Journal of Statistical Software

August 2022, Volume 103, Issue 9. *doi: 10.18637/jss.v103.i09*

Automatic Identification and Forecasting of Structural Unobserved Components Models with UComp

Diego J. Pedregal
Universidad de Castilla-La Mancha

Abstract

UComp is a powerful library for building unobserved components models, useful for forecasting and other important operations, such us de-trending, cycle analysis, seasonal adjustment, signal extraction, etc. One of the most outstanding features that makes **UComp** unique among its class of related software implementations is that models may be built automatically by identification algorithms (three versions are available). These algorithms select the best model among many possible combinations. Another relevant feature is that it is coded in C++, opening the door to link it to different popular and widely used environments, like R, MATLAB, Octave, Python, etc. The implemented models for the components are more general than the usual ones in the field of unobserved components modeling, including different types of trend, cycle, seasonal and irregular components, input variables and outlier detection. The automatic character of the algorithms required the development of many complementary algorithms to control performance and make it applicable to as many different time series as possible. The library is open source and available in different formats in public repositories. The performance of the library is illustrated working on real data in several varied examples.

Keywords: unobserved components models, state space models, Kalman filter, fixed point smoother, maximum likelihood, R, MATLAB, Octave.

1. Introduction

UComp is a library that implements comprehensive procedures for the automatic identification, estimation and forecasting of univariate unobserved components models (UC). All the implemented models belong to the class of structural UC proposed in a seminal work of Harvey (1989) and extended in many ways by a long and rich literature stream (see, e.g., De

Jong 1991; Young, Pedregal, and Tych 1999; Durbin and Koopman 2012; Proietti and Luati 2013; Pelagatti 2015, etc.).

There are many different implementations of structural UC methods nowadays, some of them focused exclusively on this class of models, but many others that include UC as part of a more general statistical or state space (SS) software. Among those dedicated exclusively to UC and probably one of the most comprehensive is **STAMP**, that is a standalone package (Koopman, Harvey, Doornik, and Shephard 1999). Other packages very popular are **rucm**, **KFAS**, **stsm**, **bsts** and **autostsm** in R (Chowdhury 2015; Helske 2017; López-de-Lacalle 2016; Scott 2022; Hubbard 2022; R Core Team 2022). A big advantage of these R packages is that they are open-source, while **STAMP** is commercial.

The list of general statistical or SS software that includes, among others, structural UC methods as part of its tools is immense. Only an illustrative non-exhaustive list may be provided here:

- General ones: **SsfPack** (Koopman, Shephard, and Doornik 2008), SAS (Selukar 2011), Stata (Drukker and Gates 2011), **Eviews** (Bossche 2011), RATS (Doan 2011) and gretl (Lucchetti 2011). See also the special issue in volume 41, 2011 of the Journal of Statistical Software (Commandeur, Koopman, and Ooms 2011).
- MATLAB toolboxes: **SSM** official toolbox (The MathWorks Inc. 2021), **SSpace** (Villegas and Pedregal 2018), **ECOTOOL** (Pedregal 2019), **CAPTAIN** (Taylor, Pedregal, Young, and Tych 2007), **SSM** (Peng and Aston 2011), **E4** (Casals, Garcia-Hiernaux, Jerez, Sotoca, and Trindade 2016), **SSMMATLAB** (Gómez 2015).
- R packages: Apart from the ones cited above, the base package **stats** and the contributed packages **astsa** (Stoffer 2022) and **statespacer** (Beijers 2022).
- Other environments with increasing relevance: **statsmodels** in Python (Seabold and Perktold 2010; Van Rossum *et al.* 2011), **StateSpaceModels** in Julia (Saavedra, Bodin, and Souto 2019; Bezanson, Edelman, Karpinski, and Shah 2017).

From a broader perspective, there are many other computer programs similar to **UComp** in the sense of providing automatic time series models, albeit taking advantage of different methods apart from UC, mainly ARIMA and exponential smoothing. Some of the most popular are those procedures implemented in **Forecast Pro** (Goodrich 2000), **Autobox** (Reilly 2000), **TRAMO** (Gómez and Maravall 2001), package **forecast** in R (Hyndman and Khandakar 2008), **SPSS** (IBM Corporation 2019), **STATA** (StataCorp 2019), **SAS** (SAS Institute Inc. 2020), and **EViews** (IHS Markit 2021). See also the CRAN Task View on Time Series Analysis (Hyndman and Killick 2022) for a long list of time series packages in R.

It is literally impossible to make a full evaluation of all the above software packages compared to **UComp** within a reasonable space. **UComp** is more restricted than some of the previous libraries/programs in some respects, as it deals with univariate linear Gaussian models only. But it offers a combination of advantages altogether in a single library that are not simultaneously available in any of them. The following list shows some important general properties of this library:

1. Models may be identified automatically, without human intervention, by information criteria. This means that the user does not need to impose any prior structure on the

model, because it may be decided by **UComp**. This algorithm is applied with complete flexibility, in the sense that the user may decide whether to let it pick all the components, only one or part of them. There are several options for the search, one that runs the whole population of models and others which run a subset of them through a stepwise procedure, see Section 3.

2. **UComp** is coded in C++. This ensures optimal execution speed and the ability to link it to many popular environments by writing the appropriate wrapper functions. At the time of writing this paper, there are versions written in R (Pedregal 2022), with package **UComp** being available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=UComp`, and MATLAB and Octave (Eaton, Bateman, Hauberg, and Wehbring 2022), available at `https://github.com/djpedregal/UComp`. Extension to other environments may be done by updating the wrapper function `UCompC.cpp` to the new environment (any contribution is welcome!).

3. The family of component models is wider than usual within the class of linear Gaussian UC models. Damped trends, seasonal components that do not include all the harmonics and give different power to each of them, or ARMA irregular components are some possibilities not common in many computer programs. Cycles are included with the option of estimating their period by maximum likelihood so that, combined with the automatic identification, this means that it is possible to test the existence of cycles automatically (see Section 5). Note that combinations of all possible component types expand the variety of time series to which the UC class may be applied.

4. **UComp** may include exogenous inputs as regression terms in the specification. Automatic outlier detection is also possible in conjunction with the automatic identification algorithm. **UComp** implements the approach given in Harvey and Koopman (1992), whereby perturbations above a certain threshold are considered as outliers and included in the model as dummy variables. Some of them are later discarded if the final t-test statistic is below a certain level set by the user.

Note that **UComp** focuses on univariate Gaussian UC models, allowing for specific design choices such as some of those listed below, but which would not necessarily be optimal for time series with properties not accounted for in the package. Therefore, **UComp** has been implemented including updated statistical developments in the field, for example:

- A disturbance smoother is implemented according to Durbin and Koopman (2012), which opens the door to further applications: (i) as an additional diagnostic check based on the estimated perturbations, also key for outlier detection; and (ii) useful to calculate exact gradients of the likelihood functions with a significant increase in model estimation speed when the unknown parameters involved are perturbation variances only.

- Exact initialization of recursive algorithms (Kalman filter, fixed interval smoother and disturbance smoother) is implemented both as the exact diffuse initialization due to Durbin and Koopman (2012) and as the augmented Kalman filter (De Jong 1991). The former is important both for the evaluation of the exact maximum likelihood and analytical gradients of the log-likelihood function and the latter is relevant for models with inputs. Note that analytical gradients are computed using the disturbance smoother

and can only be used in models where all parameters are variances (Koopman and Sehphard 1992).

- Profile likelihood is used always. As Harvey (1989) states on pp. 183, *there is no doubt that concentrating a parameter out of the likelihood is not only computationally efficient but is also likely to give more reliable results*. This is important for making estimation scale independent, and also to simplify the estimation by concentrating out of the likelihood one of the variance parameters. Additionally, for those models including inputs, the linear parameters affecting them are concentrated out with the help of the augmented Kalman filter.

  The use of profile likelihoods in this context implies estimating ratios of variances directly, instead of variances themselves. When the variance in the denominator is close to zero, all sorts of numerical troubles appear and such a variance ought to be replaced by some other value well above zero. Such problems are avoided in **UComp** by swapping the variance in the denominator by a different one while running the estimation algorithm, see below.

- Parameter search is done by a quasi-Newton algorithm included in the library. This allows on-the-fly operations, such as switching between concentrated variances whenever there is evidence that the one chosen is close to zero, or restricting parameters when they reach their boundaries (e.g., zero variances, or damping factors reaching values of 0 or 1). The optimization method is a Broyden-Fletcher-Goldfarb-Shanno (BFGS) quasi-Newton algorithm with a backtracking line search using Armijo conditions. The BFGS part ensures fast updates of the Hessian and the backtracking line search ensures the algorithm moves in the correct direction with a step-length that depends on the function properties, see Chapter 6 of Nocedal and Wright (2006) or Fletcher (2000). Covariances of parameters are estimated by the numerical Hessian evaluated at the optimum computed after estimation.

- Most parameters are constrained between certain values, e.g., variances must be non-negative, damping factors should be between 0 and 1, cycle periods are constrained to avoid confusion with seasonal periods, etc. Such constraints are actually taken into account by transforming the parameter space following Koopman *et al.* (1999) and Koopman *et al.* (2008). These references also include careful selection of initial conditions for each sort of parameter to start the non-linear search. Special mention deserves the estimation of constrained ARMA irregular models to be stationary and invertible, that are implemented according to Monahan (1984).

- The identification algorithm requires many complementary algorithms to make it as fast as possible, fully automatic and reliable. Some examples are harmonic regression, augmented Dickey-Fuller unit root tests, automatic selection of AR and ARMA models by linear procedures, constrained estimation of ARMA models to preserve stationarity and invertibility, etc.

On the operational side, **UComp** is user friendly in the sense that a complete UC analysis can be carried out with only a few functions. In fact, all the calculations can be done with a simple call to the `UC` function in which the time series is the only input. Nevertheless, the analysis may be much more sophisticated, as later examples show.

The rest of the paper is organized as follows. Section 2 presents the UC class of models available in **UComp**. Section 3 shows how the automatic identification works step by step. Section 4 provides a quick guide to the most important features of the library. Section 5 discusses several worked examples. Finally, Section 6 draws the main conclusions.

2. Unobserved components models

UC models decompose a time series into meaningful components. The general decomposition used in this paper is shown in Equation 1, where T_t, C_t, S_t, and I_t stand for a trend, cycle, seasonal and irregular components, respectively. The model allows also for linear relationships with k exogenous variables $x_{i,t}$ affected by a set of parameters $\beta_i, (i = 1, \ldots, k)$. Simplified versions of this model, without cycles and/or inputs, are common in many studies and software programs.

$$z_t = T_t + C_t + S_t + I_t + \sum_{i=1}^{k} \beta_i x_{i,t} \tag{1}$$

The structural approach of UC consists in setting the above model in a SS framework in which Equation 1 is actually the observation equation and the state equations are the block concatenation of the models representing the dynamic behavior assumed for each of the components (see below and Harrison and Stevens 1976; Harvey 1989; Young *et al.* 1999; Durbin and Koopman 2012, among many others).

