular example of the Box & Jenkins methodology. The outliers detection procedure can be run specifying beforehand the use of this or any other ARIMA model instead of running the model selection procedure. To do so, we set `tsmethod = "arima"` to indicate that an ARIMA model will be used. The ARIMA model is defined in the argument `args.tsmethod`, which is a list that may contain any of the arguments accepted by the familiar function `stats::arima`.

```
resAirP <- tso(y = log(AirPassengers), types = c("AO", "LS", "TC"),
  maxit = 1, discard.method = "bottom-up", tsmethod = "arima",
  args.tsmethod = list(order = c(0, 1, 1), seasonal = list(order = c(0, 1, 1))))
resAirP
```

```
Call:
structure(list(method = NULL), .Names = "method")

Coefficients:
          ma1      sma1     AO29      LS39      LS54      AO62     AO135
      -0.3192   -0.4410   0.0966   -0.0800   -0.0977   -0.0738   -0.1038
s.e.   0.0858    0.0873   0.0206    0.0242    0.0236    0.0204    0.0238

sigma^2 estimated as 0.0008581:  log likelihood = 275.25,  aic = -534.49

Outliers:
  type ind    time  coefhat  tstat
1   AO  29 1951:05  0.09657  4.698
2   LS  39 1952:03 -0.07999 -3.304
3   LS  54 1953:06 -0.09774 -4.134
4   AO  62 1954:02 -0.07380 -3.611
5   AO 135 1960:03 -0.10380 -4.359
```

7.2 Consumer price indices

Table 3 reports the ARIMA model chosen by `TRAMO` both when the presence of potential outliers is considered and when it is omitted. The value of the Jarque-Bera test for normality [7] is also reported. Rejection of the null hypothesis of normality of residuals at the 5% significance level is indicated with an asterisk (the reference critical value is $\chi^2_{2,0.05} = 5.99$).

[Table 3 about here.]

Including outlier regressors may lead to different ARIMA models and may especially affect the order of integration necessary to render stationary residuals. However, in these series we do not observe changes in the order of integration of the regular and seasonal part of the model. As regards the normality of residuals, we observe a significant change when outliers are taken into account. When outliers are omitted the null of normality of residuals is rejected in all cases except in the series '011300'. When the series is checked for the presence of outliers normality of residuals is achieved in 8 out of the 11 series where the null was rejected.

The same exercise is conducted in R. The automatic procedures for the selection of the ARIMA model and the detection of outliers implemented respectively in the packages `forecast` and `tsoutliers` are employed. The function `forecast::auto.arima` is run with the options `allowdrift = FALSE` and `ic = "bic"`, i.e., a drift is not included and the Bayesian information criterion is used; the arguments `maxit = 1` (the process is applied once on the original series) and `discard.method = "bottom-up"` are used in `tso`. Results are reported in Tables 4 and 5. It is observed that including outliers increases the number of cases where normality is achieved, although in fewer cases compare to `TRAMO`. It seems that `TRAMO` tends to favour the choice of a seasonal differencing filter compared to `forecast`. Although not reported, when model selection is not performed and the model chosen by `TRAMO` is passed to the function `tsouliers`, normality is achieved in the same cases as those reported for `TRAMO` in Table 3.

[Table 4 about here.]

[Table 5 about here.]

The outliers detected by each implementation are reported in Table 6.

[Table 6 about here.]

7.3 Industrial production indices

We may be interested in including other external regressors in the model. For example, calendar effects are often relevant when analysing industrial production indices. Regressors other than outliers can be specified by means of the argument `xreg`. These regressors are included in the process of detection of outliers.

Calendar effects (by default trading day and Easter) can be defined by means of the function `calendar.effects`. See the documentation of this function for details.

