bsm.airp Reference Guide

Javier López-de-Lacalle

January 25, 2013

Abstract

This is a reference guide to the package `bsm.airp` that complements the help files of the package. In this guide we describe the process that replicates the results shown in the paper *101 variations on a maximum likelihood procedure for a structural time series model.*

Contents

1 Introduction

The package bsm.airp replicates the results shown in the paper *101 variations on a maximum likelihood procedure for a structural time series model.* In this document we will sketch the process necessary to run all the variations discussed in the paper.

Two structural models are considered: the basic structural model (BSM) and the local level plus seasonal component model (LLS). See Section 2 of the reference paper for a brief description of these models and to check the notation used in this document.

2 Naming convention

Given a model, a variation on the procedure to obtain maximum likelihood parameter estimates is characterized by six issues that are split in two levels (for convenience to run the computations and store the results):

1. Restrictions on the parameters (if any) and number of steps of the procedure.
2. Kalman filter interface, optimization procedure for the standard or the concentrated likelihood function, parameterization of the model and structure of the covariance matrix of the initial state vector.

The combination of different choices leads to several variations on the maximum likelihood procedure. The results are stored in a comma-separated values (csv) file containing the output from each variation by rows. The model and the choices in the first level of options determine the name of the csv file, while the choices in the second level are used to label each variation within the csv file.

The following labels are used respectively for each option:

1. Restrictions: some of the variance parameters may be fixed to zero.

   `r0`: no restrictions.

   `r1`: the variances of the disturbance term in the observation equation and in the drift term of the trend are set equal to zero ($\sigma^2_\epsilon = \sigma^2_\zeta = 0$).

   `r2`: the variance of the drift is set equal to zero ($\sigma^2_\zeta = 0$).

2. Number of steps of the procedure.

   `s1`: one step procedure. The variance parameters are estimated given an initial state vector, a_0, that is determined beforehand (the first observation of the series and zeros for the remaining elements).

   `s2`: two steps procedure. Given the variance parameter estimates obtained in the previous step, the vector a_0 is estimated.

   `s3`: three steps procedure. Refinement of the variance parameter estimates. Given the estimate of a_0 obtained in the previous step, the variance parameters are reestimated starting from the estimates obtained in the first step.

3. Kalman filter interface. The following interfaces are considered to evaluate the log-likelihood function:[1] `dse`,[2] `dlm`, `FKF`, `KFAS`, `KFAS.d`, `sspir`, `stsm` and `StructTS`. These labels are the name of the packages the interfaces belong to, except for `StructTS` (it is part of the stats package) and `KFAS.d` (it refers to the diffuse version of the filter implemented in the package KFAS).

4. Optimization algorithm and concentration of a parameter.

[1]The package stsm evaluates also the concentrated log-likelihood function.

[2]This interface is not included in the results of the paper as it evaluates a large sample approximation of the likelihood function that is not comparable in the application presented in the paper.

`uo0`: unconstrained optimization (BFGS algorithm available in the function `optim` in package stats).

`co0`: constrained optimization (L-BFGS-B algorithm available in the function `optim` in package stats).

`ab0`: adapted barrier to enforce the constraints (non-negative variance parameters) and Nelder-Mead optimization algorithm (available in the function `constrOptim` in package stats).

If a parameter is concentrated out of the likelihood function the `0` is replaced by `1`. By default, the parameter that is concentrated is σ^2_ϵ (variance of the disturbance in the observation equation) in the unrestricted model and σ^2_ξ (variance of the level) in the restricted models.

5. Parameterization.

   `0`: standard parameterization in terms of variance parameters.

   `1`: regularization. The variance parameters are multiplied by the variance of the data divided by one hundred.

   `2`: the variance parameters are the square of the parameters θ that will be actually estimated by maximum likelihood, $\sigma^2 = \theta^2$.

   `3`: the variance parameters are the exponential of the parameters θ that will be actually estimated by maximum likelihood, $\sigma^2 = \exp(\theta)$.

6. Covariance matrix of the initial state vector.

   `0`: the covariance matrix of the initial state vector is diagonal. It takes on a large value to reflect the uncertainty in a_0 (by default 10 000 times the variance of the data).

   `1`: the initial covariance matrix is non-diagonal (the elements outside the diagonal take on the same value as the diagonal).

The name for the variation is then given by concatenating with dashes the labels for each choices in the order given above. For example, the name `r0-s1-StructS-co0-1-1` is referred to the process implemented in `StructTS` from the core package stats. It stands for an unresticted model, one step procedure, the `StructTS` Kalman filter interface is used along with constrained optimization without concentrating a parameter, the parameters are regularized by a factor and the covariance matrix of the initial state vector is non-diagonal.

3 Scripts

The directory bsm.airp/inst/scripts of the source file of the package contains the scripts that replicate the results of the paper. In this section we will briefly describe the structure and usage of those scripts.