Since a picture is worth one thousand words, two examples illustrate how any UC model may be written in SS form. The first example is shown in Equation 2 and is composed of a trend, seasonal and irregular components. The equations in matrix form on top are the so called 'state equations' which establish the dynamic mechanism of the unobserved state vector (α_t), by relating it in two consecutive time stamps. The equation at the bottom is the 'observation equation' and is only a selection of state elements to replicate the model in Equation 1.

$$\begin{aligned}
\alpha_{t+1} = \begin{bmatrix} T_{t+1} \\ \hline S_{1,t+1} \\ S^*_{1,t+1} \\ S_{2,t+1} \\ S^*_{2,t+1} \end{bmatrix} &= \left[\begin{array}{c|cccc} 1 & 0 & 0 & 0 & 0 \\ \hline 0 & \cos\omega_1 & \sin\omega_1 & 0 & 0 \\ 0 & -\sin\omega_1 & \cos\omega_1 & 0 & 0 \\ 0 & 0 & 0 & \cos\omega_2 & \sin\omega_2 \\ 0 & 0 & 0 & -\sin\omega_2 & \cos\omega_2 \end{array}\right] \begin{bmatrix} T_t \\ \hline S_{1,t} \\ S^*_{1,t} \\ S_{2,t} \\ S^*_{2,t} \end{bmatrix} + \begin{bmatrix} \eta_{T,t} \\ \hline \eta_{1,t} \\ \eta^*_{1,t} \\ \eta_{2,t} \\ \eta^*_{2,t} \end{bmatrix} \\
z_t &= \left[\begin{array}{c|cccc} 1 & 1 & 0 & 1 & 0 \end{array}\right] \alpha_t + I_t
\end{aligned} \tag{2}$$

The trend is the first element of the state vector and obeys to a random walk model (RW), i.e., the element T_t of the state vector α_t depends on the previous sample and is affected by a Gaussian white noise $(\eta_{T,t})$ with constant variance σ^2_T. The rest of the elements in the state vector define the seasonal component, that is actually the sum of two terms $(S_{1,t} + S_{2,t}$, see how the observation equation collects these two elements from the state vector in one single component), and their definition involves two frequencies (ω_1 and ω_2) that are assumed to play the role of the frequency associated to the fundamental seasonal periodicity and one of its harmonics. The seasonal component involves four Gaussian noises $(\eta_{1,t}, \eta^*_{1,t}, \eta_{2,t}, \eta^*_{2,t})$ that in usual formulations are mutually independent with common variance σ^2_S. The final element is the irregular component (I_t) that is simply white noise with constant variance σ^2_I.

The seasonal model implemented in **UComp** is of a trigonometric form as in **STAMP**, as Equation 2 shows. Another possibility is the dummy seasonal component with time varying

coefficients. This latter approach is useful when different variances are perceived among different seasonal effects. On the contrary, it is somewhat more complex because it needs a time varying SS system (see Proietti 1998, for a comprehensive review of models, advantages and disadvantages of all these possibilities).

In general, a full seasonal component in the trigonometric model is built as the sum of individual sinusoidal terms as in Equation 2 for the fundamental period $P_1 = 2\pi/w_1$ (the number of observations per year) and all its harmonics. The number of harmonics in discrete data in general is $[P_1/2] = P_1/2$ for even P_1 numbers, and $[P_1/2] = (P_1 - 1)/2$ for uneven P_1 numbers, with periods $P_j = P_1/j$, $(j = 1, 2, \ldots, [P_1/2])$. For example, a seasonal component for a monthly time series would exhibit 6 harmonics with periods 12, 6, 4, 3, 2.4 and 2 months per year, while there are only 2 harmonics for quarterly data with periods 4 and 2 quarters. Note that each harmonic is defined by two states in Equation 2, but it reduces to a single equation in the case of the last harmonic with period $P_j = 2$ samples, because in that case $\sin(2\pi/P_j) = \sin(\pi) = 0$.

This model only depends on three unknown variances, namely σ_T^2, σ_S^2 and σ_I^2. It can get a bit more complicated if the seasonal component is allowed to have different variances for each harmonic, in which case the seasonal variance splits into two. In such a case, one extra parameter is added, and the seasonal component gains in flexibility.

A second example is shown in Equation 3 and includes a damped trend (DT), a cycle component with frequency ω and an AR(2) model for the irregular component. To have a DT the parameter γ should be constrained between zero and one. Other common trend models may be treated as particular cases of a DT, such as the so called local linear trend (LLT, with $\gamma = 1$), the integrated random walk (IRW, $\gamma = 1$ and $\sigma^2_{\eta_T} = 0$), the RW with drift ($\gamma = 1$, $\sigma^2_{\eta^*_T} = 0$, $T_1^* \neq 0$), and a deterministic linear trend ($\gamma = 1$, $\sigma^2_{\eta_T} = 0$, $\sigma^2_{\eta^*_T} = 0$, $T_1^* \neq 0$). The cycle model is exactly like the seasonal one but with a transition matrix that is multiplied by a damping factor $0 < \rho < 1$.

The unknown parameters in this example are γ, ϕ_1, ϕ_2, $\sigma^2_{\eta_T}$, $\sigma^2_{\eta^*_T}$, σ^2_{η}, $\sigma^2_{\eta_I}$, ρ, and ω if the cycle frequency is unknown and ought to be estimated.

$$\alpha_{t+1} = \left[\begin{array}{c} T_{t+1} \\ T^*_{t+1} \\ \hline C_{t+1} \\ C^*_{t+1} \\ \hline I_{t+1} \\ I^*_{t+1} \end{array}\right] = \left[\begin{array}{cc|cc|cc} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & \gamma & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & \rho\cos\omega & \rho\sin\omega & 0 & 0 \\ 0 & 0 & -\rho\sin\omega & \rho\cos\omega & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & \phi_1 & 1 \\ 0 & 0 & 0 & 0 & \phi_2 & 0 \end{array}\right] \left[\begin{array}{c} T_t \\ T^*_t \\ \hline C_t \\ C^*_t \\ \hline I_t \\ I^*_t \end{array}\right] + \left[\begin{array}{c} \eta_{T,t} \\ \eta^*_{T,t} \\ \hline \eta_t \\ \eta^*_t \\ \hline \eta_{I,t} \\ 0 \end{array}\right] \tag{3}$$

$$z_t = \left[\begin{array}{cc|cc|cc} 1 & 0 & 1 & 0 & 1 & 0 \end{array}\right] \alpha_t$$

Note that, although the model is relatively simple (just three components), it involves many more parameters than the first example. In addition, the parameters have to meet a number of constraints, namely, all variances and ω ought to be estimated as positive, ρ and γ must be estimated between zero and one and the roots of the AR(2) polynomial should lie outside the unit circle to ensure a stationary irregular component.

Once the model is setup, well-known recursive algorithms provide the estimation of the states, perturbations and their covariance structures, namely the Kalman filter, fixed interval and disturbance smoothers. However, these algorithms work only on fully specified models, i.e.,

models in which there are no unknown elements, which is generally not true, as shown in the examples above.

Therefore the parameters must be estimated in some way. Maximum likelihood is the usual method applied to these systems. There are many issues related to the identification and estimation of UC models that are beyond the scope of this paper and are addressed in many excellent references (see, for example, Harvey 1989; Young *et al.* 1999; De Jong 1991; Durbin and Koopman 2012; Proietti and Luati 2013; Pelagatti 2015; Casals *et al.* 2016; Villegas and Pedregal 2018). Only those issues specifically relevant to this paper will be mentioned below.

3. An automatic identification algorithm for UC models

As far as the author is concerned, **UComp** is the first software implementing an exhaustive automatic algorithm for the identification of structural UC models. The algorithm relies on information criteria and involves a wide range of auxiliary algorithms and technical details that are briefly presented in what follows. It may be considered as a set of automatic forecasting and modeling tools complementary to the **forecast** package and other packages quoted in Section 1. The algorithm is designed to find the best model possible within the class of univariate UC models according to information criteria, using a wide range of typical component models found in the literature. Then, if a different UC model not considered in **UComp** is the 'true' representation of certain data, **UComp** would provide the best approximation.

One way to reduce the number of models to estimate is to divide the algorithm in modules that run sequentially one after another:

- **Module 1:** *Pre-testing.*

  A seasonality test is run based on a harmonic regression with a third order polynomial trend. This test determines which harmonics are involved in the seasonal component and the rest of the algorithm proceeds according to this information. One harmonic is considered absent when the t-test statistics of its coefficients are below 1.645 in absolute value, equivalent to a 10% significance test assuming Gaussian errors. Such harmonic is present if the t-test statistic is above 3 in absolute value and inconclusive if the t-test statistic is within 1.645 and 3.

- **Module 2:** *Model selection.*

  A battery of models combining different configurations of components are estimated and the best is chosen by minimizing well-known information criteria, such as the standard or corrected Akaike (AIC and AICc), or the Bayesian (BIC), see Harvey (1989).

  A critical issue in this module is the number and type of models for each of the components to be combined, so that they can represent a sufficiently wide range of time series, but avoiding an unnecessary increase in computational burden. The models finally considered are the following:

  - Trends: None, random walk, local linear trend or damped trend.
  - Cycles: None, just one or several cycles. Periods of each cycle may be fixed or estimated independently of each other.
  - Seasonal components: None, with one single variance for all the harmonics or with different variances for each one.

  - Irregular component: None or white noise.

- **Module 3:** *ARMA model selection.*

  If the innovations of the model identified in Module 2 are not white noise, a stationary and invertible ARMA(p, q) model of orders smaller than the seasonal period is identified automatically using an adaptation of the procedure by Hyndman and Khandakar (2008).

- **Module 4:** *Joint estimation.*

  If either orders p or q are greater than zero in Module 3 a joint estimation is run including the unobserved components identified in previous modules including the ARMA(p, q) model for the irregular component. The joint model is only the final model chosen if it shows a better information criterion than the one in Module 2.

UComp allows for three different implementations of Module 2 above: Firstly, a full implementation that consists barely of running all the possible combinations of components in Module 2 (23 models in total and 47 if cycles are included). Secondly, a stepwise version described below, which reduces the number of models to run. Finally, stepwise with unit root testing added in Module 1.

The stepwise versions use the harmonic regression of Module 1 in an additional way, consisting of deciding whether or not a seasonal component is present in the data. First, if one of the t-test statistics is greater than 3 the seasonal component is present and all models without seasonal components are discarded from the search. Second, if all the t-test statistics are lower than 1.645, then the seasonal component is not relevant and only models without a seasonal components are considered in further steps. Finally, if any of the t-test statistics are between 1.645 and 3 and none is higher than 3, then the test is inconclusive and all seasonal possibilities are taken into account in the following steps.

Then, the stepwise procedures reduce the number of models to estimate according to the following steps:

1. The best of all models with none and RW trends is selected, according to the preferred information criterion. If the best model is one without a trend, then go to Step 2, otherwise proceed to Step 3. This step is similar to testing for a unit root tests.

2. Models with DT trends are estimated and the best model overall is selected. End.

3. Run all possible models with a LLT trend. In fact this step is testing empirically for a second unit root.

4. All the models for the components in Steps 1 and 3 are combined with a DT trend. The best model overall is selected. End.

The stepwise algorithm with unit root tests replaces Step 1 above by an augmented Dickey-Fuller unit root tests with the number of output lags identified by BIC. Test statistic values below -5, between -5 and -2, and above -2 define the areas of stationarity, inconclusive and non-stationarity regions of the test, respectively. The algorithm then proceeds according to the combined information offered by the unit root tests and the seasonality test in Module 1, except in the case of inconclusive unit root tests, where the algorithm switches to the standard stepwise procedure described above. The statistical tests are done with such conservative

values for the test statistics that the combination of testing and information criterion in **UComp** is free from usual criticisms (see, e.g., Flom 2018), and actually stepwise and full procedures provide the same results in all the experiments carried out so far.

The previous algorithms can be run either with outlier or without outlier detection, meaning that for each model in the process an outlier detection procedure inspired by Harvey and Koopman (1992) may be activated. At the moment, **UComp** includes three types of outliers, namely additive outliers (AO), level shifts (LS) and slope changes (SC). The outlier detection is based on the auxiliary residuals or the standardized disturbances obtain from the disturbance smoother. LS outliers are detected on abnormally big level auxiliary residuals, SC on slope residuals and AO on observation residuals.