Below, we show some of the options available in `tsoutliers`. We apply the procedure on the Italian industrial production index, `gipi`. This series is used by [8] to illustrate the relevance of seasonal level shifts. We work with the logarithms of the data, although in the reference paper it is not clear whether they use logarithms or levels. We specify the ARIMA(0,1,1)(0,1,1) model (the model chosen in the reference paper) and set the same critical value used in the reference paper, `cval` = 3.5.

```
data("bde9915")
gipi <- log(bde9915$gipi)
ce <- calendar.effects(gipi)
resGIPI1 <- tso(y = gipi, xreg = ce, cval = 3.5,
  types = c("AO", "LS", "TC", "SLS"), maxit = 1, discard.method = "bottom-up",
  tsmethod = "arima", args.tsmethod = list(order = c(0, 1, 1),
    seasonal = list(order = c(0, 1, 1))))
resGIPI1

Call:
structure(list(method = NULL), .Names = "method")

Coefficients:
          ma1      sma1  trading-day    Easter    SLS28    AO44    SLS49   SLS92  SLS164
      -0.5756   -0.9265       0.0085   -0.0273   -0.062  0.1068  -0.0606  0.0594  0.0704
s.e.   0.0581    0.1132       0.0005    0.0077    0.015  0.0205   0.0117  0.0114  0.0136

sigma^2 estimated as 0.0004802:  log likelihood = 418.52,  aic = -817.05

Outliers:
  type ind    time  coefhat  tstat
1  SLS  28 1983:04 -0.06198 -4.128
2   AO  44 1984:08  0.10679  5.215
3  SLS  49 1985:01 -0.06062 -5.186
4  SLS  92 1988:08  0.05945  5.217
5  SLS 164 1994:08  0.07043  5.187
```

We see that calendar effects are significant and several SLS are found. The outliers detected in the reference paper are AO 1984:08, AO 1987:01, SLS 1994:08 and SLS 1988:08. Two more SLS and one less AO are found with the options chosen above. It must noticed that here we specified the airlines model beforehand while in the reference paper a model selection procedure is used. Hence, despite the airlines model is eventually chosen, different models may have been used in the procedure.

Running the procedure along with model selection the ARIMA(1,0,1)(2,1,1) is chosen and a SLS at 1985 : 01 is found.

```
resGIPI2 <- tso(y = gipi, xreg = ce, types = c("AO", "LS", "TC", "SLS"),
  maxit = 1, discard.method = "bottom-up", tsmethod = "auto.arima",
  args.tsmethod = list(allowdrift = FALSE, ic = "bic"))
resGIPI2
Series: gipi
Regression with ARIMA(0,1,1) errors
```

```
Coefficients:
          ma1  trading-day   Easter   SLS176
      -0.9503       0.0091  -0.0720  -0.6584
s.e.   0.0177       0.0061   0.0868   0.1625

sigma^2 estimated as 0.05275:  log likelihood=10.82
AIC=-11.64   AICc=-11.32   BIC=4.62

Outliers:
  type ind    time coefhat tstat
1  SLS 176 1995:08 -0.6584 -4.05
```

Using `discard.method = "en-masse"`, the model ARIMA(0,1,1)(0,0,2) was chosen and the a SLS at 1988 : 08 is found. Changing the method to discard outliers leads also to a different set of outliers with the ARIMA(0,1,1)(0,1,1) model.

```
resGIPI3 <- tso(y = gipi, xreg = ce, cval = 3.5,
  types = c("AO", "LS", "TC", "SLS"), maxit = 1, discard.method = "en-masse",
  tsmethod = "arima", args.tsmethod = list(order = c(0, 1, 1),
    seasonal = list(order = c(0, 1, 1))))
resGIPI3

Call:
structure(list(method = NULL), .Names = "method")

Coefficients:
          ma1     sma1  trading-day   Easter    AO44    SLS49   AO176
      -0.5972  -0.6995       0.0089  -0.0200  0.0898  -0.0638  0.0963
s.e.   0.0578   0.0673       0.0005   0.0082  0.0216   0.0171  0.0227

sigma^2 estimated as 0.00065:  log likelihood = 398.56,  aic = -781.12

Outliers:
  type ind    time  coefhat  tstat
1   AO  44 1984:08  0.08984  4.164
2  SLS  49 1985:01 -0.06376 -3.730
3   AO 176 1995:08  0.09631  4.237
```