This algorithm consists of the following steps:

- Initial estimation: a model without outliers is estimated.
- Forward addition step: all observations in auxiliary residuals above certain critical values (2.5 for level, 3 for slope and 2.3 for observation disturbances, respectively) are considered as potential outliers. Note that these numbers are bigger than 2 to take into account the fact that the auxiliary residuals are serially correlated on a correctly specified UC model (see Harvey and Koopman 1992). Taking these figures as lower critical limits avoids the proliferation of countless outliers in a model
- Intermediate estimation: a model including all the outliers is estimated.
- Backward deletion step: all outliers with t-test statistics smaller than the value demanded by the user are dropped off and a new model with less outliers is estimated. This procedure is repeated until all outliers in the final model have t test statistics above the value required.
- Selection of final model: if the model in the previous step has worse information criteria metrics than the initial one, all outliers are rejected.

4. UComp overview

UComp comprises functions that allow a comprehensive analysis of structural UC time series models, either manually or automatically identified. Table 1 lists all the functions available to handle the modeling of UC models. The function names begin with 'UC' followed by the particular action each function performs.

All these functions either create or work on **UComp** objects, which are either S3 classes in R or MATLAB/Octave structures, depending on the environment. The upper section of Table 1 shows the functions that create such objects and allow to configure all the options that control the operation of the package. The lower section shows the rest of the functions that work directly on the **UComp** objects that already exist and have a common simple syntax.

The syntax in R and MATLAB/Octave is kept as similar as possible, so that the transition from one to the other is trivial. In addition, to provide compatibility among R packages, several popular methods available in a similar way in well-known R packages are also implemented for **UComp** objects, see Table 2. All of the listed functions return objects of appropriate classes.

Name	Description
`UC`	Overall function that runs all the rest.
`UCsetup`	Creates an **UComp** object and sets all input options controlling how the rest of the functions work.
`UCmodel`	Runs `UCsetup` and `UCestim`.
`UCestim`	Identifies UC model, estimates it by maximum likelihood and computes forecasts.
`UCvalidate`	Output table with diagnostic checks of UC model estimated.
`UCfilter`	Optimal Kalman filtering of UC models.
`UCsmooth`	Optimal fixed interval smoother.
`UCdisturb`	Optimal disturbance smoother.
`UCcomponents`	Components estimation.
`UChp`	Hodrick-Prescott filtering.
`getp0`	Get initial conditions for parameter estimation.

Table 1: Main functions of **UComp**.

Name	Description
`print`	Prints a summary of models estimated (same as `UCvalidate`).
`summary`	Same as `print`.
`plot`	Plots estimated components.
`fitted`	Extracts fitted values from **UComp** objects.
`residuals`	Extracts model residuals from **UComp** objects.
`logLik`	Log-likelihood of model at optimum.
`coef`	Extracts model coefficients from **UComp** objects.
`predict`	Extract predictions of the model.
`tsdiag`	Plots some time series diagnostics on the residuals.
`AIC`	Extract AIC from model.
`BIC`	Extract BIC from model.

Table 2: Methods available in **UComp** common to other R packages.

The work flow of **UComp** is straightforward. The model is created with `UCsetup` and the unknown parameters are estimated by `UCestim` (alternatively, both actions may be carried out with one single call to `UCmodel`, while `UC` runs all functions one after another). The suitability of the identified model from a statistical point of view is checked with `UCvalidate` (R users may prefer `summary` or `print`, instead). This function produces a table showing the estimated parameters, their asymptotic standard errors, information criteria and some diagnostic checks on auto-correlation, Gaussianity and heteroskedasticity. If the model is not acceptable, the model or the additional options ought to be re-formulated with the previous functions.

If the model is judged appropriate, then the optimal states may be obtained either with `UCfilter` or `UCsmooth`, depending on whether filtered or smoothed output is required. Filtered states are obtained with the Kalman filter that takes into account the information up to the current time stamp, while smoothed states are estimated by the fixed interval smoother and are based on the whole sample. Note that full covariance matrices of states are not returned by these functions, but only their diagonal or just the variances of the states in field

Name	Description
y	Single time series output data.
u	Matrix of external regressors affecting only the observation equation.
model	Model string.
stepwise	Stepwise algorithm TRUE/FALSE.
periods	Vector of periods to include in seasonal component.
tTest	Unit root tests TRUE/FALSE for stepwise algorithm.
p0	Initial parameters for estimation.
h	Forecasting horizon.
criterion	Information criterion for identification.
verbose	Show intermediate results in estimation or identification.
arma	Identify ARMA irregulars TRUE/FALSE.
outlier	Critical value for outliers.

Table 3: Input fields of **UComp** objects.

P of any **UComp** object. If desired, the disturbances involved in the model may be estimated with the disturbance smoother, implemented in UCdisturb; see details of these recursive algorithms in Durbin and Koopman (2012); De Jong (1991) and references therein. Finally, the unobserved components may be estimated by UCcomponents (and plotted with plot), that in fact is nothing else than the grouping of smoothed states estimated by UCsmooth. If convergence problems are encountered, the user can try different starting points for the parameters using the function getp0, which provides the initial parameters **UComp** would select to start the search for the optimization with their names.

An inspection of the inputs to function UCsetup (or UC or UCmodel) is useful to highlight the power and flexibility of **UComp**. All these inputs are actually fields of any **UComp** object (listed in Table 3):

- Obvious inputs to these functions are the output and input data of the model. The output data is y, a data vector or a time series object in R. Exogenous inputs to the model affecting only the observation equation are represented by a numerical or time series matrix u. If output forecasts are required in an input-output model, matrix u must contain future values for inputs. Only y is mandatory in R if it is a time series object, but frequency (the number of observations per year) needs to be supplied as well in MATLAB/Octave.

- The key input is model, which selects the model to be identified or estimated. It is a single string that indicates the type of model for each component. It allows two formats, either "trend/seasonal/irregular" or "trend/cycle/seasonal/irregular". The possibilities available for each component are discussed in the previous section, but a question mark indicates that such component is not known and **UComp** must select it:

  - Trend: ? / none / rw / irw / llt / dt.
  - Cycle: ? / none / combination of positive or negative numbers, see below.
  - Seasonal: ? / none / equal / different.
  - Irregular: ? / none / arma(0,0) / arma(p,q), with p and q integer positive orders.

Name	Description
After running `UCestim` or `UCmodel`:	
`p`	Estimated parameters.
`v`	Estimated innovations.
`yFor`	Output forecasts.
`yForV`	Output forecast variances.
`criteria`	Criteria value for selected model.
After running `UCvalidate`:	
`table`	Estimation and validation table.
After running `UCfilter`, `UCsmooth` or `UCdisturb`:	
`yFit`	Fitted values of output.
`yFitV`	Variance of fitted values of output.
`a`	State estimates.
`P`	Variance of state estimates.
After running `UCdisturb`:	
`eta`	State perturbations estimates.
`eps`	Observed perturbations estimates.

Table 4: Output fields of **UComp** object.

- Other standard inputs that manage the workflow are: (i) stepwise or full procedure (`stepwise`, may be either `TRUE` or `FALSE`, see previous section); (ii) a vector of periods for the seasonal component (`periods`), normally the fundamental period and all its harmonics ($P_j = P_1/j$, $j = 1, 2, \ldots, [P_1/2]$, see previous section); (iii) forecasting horizon (`h`); (iv) information criterion for model selection (`criterion`, either `"aic"`, `"aicc"` or `"bic"`) ; (v) verbose intermediate output during estimation or identification process (`verbose`, may be `TRUE` or `FALSE`); (vi) use of ARMA identification of innovations (`arma`, either `TRUE` or `FALSE`, see Module 3 in the previous section).

- The user may select arbitrary initial conditions for the parameters to start the maximum likelihood estimation. A lot of work has been done to look for appropriate general initial values, and actually **UComp** provides its choices for any model calling function `getp0`, see the sales example in Section 5.

- The possibility of running the stepwise procedure shown in the previous section with unit root tests is triggered by setting both `tTest` and `stepwise` to `TRUE`.

- The automatic detection of outliers can be activated by selecting a positive number for `outlier`, which stands for the cutting t-test for the inclusion of outliers in the model (a recommended value is 4).

Setting cycles in `model` deserves further consideration. Any positive number sets the period of the cycle while negative values are taken as initial estimates (in absolute value) to search for the optimization of the period, along with the rest of the parameters. Several cycles with any combination of positive or negative values are possible and if a question mark is included, the existence of cycles is tested like any other component.

UComp is able to handle the input `model` with entire flexibility. For example, the following cases are valid specifications when there is no cycle: `"?/?/?"`, `"?/none/arma(1,1)"`,

`"llt/?/arma(0,0)"`, `"dt/equal/arma(0,0)"`. In the case of cycles, all the following specifications have different meanings: `"?/?/?/?"`, `"?/24/?/?"`, `"?/-24/?/?"`, `"?/24?/?/?"`, `"?/-24?/?/?"`. Several cycles are possible as well, the following are valid specifications in this case: `"?/-48+24/?/?"`, `"?/+48-24/?/?"`, `"?/-48+24?/?/?"`, `"?/+48-24?/?/?"`.

The syntax of any function in the middle section of Table 1 is fixed. They all work on an input **UComp** object and return the same or a different **UComp** object that stores the results of the function operations. Table 4 shows the fields filled in by these functions.

5. Examples

Several examples are shown in this section to illustrate the power, flexibility and ease of use of the library. Code listings are provided mainly in R. The reason is that the code is very similar in any environment, and then MATLAB/Octave code is restricted to some cases in the first example. The code is simplified to make the text more readable, and comprehensive R code to obtain all the results and figures in this paper is included in the supplementary material. Some of the examples are also provided in code written for MATLAB/Octave [1].

5.1. US air passengers data

This well-known time series taken from Box, Jenkins, Reinsel, and Ljung (2015) offers an excellent opportunity to show the power of the algorithms described in the previous sections. The reason is that the UC identification highlights some interesting properties that result in better forecasts, improving upon standard UC models as well.

Taking advantage that there is a standard function in R that loads the data (`AirPassengers`), the following listing shows the commands to fit a standard Basic Structural Model (BSM) to the variable in logs. All the results are stored in an **UComp** object called `m`.

```
R> y <- log(AirPassengers)
R> autoplot(AirPassengers, ylab = "Thousands")
R> m <- UC(y, model = "llt/equal/arma(0,0)")
R> summary(m)
```

The main call in MATLAB/Octave code would be

```
m = UC(y, 12, 'model', 'llt/equal/arma(0,0)');
```

Note that both code chunks are very similar, with the difference that the number of observations per year have to be supplied compulsorily to MATLAB/Octave. In the latter case, also the inputs are supplied in pairs, the first element of which is always a string representing the input argument name.

Calling the `UC` function in reality is equivalent to sequentially calling `UCmodel`, `UCdisturb`, `UCvalidate` and `UCcomponents`. When forecasts are only required in contexts where the

[1] The post-processing stage of the paper revealed that the same source code compiled on different systems (Windows 10, Linux, etc.) resulted in tiny differences in the estimation of the models. In no case did such differences imply a change in the conclusions drawn for any of the analyses. The results shown in this section were obtained with R version 4.1.2 (2021-11-01) on an x86_64-w64-mingw32/x64 (64-bit) machine running Windows 10 x64.

computing time is to be optimized, running `UCmodel` would provide the forecasts avoiding the rest of operations. The following listing shows the estimation results, that are actually the output of `UCvalidate` (`print` or `summary` in R would produce the same output).

```
------------------------------------------------------------
  Model: llt/none/equal/arma(0,0)
  Periods:  12.0 /  6.0 /  4.0 /  3.0 /  2.4 /  2.0
  Q-Newton: Function convergence
  (*)  concentrated out parameters
  (**) constrained parameters during estimation
 ------------------------------------------------------------
                      Param   asymp.s.e.        |T|     |Grad|
 ------------------------------------------------------------
        Level:     2.98e-04*
        Slope:       0.0000**
    Seas(All):     3.56e-06     8.63e-06     0.4123   2.21e-05
    Irregular:     2.34e-04     7.08e-05     3.3093   2.35e-05
 ------------------------------------------------------------
   AIC:      -2.7807   BIC:      -2.4508   AICc:      -2.7530
           Log-Likelihood:     216.2139
 ------------------------------------------------------------
    Summary statistics:
 ------------------------------------------------------------
         Missing data:
         Q( 1):       1.1707         Q( 4):       5.2652
         Q( 8):       6.7149         Q(12):      10.3053
   Bera-Jarque:       1.5154       P-value:       0.4687
       H(  44):       0.6122       P-value:       0.1073
------------------------------------------------------------
```

This table is divided into four sections. The first section shows general information about the model and the estimation process, in particular the harmonics included in the model, and some information about numerical optimization. The second section presents the parameter values, their transformed counterparts, the asymptotic standard error, the ratio of the two and the gradient at the optimum. A variance with an asterisk marks the one that has been concentrated out of the likelihood. Some parameters are marked with two asterisks to highlight that this parameter has been constrained along the estimation procedure because it reached its boundary. Note that standard errors and gradients are shown only for parameters that have been estimated explicitly (only two in this case). The third section shows the information criteria and log-likelihood value at the optimum. The final section shows some diagnostic statistics about the innovations, such as the Ljung-Box Q autocorrelation test for several lags (that should be compared with a χ^2 distribution with $n - npar$ degrees of freedom, where n is the sample length and $npar$ are the number of parameters), a Gaussianity and a variance test ratio (Ljung and Box 1978; Bera and Jarque 1981). Innovations are stored in `m$v` (or `m.v`) so that any further tests with any other library may be run.

The estimated model shows that a random-walk-with-drift trend appears naturally since the slope variance is strictly zero (constrained during estimation), and there is no strong evidence against it because innovations are uncorrelated, Gaussian and homoskedastic.

The following listing shows the code to identify the best possible model. Several options are shown to illustrate the flexibility of **UComp**, namely the full algorithm and the stepwise algorithm with and without unit root tests. Both stepwise procedures result in the same model as the full one. Note that the call to `UC` only requires the output time series when declared as an R time series object (and `frequency` in MATLAB/Octave code). If the data is stored as a numerical vector, then an additional input is mandatory, namely `periods` that specifies explicitly the period for the seasonal component and all or part of its harmonics. In any case, `periods` may be chosen as any set of arbitrary values and the seasonality test in Module 1 (see previous section) would select the most relevant among them.

```
R> m <- UC(y)
R> m <- UC(y, stepwise = TRUE, tTest = TRUE)
R> m <- UC(y, stepwise = TRUE)
```

For the sake of comparison, the listing is also shown below in MATLAB/Octave format.

```
m = UC(y, 12);
m = UC(y, 12, 'stepwise', true, 'tTest', true);
m = UC(y, 12, 'stepwise', true);
```

```
-------------------------------------------------------------
  Model: llt/none/different/arma(0,0)
  Periods:  12.0 /  6.0 /  4.0 /  3.0 /  2.4
  Q-Newton: Function convergence
  (*)  concentrated out parameters
  (**) constrained parameters during estimation
 -------------------------------------------------------------
                      Param   asymp.s.e.        |T|     |Grad|
 -------------------------------------------------------------
       Level:     2.34e-04     8.68e-05     2.6950   1.13e-04
       Slope:       0.0000**
   Seas(12.0):    1.10e-05     1.86e-05     0.5930   6.93e-05
    Seas(6.0):    5.17e-06     2.11e-05     0.2447   7.32e-05
    Seas(4.0):      0.0000**
    Seas(3.0):    2.19e-06     2.01e-05     0.1092   5.93e-05
    Seas(2.4):    1.24e-06     9.62e-06     0.1293   5.84e-05
    Irregular:    3.45e-04*
 -------------------------------------------------------------
   AIC:      -2.9056   BIC:      -2.5138   AICc:      -2.8640
           Log-Likelihood:     228.2060
 -------------------------------------------------------------
    Summary statistics:
 -------------------------------------------------------------
         Missing data:
         Q( 1):       0.0229         Q( 4):       1.3965
         Q( 8):       2.9684         Q(12):       8.1034
   Bera-Jarque:       5.0102       P-value:       0.0817
       H(  46):       0.5517       P-value:       0.0464
-------------------------------------------------------------
```

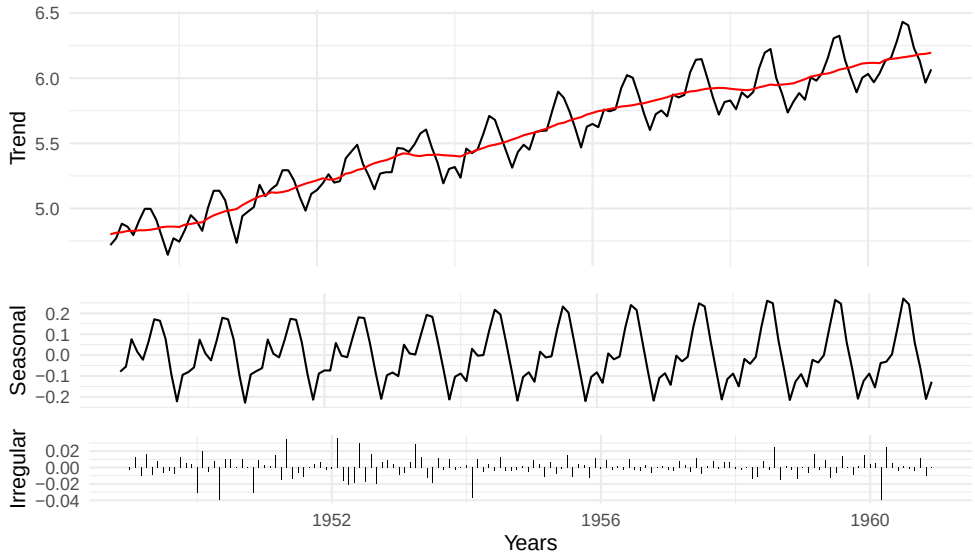

Figure 1: Estimated optimal components for the air passengers data.

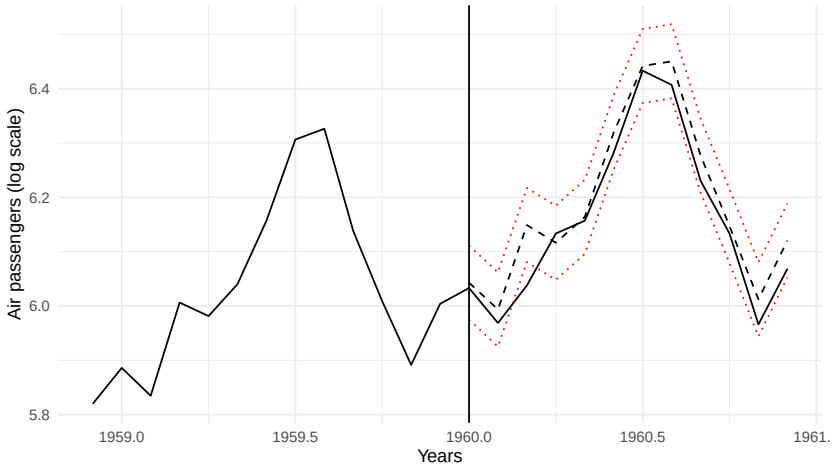

Figure 2: Forecasts for air passengers data with 95% prediction intervals.

The model finally identified with the whole sample has some interesting features, namely: (i) the trend model is again a random-walk-with-drift; (ii) the Nyquist seasonal frequency is not relevant; (iii) the seasonal component selected has different variances for each harmonic, with the one corresponding to period 4 constrained to zero during estimation, and the rest ranging between 1.24×10^{-6} and 1.10×10^{-5}; (iv) the irregular component is white noise with variance 3.45×10^{-4}; v) no correlation is left in the innovations (the Ljung-Box statistic with 12 lags is 8.10).

The estimated components and the forecasts with 95% prediction intervals are shown in Figures 1 and 2, respectively. Components are produced by the `UCcomponents` function and stored in `m$comp` or `m.comp` (variances in `m$compV` or `m.compV`). Forecasts and their variances are stored in `m$yFor` (`m.yFor`) and `m$yForV` (`m.yForV`), respectively.

Criteria	Log-Lik	AIC	BIC	AICc
AIC	228.206	−2.906	−2.514	−2.864
BIC and AICc	222.713	−2.885	−2.576	−2.864
BSM model	216.214	−2.781	−2.451	−2.753

Table 5: Values of different criteria depending on the criterion used to select the best model for the natural logarithms of the air passengers data. The first and second rows show the criteria values for the best models identified with AIC, BIC and AICc. The third row shows the standard BSM with the seasonal component including all harmonics and one single variance for all of them.

Table 5 shows some complementary information, namely the values of the log-likelihood and the information criteria for different UC specifications. The first two rows in Table 5 show all the criteria values when each criterion (AIC, BIC and AICc) is used in turn to select the optimal model. The AIC selects the model in the previous paragraph, reaching the maximum value of the likelihood possible (228.206). However, BIC and AICc preferred the model with equal variances for all harmonics (but excluding the Nyquist one). As is generally acknowledged, BIC and AICc are more demanding in terms of parsimony.

In order to check the ability of the automatic algorithm to find the optimal model, the criteria values for the standard BSM are included in Table 5. The BSM is worse than models in the previous rows according to all criteria because it includes all harmonics (even the one for the Nyquist frequency) and variances of harmonics are constrained to be the same. This suggests that a single parameter for all harmonics, including the one that is not present, is producing a poorer model.

All versions of the identification algorithm (full, stepwise with and without unit root tests) end up selecting the same model.

Forecasting performance is finally tested based on an experiment with rolling origins for 12 months ahead forecasts, starting on observation 108 and moving one month at a time with re-identification of all models at each single step. Table 6 shows the mean of all the mean absolute scaled error metrics (MASE, Hyndman and Koehler 2006) for ETS, ARIMA, UC, BSM, seasonal naïve and the mean and median of all of them. The conclusions are the same for symmetric mean absolute percentage error metrics (sMAPE, Makridakis and Hibon 2000).

Results are cut clear: (i) all models outperform the seasonal naïve by a wide margin; (ii) median performs better than mean, ARIMA, and ETS; (iii) ARIMA is better than ETS; (iv) BSM is better than all of them (except for the first two months); (v) still the UC identified by the procedures proposed in this paper is the best of all for every single horizon.

5.2. Monthly sales index for food in large retail stores in Spain

Figure 3 shows the monthly index for food in large retail stores in Spain. The series exhibits a very strong seasonal pattern with a clear peak in December each year and a growing trend in the first half of the sample followed by a long stagnation period after 2008.

The listing below shows the commands to separate the last year's data as a validation set, create a time series object based on the logs of the raw data and run a battery of nine models, some of which include the identification procedure. In this listing `u` is a matrix containing one calendar effect, a leap year effect created with R package **tsoutliers** (López-de-Lacalle 2019).

h	ETS	ARIMA	UC	BSM	Naïve	Mean	Median
1	0.4469	0.3634	0.3436	0.3853	1.0548	0.4306	0.3720
2	0.5302	0.4150	0.3888	0.4194	1.0724	0.4672	0.4121
3	0.6100	0.4747	0.4254	0.4651	1.0777	0.5136	0.4623
4	0.6538	0.5171	0.4442	0.4808	1.1003	0.5390	0.4808
5	0.6755	0.5419	0.4466	0.4874	1.1279	0.5476	0.4907
6	0.6790	0.5664	0.4468	0.4881	1.1588	0.5515	0.4952
7	0.6841	0.5853	0.4500	0.4835	1.1925	0.5565	0.4958
8	0.6796	0.6084	0.4530	0.4818	1.2216	0.5588	0.4980
9	0.6754	0.6301	0.4571	0.4846	1.2454	0.5571	0.5033
10	0.6697	0.6446	0.4607	0.4835	1.2717	0.5582	0.5047
11	0.6492	0.6552	0.4630	0.4825	1.2952	0.5556	0.5022
12	0.6384	0.6666	0.4640	0.4816	1.3173	0.5537	0.5024

Table 6: Mean of MASE metrics for 1 to 12 steps ahead forecasts of air passengers data based on rolling forecasting origins starting on observation 108.