In the next example the number of seasonal differences is fixed to one, `D = 1`. Despite the same model as in `resGIPI2` shown above is chosen and the same method to remove outliers is used, here two additional SLS are found. The reason for that is that although the final model is the same in both cases, different models may probably have been considered during the process, in particular a different order of integration may have been used, leading to a different set of outliers obtained at each step.

```
resGIPI4 <- tso(y = gipi, xreg = ce, types = c("AO", "LS", "TC", "SLS"),
  maxit = 1, discard.method = "bottom-up", tsmethod = "auto.arima",
  args.tsmethod = list(allowdrift = FALSE, D = 1, ic = "bic"))
resGIPI4
Series: gipi
Regression with ARIMA(2,0,2)(2,1,1)[12] errors

Coefficients:
         ar1      ar2      ma1     ma2    sar1     sar2     sma1  trading-day   Easter    AO44
      1.7927  -0.7966  -1.5437  0.6652  0.2107  -0.3535  -0.7553        8e-03  -0.0172  0.0857
s.e.  0.1071   0.1063   0.1042  0.0793  0.0973   0.0820   0.0707        6e-04   0.0066  0.0191
         AO73   SLS92  SLS176
      -0.0647  0.0627  0.0781
s.e.   0.0178  0.0132  0.0179

sigma^2 estimated as 0.0005299:  log likelihood=422.62
AIC=-817.25   AICc=-814.7   BIC=-772.55

Outliers:
  type ind    time  coefhat  tstat
1   AO  44 1984:08  0.08565  4.489
2   AO  73 1987:01 -0.06466 -3.639
3  SLS  92 1988:08  0.06267  4.758
4  SLS 176 1995:08  0.07814  4.356
```

Using different options leads to differences in the results. An inspection of the residuals and a further insight is required from the user in order to choose the model that best fit the data.

8 Conclusion

We have introduced the `tsoutliers` R package. The package implements an automatic procedure for the detection of outliers in time series. The original procedure proposed in the literature relies on the framework of ARIMA models. The package `tsoutliers` adapts the procedure to the context of structural time series models. Currently this extension is an experimental version that requires further development.

The functions available in the package `tsoutliers` allows the user to do a manual run of each step of the procedure, i.e. not in automatic mode. Thus, the package is also useful to track and inspect the behaviour of the procedure and come up with ideas for possible improvements or enhancements.

The application of the `gipi` time series showed the flexibility of the `tso` interface to set and choose different options that may be relevant in practice. Those examples suggested also that pursuing a fully automatic procedure may become too cumbersome. Therefore, alternative manual runs of the automatic procedure are advisable when possible.

References

[1] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control.* San Francisco: Holden-Day, 1970.

[2] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting.* Springer Texts in Statistics. Springer, 2002.

[3] Chung Chen and Lon-Mu Liu. Joint Estimation of Model Parameters and Outlier Effects in Time Series. *Journal of the American Statistical Association*, 88(421):284–297, 1993.

[4] James Durbin and Siem Jan Koopman. *Time Series Analysis by State Space Methods.* Oxford Statistical Science Series. Oxford University Press, 2001.

[5] Andrew C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter.* Cambridge University Press, 1989.

[6] Rob J. Hyndman and Yeasmin Khandakar. Automatic Time Series Forecasting: The `forecast` Package for R. *Journal of Statistical Software*, 27(3):1–22, 2008.

[7] Carlos M. Jarque and Anil K. Bera. Efficient Test for Normality, Homoscedasticity and Serial Independence of Residuals. *Economic Letters*, 6(3):255–259, 1980.

[8] Regina Kaiser and Agustín Maravall. *Measuring Business Cycles in Economic Time Series.* Springer Verlag, 2001.

[9] Javier López-de-Lacalle. *stsm: Structural Time Series Models*, 2016. R package version 1.9.

[10] D. S. G. Pollock. *A Handbook of Time-Series Analysis Signal Processing and Dynamics.* Academic Press, 1999.

[11] Rob J Hyndman with contributions from George Athanasopoulos, Slava Razbash, Drew Schmidt, Zhenyu Zhou, Yousaf Khan, Christoph Bergmeir, and Earo Wang. *forecast: Forecasting Functions for Time Series and Linear Models*, 2014. R package version 5.3.

Table 1: Type I error. Given the model of the data generating process. Version November 2014

		TRAMO				tsoutliers[a]				tsoutliers[b]			
θ_s[c]	cval[d]	IO	AO	LS	TC	AO	LS	TC	SLS	AO	LS	TC	SLS
-0.1	2.80	0.40	0.40	0.36	0.37	0.46	0.50	0.46	0.48	0.38	0.42	0.39	0.37
-0.3		0.41	0.40	0.36	0.35	0.55	0.56	0.55	0.54	0.39	0.38	0.35	0.4
-0.6		0.33	0.39	0.36	0.35	0.56	0.60	0.55	0.55	0.35	0.41	0.35	0.38
-0.9		0.30	0.43	0.39	0.39	0.45	0.49	0.49	0.30	0.34	0.37	0.34	0.22
-0.1	3.17	0.15	0.14	0.13	0.13	0.15	0.19	0.16	0.13	0.14	0.17	0.15	0.14
-0.3		0.14	0.13	0.13	0.12	0.13	0.18	0.13	0.15	0.13	0.15	0.11	0.15
-0.6		0.12	0.14	0.14	0.12	0.15	0.21	0.15	0.14	0.13	0.17	0.12	0.15
-0.9		0.09	0.16	0.14	0.15	0.13	0.18	0.14	0.06	0.12	0.16	0.12	0.07
-0.1	3.50	0.06	0.06	0.05	0.05	0.04	0.06	0.05	0.04	0.04	0.06	0.05	0.04
-0.3		0.05	0.04	0.04	0.04	0.03	0.07	0.03	0.04	0.04	0.06	0.03	0.06
-0.6		0.05	0.04	0.05	0.04	0.04	0.07	0.04	0.04	0.04	0.06	0.04	0.05
-0.9		0.03	0.05	0.07	0.06	0.04	0.06	0.04	0.01	0.04	0.06	0.04	0.02
-0.1	4.00	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.00	0.01	0.02	0.01	0.00
-0.3		0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02
-0.6		0.01	0.01	0.00	0.01	0.01	0.01	0.00	0.00	0.01	0.01	0.01	0.00
-0.9		0.00	0.02	0.01	0.00	0.01	0.02	0.01	0.00	0.01	0.02	0.01	0.00

Cells report the percentage of cases where a type of outlier was found. Based on $1,000$ series of length 120 generated from the Airlines model, ARIMA(0,1,1)(0,1,1), with no outliers. Model identification is not performed; the Airlines model is used.

[a]Remove method in the second stage of the procedure is the "en-masse" approach.

[b]Remove method in the second stage of the procedure is the "bottom-up" approach.

[c]Seasonal moving average coefficient. The non-seasonal moving average coefficient is fixed to -0.6.

[d]Critical value to determine the presence of outliers.

Table 2: Type I error. Given the model of the data generating process. Version May 2017

		TRAMO				tsoutliers[a]				tsoutliers[b]			
θ_s[c]	cval[d]	IO	AO	LS	TC	AO	LS	TC	SLS	AO	LS	TC	SLS
−0.1	2.80	0.40	0.40	0.36	0.37								
−0.3		0.41	0.40	0.36	0.35								
−0.6		0.33	0.39	0.36	0.35								
−0.9		0.30	0.43	0.39	0.39								
−0.1	3.17	0.15	0.14	0.13	0.13	0.24	0.28	0.27	0.24	0.23	0.23	0.23	0.23
−0.3		0.14	0.13	0.13	0.12	0.24	0.27	0.22	0.27	0.23	0.22	0.19	0.25
−0.6		0.12	0.14	0.14	0.12	0.25	0.30	0.25	0.23	0.19	0.25	0.21	0.22
−0.9		0.09	0.16	0.14	0.15	0.22	0.25	0.23	0.11	0.20	0.22	0.19	0.12
−0.1	3.50	0.06	0.06	0.05	0.05	0.08	0.10	0.09	0.07	0.09	0.10	0.09	0.08
−0.3		0.05	0.04	0.04	0.04	0.07	0.12	0.07	0.09	0.08	0.10	0.07	0.10
−0.6		0.05	0.04	0.05	0.04	0.08	0.11	0.08	0.08	0.08	0.11	0.08	0.09
−0.9		0.03	0.05	0.07	0.06	0.08	0.10	0.08	0.03	0.07	0.09	0.07	0.03
−0.1	4.00	0.01	0.01	0.01	0.00	0.02	0.03	0.02	0.01	0.02	0.02	0.02	0.01
−0.3		0.00	0.01	0.01	0.01	0.01	0.02	0.01	0.02	0.01	0.02	0.01	0.02
−0.6		0.01	0.01	0.00	0.01	0.02	0.02	0.01	0.01	0.02	0.02	0.01	0.01
−0.9		0.00	0.02	0.01	0.00	0.02	0.03	0.01	0.00	0.02	0.03	0.01	0.01

Cells report the percentage of cases where a type of outlier was found. Based on 1,000 series of length 120 generated from the Airlines model, ARIMA(0,1,1)(0,1,1), with no outliers. Model identification is not performed; the Airlines model is used.