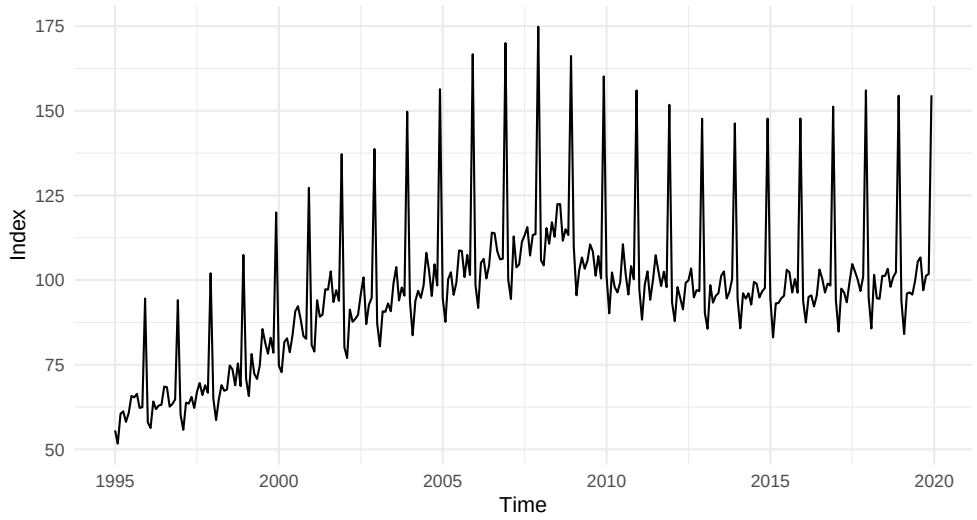

Figure 3: Monthly sales index for food in large retail stores in Spain from 1995 to 2019.

The leap year effect takes into account February months with 29 days. Several specifications of Easter and trading day effects were also included initially, but they turned out to be not significant.

The models called `m1` store the results of the identification procedure with the full and both stepwise algorithms (the name is the same because the resulting models are the same). Models `m2` repeat the identification procedure, but including the calendar effects and models `m3` expand the previous models to take into account the automatic outlier identification by selecting all those with a t-test value greater or equal to 4.

```
R> y <- window(y, end = c(2018, 12))
R> yl <- log(y)
```

```
R> u <- calendar.effects(sales, trading.day = FALSE, leap.year = TRUE,
+    easter = 0)
R> m1 <- UC(yl)
R> m1 <- UC(yl, stepwise = TRUE)
R> m1 <- UC(yl, stepwise = TRUE, tTest = TRUE)
R> m2 <- UC(yl, u = u)
R> m2 <- UC(yl, u = u, stepwise = TRUE)
R> m2 <- UC(yl, u = u, stepwise = TRUE, tTest = TRUE)
R> m3 <- UC(yl, u = u, outlier = 4)
R> m3 <- UC(yl, u = u, outlier = 4, stepwise = TRUE)
R> m3 <- UC(yl, u = u, outlier = 4, stepwise = TRUE, tTest = TRUE)
```

By default, the qualifier `verbose` in function `UC` or `UCmodel` is set to `FALSE`. By switching it to `TRUE` a list of all the models with their information criteria is shown in table form. That allows the user to check if the best model is very different from others or if all the criteria point toward the same model. Below is a truncated output of the command to estimate model `m1` above (note that the full table is 23 models long).

```
  ------------------------------------------------------------
         Model                          AIC      BIC     AICc
  ------------------------------------------------------------
    none/none/none/arma(0,0):  -0.1600  -0.1473  -0.1600
        none/none/equal/none:   1.8040   1.9566   1.8074
   none/none/equal/arma(0,0):  -0.2121  -0.0468  -0.2087
                 *** (17 models removed) ***
      dt/none/equal/arma(0,0):  -3.6690  -3.4655  -3.6621
       dt/none/different/none:  -3.5437  -3.2894  -3.5333
  dt/none/different/arma(0,0):  -3.7056  -3.4385  -3.6952
  ------------------------------------------------------------
```

The output for model `m3` is shown in the following listing. Mind that the model exhibits certain complexity, as it involves 12 parameters for a univariate time series. The model is composed of a local linear trend with a seasonal component with different variances for each harmonic, a leap year effect that assumes an increase of 2.48% every February with 29 days, i.e., $(e^{0.0245} - 1) \times 100$; and two outliers, a step starting on December 2001 (observation 84) and a step in the slope at May 2008 (observation 161). Figures 4 and 5 show the estimated components and the forecasts for model `m3`.

```
------------------------------------------------------------
  Model: llt/none/different/arma(0,0) + inputs
  Periods:  12.0 /  6.0 /  4.0 /  3.0 /  2.4 /  2.0
  Q-Newton: Function convergence
  (*)  concentrated out parameters
  (**) constrained parameters during estimation
 ------------------------------------------------------------
                      Param   asymp.s.e.        |T|     |Grad|
 ------------------------------------------------------------
```

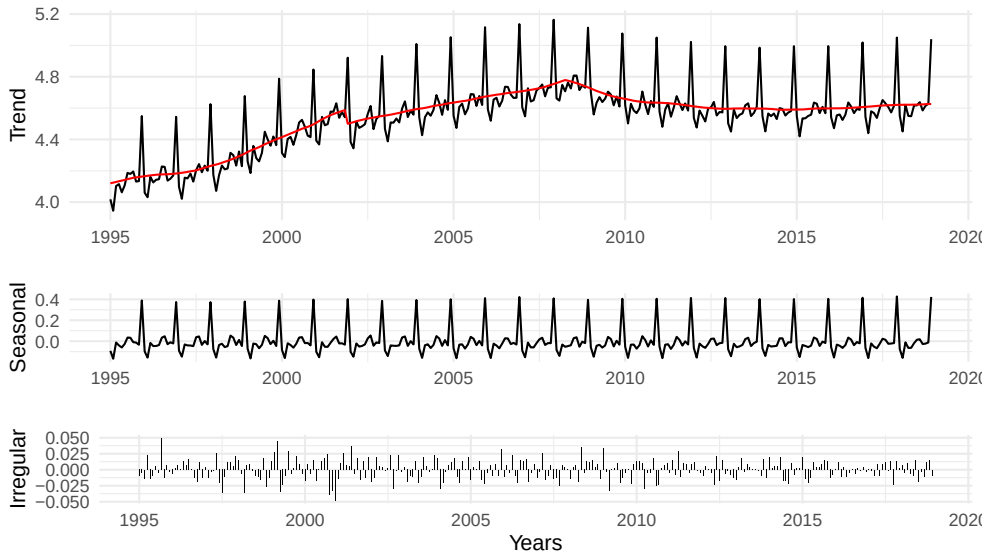

Figure 4: Components of retail sales in Spain.

```
      Level:     1.61e-05     1.83e-05     0.8770   2.87e-05
      Slope:     3.30e-07     3.06e-06     0.1079   4.04e-05
 Seas(12.0):     5.89e-07     6.20e-06     0.0949   4.38e-05
  Seas(6.0):     4.42e-07     5.58e-06     0.0792   1.66e-05
  Seas(4.0):       0.0000**
  Seas(3.0):     1.75e-05     1.01e-05     1.7242   8.24e-05
  Seas(2.4):     4.13e-08     4.46e-06     0.0093   3.29e-05
  Seas(2.0):     2.51e-07     6.02e-06     0.0418   3.62e-05
  Irregular:     4.18e-04*
    Beta(1):       0.0245*      0.0100     2.4476
       LS84:      -0.0930*      0.0153     6.0593
      SC161:      -0.0107*      0.0024     4.4456
------------------------------------------------------------
  AIC:      -3.7289   BIC:      -3.4237   AICc:      -3.7151
          Log-Likelihood:     560.9683
 ------------------------------------------------------------
```

One interesting feature of **UComp** is that the user can choose arbitrary initial parameters to start the quasi-Newton search with the function `getp0`. This function provides the particular choice of initial parameters done by **UComp**. Once the parameters are suggested, the user may change any or all of them to preferred values and put them into a call to standard functions. A typical code flow is shown in the following listing.

```
R> p0 <- getp0(yl, model = "llt/equal/arma(0,0)")
R> p0[1] <- 0
R> m <- UC(yl, model = "llt/equal/arma(0,0)", p0 = p0)
```

The printout of `p0` shows the values and labels of each parameter. In this example, the first parameter (the level variance) is initialized at 0 in a standard BSM model. It should be

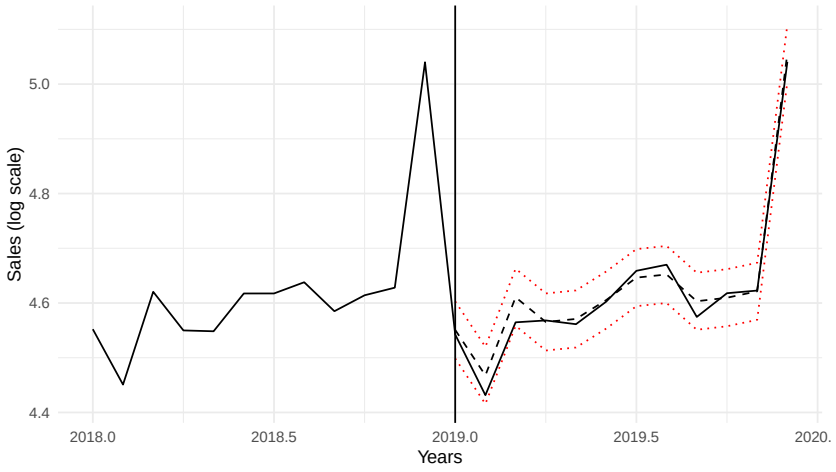

Figure 5: Forecasts of sales data in Spain with 95% prediction intervals.

noted that the estimation may or may not converge starting on the new initial conditions and therefore users should use the handling of initial parameters with extreme care and at their own risk. To assist the user in this task, many checks have been added in the code to avoid meaningless initial parameters, such as negative variances, damping parameters smaller than zero or greater than one, non-stationary and/or non-invertible ARMA irregular components, etc.

Forecasting performance is tested based on a rolling experiment for 12 months ahead horizons similar to the previous example, starting on observation 228 (December 2013). Table 7 shows the overall results in a similar format to Table 6.

One remarkable fact is that the naïve model performance is much better than in the previous example, due to the flatness of the time series in the test period. As a matter of fact, it is better than ETS. This time mean is better than median and struggles to be the best against UC (both are actually close to each other). ARIMA lies in a middle range between the extremes and BSM is not good. One point worth mentioning is that, given BSM is not particularly good (it is a fixed UC structure commonly used in the UC literature), the identification procedure implemented in **UComp** managed to find a better UC model according to the forecasting point of view.

5.3. Cycles

Business cycle analysis of macro- or micro-economic variables is an area that has attracted much interest among researchers (see, e.g., Leser 1961; Hodrick and Prescott 1997; Baxter and King 1999; Gómez and Maravall 2001; Galati, Hindrayanto, Koopman, and Vlekke 2016). **UComp** offers the possibility to analyze this issue from different points of view, as is shown in this section on the OECD seasonally adjusted quarterly real gross domestic product (GDP). The data are available from 1962 to 2019 and are shown in Figure 6.