[a]Remove method in the second stage of the procedure is the "en-masse" approach.

[b]Remove method in the second stage of the procedure is the "bottom-up" approach.

[c]Seasonal moving average coefficient. The non-seasonal moving average coefficient is fixed to −0.6.

[d]Critical value to determine the presence of outliers.

Table 3: Test for normality of residuals in ARIMA models chosen by software TRAMO

	Omitting outliers		Including outliers	
Series	Model	JB	Model	JB
foodpr	$(2,1,0)(1,0,1)$	637.60 ∗	$(1,1,1)(1,0,1)$	33.99 ∗
igxe00	$(0,1,2)(0,1,0)$	354.30 ∗	$(3,1,0)(0,1,1)$	173.60 ∗
011600	$(0,1,0)(0,1,1)$	8.16 ∗	$(3,1,1)(0,1,1)$	0.39
011000	$(1,1,1)(0,1,1)$	10.65 ∗	$(1,1,1)(0,1,1)$	3.18
010000	$(1,1,0)(1,0,0)$	8.54 ∗	$(2,1,0)(0,1,1)$	3.35
011200	$(1,1,0)(0,1,1)$	28.12 ∗	$(1,1,0)(0,1,1)$	8.96 ∗
nrgy00	$(0,1,1)(0,0,0)$	19.09 ∗	$(0,1,1)(0,0,0)$	19.23 ∗
011700	$(0,1,1)(0,1,1)$	15.23 ∗	$(0,1,0)(0,1,1)$	0.67
foodun	$(0,1,1)(1,0,0)$	8.29 ∗	$(0,1,1)(0,1,1)$	0.21
011300	$(0,1,1)(0,1,1)$	0.05	$(0,1,0)(0,1,1)$	2.58
000000	$(0,1,1)(0,1,1)$	10.54 ∗	$(0,1,1)(0,1,1)$	1.76
serv00	$(1,1,0)(0,1,1)$	290.10 ∗	$(3,1,1)(0,1,1)$	2.66

Table 4: Test for normality of residuals in ARIMA models chosen in R. Model selection is done according to BIC. Version December 2016; `forecast` 7.3 `tsoutliers` 0.6-5

	Omitting outliers[a]		Including outliers[b]	
Series	Model	JB	Model	JB
000000	$(0,1,1)(2,1,0)$	10.27 $*$	$(0,1,0)(1,0,0)$	3.44
010000	$(1,1,0)(2,0,0)$ + drift	9.49 $*$	$(2,1,0)(2,0,0)$	0.30
011000	$(2,1,0)(2,0,0)$	32.64 $*$	$(2,1,0)(2,0,0)$	2.05
011200	$(1,1,0)(1,0,0)$ + drift	35.69 $*$	$(1,1,0)(1,0,0)$ + drift	3.38
011300	$(0,1,0)(2,0,0)$ + drift	1.41	$(0,1,0)(1,0,0)$	1.01
011600	$(1,1,0)(1,0,0)$	31.22 $*$	$(2,1,2)(1,0,0)$	21.21 $*$
011700	$(1,1,0)(2,0,0)$	74.84 $*$	[c]	
foodpr	$(2,1,0)(0,0,1)$ + drift	558.47 $*$	$(1,1,1)(1,0,0)$ + drift	129.42 $*$
foodun	$(1,1,0)(2,0,0)$	12.72 $*$	$(1,1,0)(2,0,0)$	0.04
igxe00	$(0,1,0)(0,1,2)$	802.06 $*$	$(0,1,0)(2,1,0)$	65.83 $*$
nrgy00	$(1,1,0)(1,0,0)$ + drift	20.39 $*$	$(1,1,0)(1,0,0)$ + drift	20.39 $*$
serv00	$(1,1,2)(2,1,0)$	359.06 $*$	$(2,0,2)(2,1,0)$ + drift	142.78 $*$

[a]Model selection is carried out by means of the function `forecast::auto.arima` with the Bayesian information criterion (BIC).

[b]Outliers are detected by means of the function `tsoutliers::tso`; model selection is carried out as in the column 'omitting outliers'.