Apparently, there is little variation about a smooth trend with an abrupt jump in 2009 as a result of the recession. The following listing shows the code to generate four different models to deal with these data in logs. Model `m1` is just a bottom line benchmark to compare with the rest; model `m2` represents the standard output when the cycle wanted to be identified (the

h	ETS	ARIMA	UC	BSM	Naïve	Mean	Median
1	0.4392	0.3901	0.3800	0.4318	0.4499	0.3819	0.3933
2	0.4597	0.4082	0.3832	0.4456	0.4456	0.3895	0.4040
3	0.4634	0.4203	0.3885	0.4498	0.4508	0.3950	0.4127
4	0.4728	0.4284	0.3930	0.4558	0.4530	0.3980	0.4167
5	0.4793	0.4320	0.3950	0.4600	0.4542	0.3998	0.4210
6	0.4741	0.4246	0.3937	0.4587	0.4553	0.3949	0.4154
7	0.4744	0.4214	0.3921	0.4578	0.4568	0.3924	0.4095
8	0.4726	0.4176	0.3927	0.4516	0.4586	0.3901	0.4063
9	0.4687	0.4159	0.3925	0.4504	0.4590	0.3893	0.4042
10	0.4693	0.4192	0.3982	0.4535	0.4592	0.3911	0.4064
11	0.4682	0.4211	0.4016	0.4535	0.4595	0.3917	0.4073
12	0.4663	0.4206	0.4050	0.4539	0.4590	0.3910	0.4068

Table 7: Mean of MASE metrics for 1 to 12 steps ahead forecasts of the sales at large retail stores data based on forecasts run at moving forecasting origins starting on observation 228.

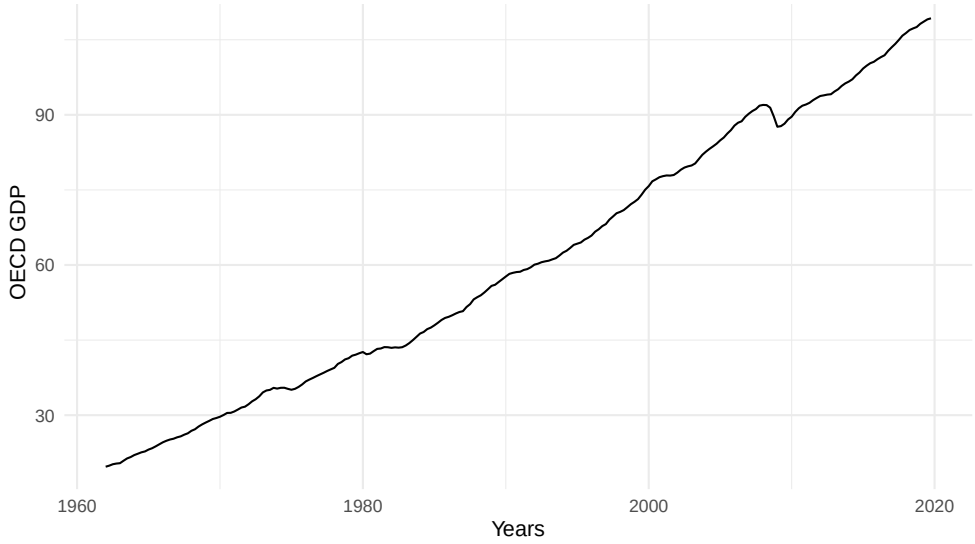

Figure 6: OECD quarterly real gross domestic product from 1962 to 2019.

inputs could have recognized the fact that the data are seasonally adjusted by selecting `model = "?/?/none/?"`); and models `m3` and `m4` replace the cycle with irregular components of type AR(2) or ARMA(2, 1), respectively, following Clark (1987) and Harvey and Jaeger (1993).

```
R> yl <- log(OECDgdp)
R> m1 <- UC(yl)
R> m2 <- UC(yl, model = "?/?/?/?")
R> m3 <- UC(yl, model = "?/?/arma(2,0)")
R> m4 <- UC(yl, model = "?/?/arma(2,1)")
```

Note that the stepwise algorithms are not worthy in this case, because the time series is not seasonal and the population of models reduces to 7 models. In addition, when any of the stepwise algorithms is used the output is always the same.

	Model	Log-Lik	AIC	BIC	Period
1	`dt/none/none/none`	901.183	−7.743	−7.698	
2	`dt/-16/none/none`	911.519	−7.806	−7.717	15.495
3	`llt/none/none/arma(2,0)`	912.917	−7.818	−7.729	29.446
4	`llt/none/none/arma(2,1)`	912.894	−7.809	−7.705	31.320

Table 8: Different specifications of UC models and some statistics for OECD GDP. The last column shows the estimated period of the cycle in quarters. For the ARMA models the calculation is based on the complex roots of the AR polynomial.

It is worth noting that in model `m2` the cycle is tested under the same conditions as the rest of the components, and therefore the optimal result could be a model without a cycle. However, the cycle is initially imposed on models `m3` and `m4` under an ARMA(p, q) disguise, though zero estimation of the parameters involved would indicate the absence of such a cycle.

Two words of caution are important at this point:

- Though it is possible in **UComp**, it is not generally recommended to mix cycles with irregular ARMA(p, q) components in the same model, because identification problems may arise as a consequence of both trying to model the cycle effectively. It may work in some cases, but the analysis should be done with extreme precaution to avoid nonsensical conclusions.
- The approach followed by models `m3` and `m4` above follow the tradition of model-based business cycle literature which specifies ARMA(p, q) models for the irregular component, and which checks the cycle period on the complex roots arising from the estimated AR polynomial (see, e.g., Clark 1987; Harvey and Jaeger 1993; Gómez and Maravall 2001). However, the estimation based on band-pass or low-pass filters does not care about the stochastic properties of the estimated cycles; one would simply call 'cycle' or 'trend-cycle' what comes out of the procedure (Leser 1961; Hodrick and Prescott 1997; Baxter and King 1999). This means that, from this point of view, even a non-periodical ARMA model would qualify as a cycle. Whether this is reasonable or not enters into an endless discussion about the exact meaning of the components that is beyond the scope of this paper.

The cycle can be imposed on model `m2` by removing the question mark and adding the period of the cycle with a plus or minus sign. A negative value (e.g., `model = "?/-16/?/?"`) gives **UComp** the initial condition for the period of the cycle, and the best model shall necessarily have a cycle. A positive value (e.g., `model = "?/16/?/?"`) imposes the period of the cycle and only its damping factor and the disturbance variance noise are estimated.

Table 8 shows a summary of the output. The cycle periods in ARMA models are calculated based on the complex roots of the AR polynomials. Strictly speaking, the best cycle estimate is specified as an ARMA$(2, 0)$ model, though in reality the three models with the cycle are very similar. The main difference is in the cycle period estimate. However, as the period for `m2` is approximately half of the other two, it may be well the case that the trigonometric model is concentrating on a harmonic of the cycle estimated by `m3` and `m4`.

The following listing shows a shortened version of the model with the trigonometric cycle `m2`. The model trend is close to an IRW trend since the level variance is constrained to zero during estimation.

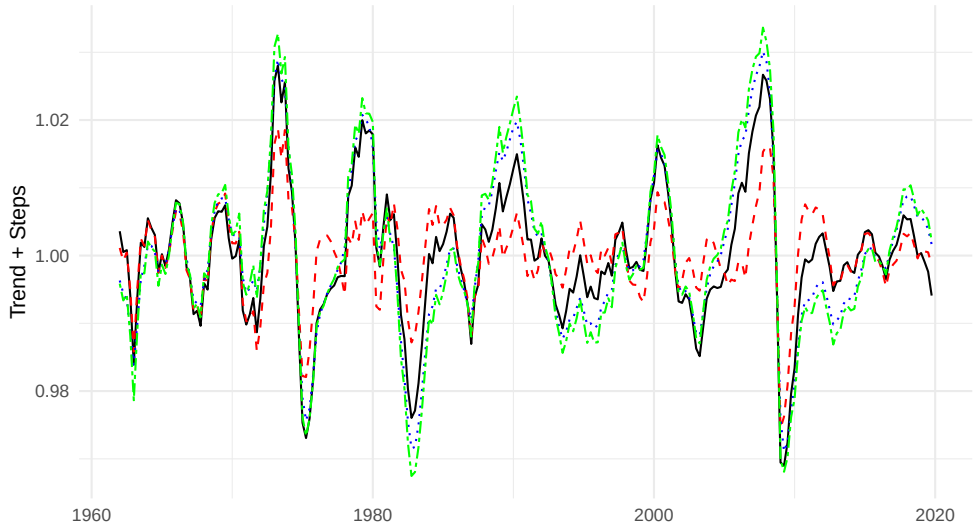

Figure 7: Four different measurements of the cycle. Hodrick-Prescott (black-solid); trigonometric cycle (red-dashed); ARMA$(2,0)$ (blue-dotted); and ARMA$(2,1)$ (green-dashed-dot).

```
-------------------------------------------------------------
  Model: dt/-16/none/none
  Periods:
  Q-Newton: Function convergence
  (*)  concentrated out parameters
  (**) constrained parameters during estimation
 -------------------------------------------------------------
                       Param    asymp.s.e.         |T|      |Grad|
 -------------------------------------------------------------
      Damping:        0.9869        0.0052    188.1536   2.20e-06
        Level:        0.0000**
        Slope:      1.93e-06      7.89e-06      0.2447   8.10e-06
       Rho(1):        0.9091        0.0236     38.5492   8.86e-06
    Period(1):       15.4952        1.0341     14.9850   5.68e-06
       Var(1):      9.79e-06*
 -------------------------------------------------------------
   AIC:       -7.8062   BIC:       -7.7171   AICc:       -7.8062
            Log-Likelihood:      911.5185
 -------------------------------------------------------------
```

Figure 7 shows the estimation of the three cycles and the Leser-Hodrick-Prescott (LHP) cycle estimated with a smoothing constant of 1600 (estimated by the command `hpCycle = UChp(yl)`). There are clear similarities between them, but their volatility range from the more stable trigonometric cycle with a variance of 4.15^{-5} to the less stable ARMA$(2,1)$, with variance 1.53^{-4}. Correlations with the LHP filter ranges from 0.85 (trigonometric cycle) to 0.94 (AR(2)).

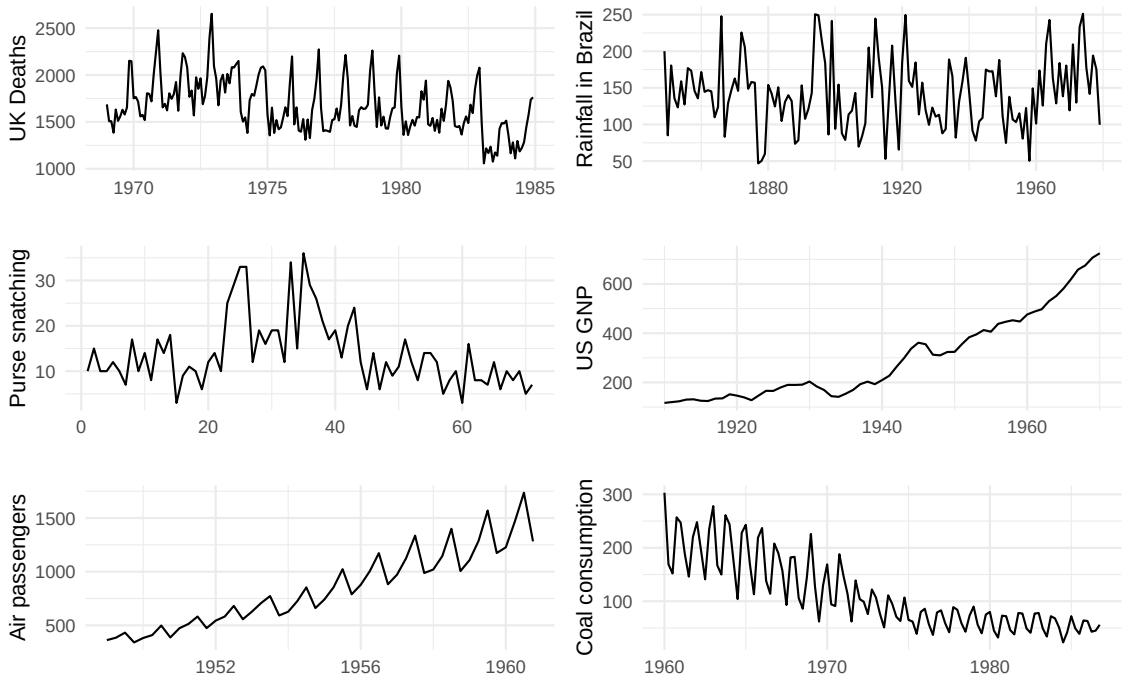

Figure 8: All series used in Section 5.4. From top to bottom and left to right: monthly fatalities and seriously injured in road accidents in the UK; annual rainfall in Fortaleza (Brazil); 28-day-period purse snatching in Chicago; annual US GNP; quarterly US air passengers in the US; and quarterly coal consumption in the UK.