[c]The default options failed when outliers were considered.

Table 5: Test for normality of residuals in ARIMA models chosen in R. Model selection is done according to AICc. Version December 2016; `forecast` 7.3 `tsoutliers` 0.6-5

	Omitting outliers[a]		Including outliers[b]	
Series	Model	JB	Model	JB
000000	$(2,1,2)(2,1,0)$	11.78 $*$	$(1,1,0)(1,0,0)$ + drift	9.66 $*$
010000	$(2,1,0)(2,0,0)$ + drift	11.40 $*$	$(2,1,0)(2,0,0)$	0.30
011000	$(2,1,0)(2,0,0)$ + drift	32.73 $*$	$(2,1,0)(2,0,0)$ + drift	2.35
011200	$(1,1,0)(1,0,0)$ + drift	35.69 $*$	$(1,1,0)(2,0,0)$ + drift	2.00
011300	$(1,1,0)(2,0,0)$ + drift	1.63	$(4,1,0)(1,0,0)$ + drift	0.92
011600	$(1,1,0)(1,0,0)$	31.22 $*$	$(3,1,2)(1,0,0)$	9.13 $*$
011700	$(1,1,2)(2,0,0)$	107.04 $*$	$(1,1,1)(2,0,0)$	1.49
foodpr	$(2,1,1)(0,0,2)$ + drift	615.43 $*$	$(2,1,0)(2,0,0)$ + drift	77.90 $*$
foodun	$(1,1,0)(2,0,0)$	12.72 $*$	$(2,1,0)(2,0,0)$	0.07
igxe00	$(0,1,0)(0,1,2)$	802.06 $*$	$(0,1,0)(0,1,2)$	242.30 $*$
nrgy00	$(1,1,0)(2,0,0)$ + drift	20.47 $*$	$(1,1,0)(2,0,0)$ + drift	20.57 $*$
serv00	$(1,1,2)(2,1,0)$	359.06 $*$	$(2,0,2)(2,1,0)$ + drift	142.78 $*$

[a]Model selection is carried out by means of the function `forecast::auto.arima` with the Aikaike information criterion modified for small samples (AICc).

[b]Outliers are detected by means of the function `tsoutliers::tso`; model selection is carried out as in the column 'omitting outliers'.

Table 6: Outliers detected in the HICP data set

	`TRAMO`[a]	`tsoutliers`[b]	`tsoutliers`[c]	`tsoutliers`[d]
Series	`Outliers`	`Outliers`		
000000	TC19 LS215 LS231 AO235	TC19 TC133 LS215 LS231	TC19 LS215	TC19 LS215 LS231 AO235
010000	AO23 TC85 TC145	AO23 TC145	AO23 TC145	AO23 TC145
011000	TC85	TC25 TC85	TC25 TC85	TC85
011200	AO28	LS29	LS29	AO28
011300	AO120 TC157 TC162 LS171	AO120 TC157 TC162 LS171	TC157 TC162 LS171	AO120 TC157 TC162 LS171
011600	AO79 LS85 AO138 TC163 AO210 LS225	TC17 AO79 AO150 AO210	TC17 AO79 AO150 AO210 LS225	AO79 TC163 AO210 LS225
011700	AO25 TC85 TC105		AO25 LS77 TC85 TC105	AO25 TC85 TC105
foodpr	AO145 LS157 LS171 LS180 TC189 TC202 LS214	TC2 TC145 AO156 TC168 LS171 LS180 TC189	TC2 LS37 TC145 AO156 AO167 LS171 LS180 TC202 LS214	AO145 LS157 LS171 TC180 TC189 TC202 LS214
foodun	TC145	AO23 TC145	AO23 TC145	TC145
igxe00	AO134 TC145 TC259	AO134 AO254 TC259 LS261	AO134 LS261	AO134 LS145 TC259
nrgy00	TC129	None	TC129	TC129
serv00	TC7 AO16 TC132 LS144 AO219 AO280	TC7 AO280	TC7 AO280	TC19 TC132 AO219 AO280

[a]Continuation of results reported in the fourth and fifth columns in Table 3.
[b]Continuation of results reported in the fourth and fifth columns in Table 4.
[c]Continuation of results reported in the fourth and fifth columns in Table 5.
[d]Outliers detected by `tsoutliers:tso` given the ARIMA models selected by TRAMO.