5.4. Comparisons with models in Harvey (1989)

In this section all the applications of Section 2.7 of Harvey (1989) are re-estimated with **STAMP**, and compared with the results obtained by **UComp**. The aim of this section is, therefore, a direct comparison of the models obtained by different sources and software. In order to make the section agile all the information about the models is restricted to the minimum necessary. All time series involved in this section are represented in Figure 8.

Road accidents in the UK

Table 9 shows several models estimated for the logs of this time series up to 1982. The second column shows the BSM model estimated by **STAMP** which matches that reported in Harvey (1989, p. 83) and the one estimated with **UComp**. This model is rather peculiar in that the slope and seasonal variances are zero (both components are deterministic).

In fact, when the automatic identification is run on the data (third column of Table 9), the optimal trend model lacks the slope (it is actually a RW), the seasonal variance is once again zero and the last harmonic is missing, resulting in a model with somewhat better metrics, as the log-likelihood and all information criteria are better than the pure BSM in the second column.

When the last optimal model selected is re-estimated with the full sample and with automatic detection of outliers, the outcome is the model in the fourth column of Table 9, that is further

	Sample up to 1982		Full sample and outliers	
Parameter	**STAMP** & **UComp**	**UComp** optimal	**UComp**	**UComp** optimal
Level	5.853×10^{-4}	5.145×10^{-4}	5.269×10^{-4}	5.173×10^{-4}
Slope	0	–	–	–
Seasonal	0	0	7.393×10^{-7}	1.589×10^{-6}
Irregular	3.703×10^{-3}	3.787×10^{-3}	4.030×10^{-3}	3.983×10^{-3}
LS Feb1983	–	–	−0.241	−0.245
Log-Lik	141.362	152.454	174.511	179.138
AIC	−1.492	−1.660	−1.662	−1.720
BIC	−1.195	−1.418	−1.407	−1.483

Table 9: Models estimated by **UComp** and reported in the literature on road accidents in the UK. The second column is the model in Harvey (1989, pp. 83). The third column is the optimal model identified by **UComp**. The fourth and fifth columns show the models re-estimated with the full sample and outlier detection.

refined by the full identification procedure (in the fifth column). The only difference between these two latter models is that the fully optimized lacks the Nyquist harmonic in the seasonal component.

The code to replicate the models in Table 9 is shown in the following listing. `m1` is the model also determined in Harvey (1989, pp. 83); `m2` is the model determined using the automatic identification, `m3` the model with outliers and the full sample and `m4` the optimal model with outliers.

```
R> y <- log(UKDriverDeaths)
R> m1 <- UC(subset(y, end = 168), model = "llt/equal/arma(0,0)")
R> m2 <- UC(subset(y, end = 168))
R> m3 <- UC(y, outlier = 4, model = "rw/equal/arma(0,0)")
R> m4 <- UC(y, outlier = 4)
```

Rainfall in north-east Brazil and purse snatching in Chicago

The rainfall time series is interesting because it is sampled at an annual rate and is used to look for modeling the cycle, see Harvey (1989, pp. 86). There are some differences between the model estimated in Harvey (1989), shown in the second column of Table 10 and the same models estimated by **STAMP** and **UComp** (that are identical, third column). Still, the optimal model obtained with **UComp** (in the fourth column) is not the model with the cycle but a simple AR(1) with a constant. This latter model has better metrics.

Those models are replicated with **UComp** with the code in the listing below, assuming `y` is a vector containing the rainfall data. Note that the first call to function `UC` imposes a cycle in the model, while the second tests for it and actually selects the AR(1) model as best, rather than the model with the cycle.

```
R> m1 <- UC(y, model = "?/-8/?/?")
R> m2 <- UC(y, model = "?/?/?/?")
```

	Rainfall in Fortaleza			Purse snatching in Chicago		
Param.	Harvey (1989)	**STAMP** & **UComp**	**UComp** optimal	Param.	**STAMP** & **UComp**	**UComp** optimal
Rho	0.84	0.832	–	–	–	–
Period	15.32	15.624	–	Level	5.151	–
Var	214	229.457	–	AR(1)	–	−0.301
Irregular	1593	1582.780	2187.159	AR(2)	–	−0.392
AR(1)	–	–	−0.242	Irregular	24.782	35.839
Const.	142.2	142.243	142.521	Const.	–	13.475
Log-Lik	–	−686.250	−685.284	Log-Lik	−228.679	−225.818
AIC	–	10.538	10.493	AIC	6.498	6.446
BIC	–	10.626	10.537	BIC	6.562	6.541

Table 10: Estimated models for rainfall in Fortaleza (Brazil) using different software and configurations in the left half of the table. Purse snatching in Chicago models in the right half.

The right side of Table 10 shows some estimated models for purse snatching in Chicago. For this series there are some differences in the random-walk-plus-noise model estimated in Harvey (1989) and **STAMP** and **UComp**, but they are really small (not shown). However, the optimal model according to **UComp** is an AR(2), with marginally better information criteria and log-likelihood values. The code to replicate those models is in the following listing.

```
R> m1 <- UC(y, model = "rw/none/arma(0,0)")
R> m2 <- UC(y)
```

Trends and cycles in US macroeconomic time series and quarterly airline passengers

The description of these two examples in Harvey (1989) are somewhat odd, because it was impossible to replicate them, although both **STAMP** and **UComp** provided identical results. With respect to annual US GNP data, the problem may lie on the fact that Harvey (1989) keeps referring to the data from 1909, whereas the database included in the book starts in 1910 and it is not clearly stated which sample was used in the models shown. Even the ARIMA$(2, 1, 0)$ model suggested in the book did not match the one with the published data. Nevertheless, the model estimated by **STAMP** and **UComp** is in the second column of Table 11, with a cycle of 7.64 years and no irregular component. The best model according to **UComp**, however, is a damped trend with no irregular component (third column). The code reproducing the models on the left hand side of Table 11 is in the following listing.

```
R> m1 <- UC(log(y), model = "llt/-8/none/arma(0,0)")
R> m2 <- UC(log(y), model = "?/?/?/?")
```

The models for the quarterly airline data reported in Harvey (1989) suffer from the same problems, they are rather odd, but, once more, there is an exact match between the models estimated by **STAMP** and **UComp**. A final interesting point is that the optimal model is one with a random walk with drift trend, no irregular component and different variances for both harmonics. The code is in the following listing.

	US GNP			Quarterly airline passengers	
Param.	**STAMP** & **UComp**	**UComp** optimal	Param.	**STAMP** & **UComp**	**UComp** optimal
Damping	–	0.725	–	–	-
Level	1.733×10^{-3}	1.071×10^{-3}	Level	6.273×10^{-4}	7.279×10^{-4}
Slope	3.607×10^{-4}	1.193×10^{-3}	Slope	3.634×10^{-9}	0
Rho	0.897	0.888	Seasonal	2.010×10^{-5}	–
Period	7.638	7.392	Season(4)	–	2.857×10^{-5}
Var	3.860×10^{-4}	3.562×10^{-4}	Season(2)	–	7.725×10^{-7}
Irregular	0	–	Irregular	2.284×10^{-9}	–
Log-Lik	75.499	80.715	Log-Lik	73.498	74.570
AIC	−2.246	−2.450	AIC	−2.729	−2.774
BIC	−2.004	−2.242	BIC	−2.417	−2.462

Table 11: US GNP models in the left hand part of the table estimated with **STAMP** and **UComp**. The right hand side of the table shows the models for the US airline passenger data in quarters.

Param.	Harvey (1989)	**STAMP** & **UComp**	**UComp** optimal
Level	6.42×10^{-4}	4.737×10^{-4}	2.921×10^{-3}
Slope	0.29×10^{-4}	7.073×10^{-6}	-
Seasonal	0.84×10^{-4}	0	3.804×10^{-7}
Irregular	1.14×10^{-2}	1.358×10^{-2}	1.187×10^{-2}
Log-Lik	–	37.396	37.994
AIC	–	−0.639	−0.696
BIC	–	−0.420	−0.531

Table 12: Models estimated for quarterly coal consumption in the UK with data up to the end of 1982.

```
R> y <- ts(log(colSums(matrix(AirPassengers, 3, 144 / 3))),
+    start = 1949, frequency = 4)
R> m1 <- UC(y, model = "llt/equal/arma(0,0)")
R> m2 <- UC(y)
```

Coal consumption in the UK

The last example quoted in Section 2.7 of Harvey (1989) concerns energy consumption in the UK. Three time series are mentioned, namely coal, gas and electricity, but only the explicit model for coal is shown. That model is replicated in the second column of Table 12 with data up to the last quarter of 1982. As usual, **STAMP** and **UComp** provide a model somewhat different to the one shown in the book, though exactly equal to each other (third column). The optimal model is also different and shown in the last column of Table 12. The models are estimated with the following listing, provided `y` is a vector containing the coal consumption data.

```
R> m1 <- UC(log(subset(y, end = 92)), model = "llt/eq/arma(0,0)")
R> m2 <- UC(log(subset(y, end = 92)))
```

6. Conclusion

This paper has presented **UComp**, a new library for the exploitation of UC models. It implements comprehensive procedures for identification, estimation and forecasting of such models.

The library incorporates most of the modern algorithms and advances in the field, following mainly Harvey (1989), Durbin and Koopman (2012) and many others. The system is capable of handling systems with trends, cycles, seasonal and irregular components of different types, and effects of exogenous input variables, as well as automatic identification of outliers.

The most novel aspect of **UComp** is that it incorporates automatic identification algorithms of UC in a way such that the program selects the optimal model for each time series without any human intervention. In addition, the family of models to be selected for each component is wider than usual in the UC literature.

Another important feature is that the library is coded in C++, allowing it to be linked to popular environments, such as R, MATLAB, Octave, Python, etc. At the moment, versions in R, MATLAB and Octave are available in a public repository.

The automatic nature of the identification procedure involves the development of many complementary algorithms to perform many secondary tasks, such as automatic selection of ARMA models, unit root testing, seasonal testing, estimation of constrained stationary and invertible ARMA models, etc.

However, from the user's point of view, handling the library is rather easy, because just a few functions are enough to make an exhaustive analysis of any time series. The core of the paper describes the automatic identification algorithm and shows how its R implementation works on several varied examples.

Acknowledgments

I would like to thank Nerea Urbina for her contribution in building the MATLAB/Octave wrapper functions and manuals. This work was supported by the European Regional Development Fund and Junta de Comunidades de Castilla-La Mancha (JCCM/FEDER, UE) under the project with reference SBPLY/19/180501/000151 and by the Vicerrectorado de Investigación y Política Científica from UCLM through the research group fund program (PREDILAB; [2021-GRIN-31210]).

References

Baxter M, King RG (1999). "Measuring Business Cycles: Approximate Band-Pass Filters for Economic Time Series." *The Review of Economics and Statistics*, **81**(4), 575–593. doi:10.1162/003465399558454.

Beijers D (2022). ***statespacer**: State Space Modelling in R*. R package version 0.4.1, URL https://CRAN.R-project.org/package=statespacer.

Bera AK, Jarque CM (1981). "Efficient Tests for Normality, Homoscedasticity and Serial Independence of Regression Residuals: Monte Carlo Evidence." *Economics Letters*, **7**(4), 313–318. doi:10.1016/0165-1765(81)90035-5.

Bezanson J, Edelman A, Karpinski S, Shah VB (2017). "Julia: A Fresh Approach to Numerical Computing." *SIAM Review*, **59**(1), 65–98. doi:10.1137/141000671.

Bossche FV (2011). "Fitting State Space Models with EViews." *Journal of Statistical Software*, **41**(8), 1–16. doi:10.18637/jss.v041.i08.

Box GEP, Jenkins GM, Reinsel GC, Ljung G (2015). *Time Series Analysis: Forecasting and Control*. John Wiley & Sons.

Casals J, Garcia-Hiernaux A, Jerez M, Sotoca S, Trindade A (2016). *State-Space Methods for Time Series Analysis: Theory, Applications and Software*. Chapman-Hall / CRC Press.

Chowdhury KR (2015). ***rucm**: Implementation of Unobserved Components Model (UCM)*. R package version 0.6, URL https://CRAN.R-project.org/package=rucm.

Clark PK (1987). "The Cyclical Component of U.S. Economic Activity." *The Quarterly Journal of Economics*, **102**(4), 797–814. doi:10.2307/1884282.

Commandeur JJF, Koopman SJ, Ooms M (2011). "Statistical Software for State Space Methods." *Journal of Statistical Software*, **41**(1), 1–18. doi:10.18637/jss.v041.i01.

De Jong P (1991). "The Diffuse Kalman Filter." *The Annals of Statistics*, **19**(2), 1073–1083. doi:10.1214/aos/1176348139.

Doan T (2011). "State Space Methods in RATS." *Journal of Statistical Software*, **41**(9), 1–16. doi:10.18637/jss.v041.i09.

Drukker R, Gates R (2011). "State Space Methods in Stata." *Journal of Statistical Software*, **41**(10), 1–25. doi:10.18637/jss.v041.i10.

Durbin J, Koopman SJ (2012). *Time Series Analysis by State Space Methods*. Oxford University Press.

Eaton JW, Bateman D, Hauberg S, Wehbring R (2022). *GNU Octave Version 7.1.0 Manual: A High-Level Interactive Language for Numerical Computations*. URL https://www.gnu.org/software/octave/doc/v7.1.0/.

Fletcher R (2000). *Practical Methods of Optimization*. John Wiley & Sons.

Flom P (2018). "Stopping Stepwise: Why Stepwise Selection Is Bad and What You Should Use Instead." https://towardsdatascience.com/stopping-stepwise-why-stepwise-selection-is-bad-and-what-you-should-use-instead-90818b3f52df.

Galati G, Hindrayanto I, Koopman SJ, Vlekke M (2016). "Measuring Financial Cycles in a Model-Based Analysis: Empirical Evidence for the United States and the Euro Area." *Economics Letters*, **145**, 83–87. doi:10.1016/j.econlet.2016.05.034.

Gómez V (2015). "**SSMMATLAB**: A Set of MATLAB Programs for the Statistical Analysis of State Space Models." *Journal of Statistical Software*, **66**(9), 1–37. doi:10.18637/jss.v066.i09.

Gómez V, Maravall A (2001). "Automatic Modeling Methods for Univariate Series." In *A Course in Time Series*, pp. 171–201. John Wiley & Sons.

Goodrich RL (2000). "The **Forecast Pro** Methodology." *International Journal of Forecasting*, **16**(4), 533–535. doi:10.1016/s0169-2070(00)00086-8.

Harrison PJ, Stevens CF (1976). "Bayesian Forecasting." *Journal of the Royal Statistical Society B*, **38**(3), 205–228. doi:10.1111/j.2517-6161.1976.tb01586.x.

Harvey AC (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.

Harvey AC, Jaeger A (1993). "Detrending, Stylized Facts and the Business Cycle." *Journal of Applied Econometrics*, **8**(3), 231–247. doi:10.1002/jae.3950080302.

Harvey AC, Koopman SJ (1992). "Diagnostic Checking of Unobserved-Components Time Series Models." *Journal of Business & Economics Statistics*, **10**(4), 377–389. doi:10.1080/07350015.1992.10509913.

Helske J (2017). "**KFAS**: Exponential Family State Space Models in R." *Journal of Statistical Software*, **78**(10), 1–39. doi:10.18637/jss.v078.i10.

Hodrick RJ, Prescott EC (1997). "Postwar U.S. Business Cycles: An Empirical Investigation." *Journal of Money, Credit and Banking*, **29**(1), 1–16. doi:10.2307/2953682.

Hubbard A (2022). ***autostsm**: Automatic Structural Time Series Models*. R package version 3.0.0, URL https://CRAN.R-project.org/package=autostsm.

Hyndman RJ, Khandakar Y (2008). "Automatic Time Series Forecasting: The **forecast** Package for R." *Journal of Statistical Software*, **27**(3), 1–22. doi:10.18637/jss.v027.i03.

Hyndman RJ, Killick R (2022). *CRAN Task View: Time Series Analysis*. Version 2022-05-04, URL https://CRAN.R-project.org/view=TimeSeries.

Hyndman RJ, Koehler AB (2006). "Another Look at Measures of Forecast Accuracy." *International Journal of Forecasting*, **22**(4), 679–688. doi:10.1016/j.ijforecast.2006.03.001.

IBM Corporation (2019). *IBM SPSS Statistics 26*. IBM Corporation, Armonk. URL http://www.ibm.com/software/analytics/spss/.

IHS Markit (2021). *EViews 12 for Windws*. Irvine. URL https://www.EViews.com/.

Koopman SJ, Harvey AC, Doornik J, Shephard N (1999). *Structural Time Series Analysis, Modelling, and Prediction Using **STAMP***. Timberlake Consultants Press, London.

Koopman SJ, Sehphard N (1992). "Exact Score for Time Series Models in State Space Form." *Biometrika*, **79**(4), 823–826. doi:10.1093/biomet/79.4.823.

Koopman SJ, Shephard N, Doornik J (2008). *Statistical Algorithms for Models in State Space Form: **SsfPack** 3.0*. Timberlake Consultants Press.

Leser CEV (1961). "A Simple Method of Trend Construction." *Journal of the Royal Statistical Society B*, **23**(1), 91–107. doi:10.1111/j.2517-6161.1961.tb00393.x.

Ljung GM, Box GEP (1978). "On a Measure of Lack of Fit in Time Series Models." *Biometrika*, **65**(2), 297–303. doi:10.1093/biomet/65.2.297.

López-de-Lacalle J (2016). ***stsm**: Structural Time Series Models*. R package version 1.9, URL https://CRAN.R-project.org/package=stsm.

López-de-Lacalle J (2019). ***tsoutliers**: Detection of Outliers in Time Series*. R package version 0.6-8, URL https://CRAN.R-project.org/package=tsoutliers.

Lucchetti R (2011). "State Space Methods in gretl." *Journal of Statistical Software*, **41**(11), 1–22. doi:10.18637/jss.v041.i11.

Makridakis S, Hibon M (2000). "The M3-Competition: Results, Conclusions and Implications." *International Journal of Forecasting*, **16**, 451–476. doi:10.1016/s0169-2070(00)00057-1.

Monahan JF (1984). "A Note on Enforcing Stationarity in ARMA Models." *Biometrika*, **71**(2), 403–404. doi:10.1093/biomet/71.2.403.

Nocedal J, Wright SJ (2006). *Numerical Optimization*. Springer-Verlag. doi:10.1007/b98874.

Pedregal DJ (2019). "Time Series Analysis and Forecasting with **ECOTOOL**." *PLoS ONE*, **14**(10), 1–23. doi:10.1371/journal.pone.0221238.

Pedregal DJ (2022). ***UComp**: Automatic Unobserved Components Models*. R package version 2.2.3, URL https://CRAN.R-project.org/package=UComp.

Pelagatti M (2015). *Time Series Modelling with Unobserved Components*. Chapman & Hall/CRC.

Peng JY, Aston JAD (2011). "The State Space Models Toolbox for MATLAB." *Journal of Statistical Software*, **41**(6), 1–26. doi:10.18637/jss.v041.i06.

Proietti T (1998). "Seasonal Heteroscedasticity and Trends." *Journal of Forecasting*, **17**(1), 1–17. doi:10.1002/(sici)1099-131x(199801)17:1<1::aid-for675>3.0.co;2-g.

Proietti T, Luati A (2013). "Maximum Likelihood Estimation of Time Series Models: The Kalman Filter and Beyond." In *Handbook of Research Methods and Applications in Empirical Macroeconomics*, pp. 334–362. Edward Elgar Publishing.

R Core Team (2022). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Reilly D (2000). "The **AUTOBOX** System." *International Journal of Forecasting*, **16**(4), 531–533. doi:10.1016/s0169-2070(00)00085-6.

Saavedra R, Bodin G, Souto M (2019). "**StateSpaceModels.jl**: A Julia Package for Time-Series Analysis in a State-Space Framework." *arXiv 1908.01757*, arXiv.org E-Print Archive. doi:10.48550/arXiv.1908.01757.

SAS Institute Inc (2020). *The SAS System, Version 15.2*. SAS Institute Inc., Cary. URL https://www.sas.com/.

Scott SL (2022). ***bsts**: Bayesian Structural Time Series*. R package version 0.9.8, URL https://CRAN.R-project.org/package=bsts.

Seabold S, Perktold J (2010). "**statsmodels**: Econometric and Statistical Modeling with Python." In *9th Python in Science Conference*.

Selukar R (2011). "State Space Modeling Using SAS." *Journal of Statistical Software*, **41**(12), 1–13. doi:10.18637/jss.v041.i12.

StataCorp (2019). *Stata Statistical Software: Release 16*. StataCorp LLC, College Station. URL https://www.stata.com/.

Stoffer D (2022). ***astsa**: Applied Statistical Time Series Analysis*. R package version 1.15, URL https://CRAN.R-project.org/package=astsa.

Taylor CJ, Pedregal DJ, Young PC, Tych W (2007). "Environmental Time Series Analysis and Forecasting with the **Captain** Toolbox." *Environmental Modelling & Software*, **22**(6), 797–814. doi:10.1016/j.envsoft.2006.03.002.

The MathWorks Inc (2021). *MATLAB – The Language of Technical Computing, Version R2021a*. Natick. URL https://www.mathworks.com/products/matlab/.

Van Rossum G, *et al.* (2011). *Python Programming Language*. URL https://www.python.org/.

Villegas MA, Pedregal DJ (2018). "**SSpace**: A Toolbox for State Space Modelling." *Journal of Statistical Software*, **87**(5), 1–26. doi:10.18637/jss.v087.i05.

Young PC, Pedregal DJ, Tych W (1999). "Dynamic Harmonic Regression." *Journal of Forecasting*, **18**(6), 369–394. doi:10.1002/(sici)1099-131x(199911)18:6<369::aid-for748>3.0.co;2-k.

Affiliation:

Diego J. Pedregal
ETSI Industriales
Institute of Applied Mathematics in Science and Engineering (IMACI)
Universidad de Castilla-La Mancha
PREDILAB Research Group
13071 Ciudad Real, Spain
Telephone: +34/(9)26/295430
E-mail: Diego.Pedregal@uclm.es
URL: https://blog.uclm.es/diegopedregal/

Journal of Statistical Software https://www.jstatsoft.org/
published by the Foundation for Open Access Statistics https://www.foastat.org/
August 2022, Volume 103, Issue 9 *Submitted:* 2020-11-03
doi:10.18637/jss.v103.i09 *Accepted:* 2022-03-24