The first step is to load the package and define some parameters to select the computations and how the results are stored. Then, the data and the model are defined.

```
> library("bsm.airp")
> .path.results <- file.path(getwd(), "results")
> .save.rda <- FALSE
> .fvar <- 10000
> .y <- log(AirPassengers)
> .mtype <- "BSM"
```

The results will be stored in the folder results within the current directory. The objects created during the computations related to a single variation can be saved as an rda file. Here we do not use that option. The variance parameters will be saved multiplied by the factor `.fvar`.

The combination of the options introduced in Section 2 leads to several variations. Running the maximum likelihood procedure for a variation requires setting by hand the value of the choices characterizing the variation. Since it is cumbersome for all the cases discussed in the paper, the computations are carried out through a loop around a set of options that are succinctly selected as shown below.

```
> .restricted <- 0
> .steps <- 1
> .KF.version <- c("StructTS")
> .proc <- c("co", "ab")
> .transP <- c("0", "1")
> .P0cov <- c(TRUE, FALSE)
> .clik <- FALSE
```

The scripts will loop around all the objects defined above, which can be of any length. We have selected the variations that use an unrestricted model along with a one step procedure, using the `StructTS` Kalman filter interface. Constrained optimization is selected by means of the L-BFGS-B (`"co"` and AB-NM algorithms (`"ab"` Two different parameterizations are considered in `.transP` according to the notation introduced in Section 2 (standard and regularization). The model will be fit for both a diagonal and a non-diagonal initial covariance matrix. Concentration of a parameter will not be considered in this set of variations. The combination of all these settings leads to $1 \times 1 \times 1 \times 2 \times 2 \times 2 \times 1 = 8$ variations.

The objects above are defined as hidden objects (their names start with a dot). Once the computations related to a variation are finished, the workspace is cleaned for safety. Only hidden objects, which are necessary throughout the entire process, are kept in the workspace over all iterations.

The settings chosen above can be specified through the file params.R in the same directory as the other scripts.

After selecting the options the main script can be run. The script main.R takes care of all the exercise: preparing the data, fitting the models according to the selected variations, performing some diagnostics tests and storing the results.

```
> source("main.R")
```

Here is an abridged version of the main script.

```
> source("params.R")
> for (.kfv in .KF.version)
+ {
+   rm(list = ls(all = FALSE))
+   for (.pr in .proc)
+   {
+     for (.tp in .transP)
+     {
+       if (.tp == "0")
+         .tp  <- NULL
+       for (.p0 in .P0cov)
+       {
+         # build the model
+         pnc <- define.allpars(.restricted, .clik, .mtype)
+         m0 <- stsm.model(model = .mtype, y = .y,
+           pars = pnc$pars, cpar = pnc$cpar, nopars = pnc$nopars,
+           lower = NULL, upper = NULL, transPars = .tp,
+           ssd = FALSE, sgfc = FALSE)
+         KF.args <- list(P0cov = .p0)
+
+         # fit the model
+         source("step1.R")
+         if (.steps > 1)
+         {
+           source("step2.R")
+           if (.steps == 3)
+           {
+             source("step3.R")
+           }
+         }
+
+         # compute smoothed components and diagnostic tests
+         source("diagnostics.R")
+       }
+     }
+   }
+ }
```

The main script consists of nested loops. Given the selected model, the restrictions (if any) and the number of steps of the procedure, the loops go through all the combinations based on the remaining choices: `.kfv`, Kalman filter interface; `.pr`, optimization procedure; `.tp`, parameterization and `.p0`, initial covariance matrix of the state vector.

The computations are carried out within the innermost loop. First, the model is built passing to the function `stsm.model` in package stsm.class the arguments that define the current variaton. Then, depending on the selected number of steps, the scripts containing the code to fit the model are called. Then, the smoothed components are computed for the obtained solution, as well as some diagnostic tests (see original script diagnostics.R): significance of lags of order 1 and 12 in the smoothed components and in the standardized residuals, significance of seasonal dummies in the estimated trend and residuals and homoscdasticity of the smoothed seasonal component.

The models are fit by means of the function `maxlik.td.optim` in package stsm. For example, in the step one procedure, the model is fit as follows.

```
> optmethod <- switch(.pr, "co" = "L-BFGS-B", "uo" = "BFGS", "ab" = "AB-NM")
> fita <- try(maxlik.td.optim(m = m0,
+   KF.version = .kfv, KF.args = KF.args,
+   method = optmethod, gr = "numerical", hessian = FALSE),
+   silent = TRUE)
```

The results are stored in matrix form. The variance parameter estimates and the value of the log-likelihood function for the selections done above are the following.

```
> id <- c("var1", "var2", "var3", "var4", "loglik")
> tab <- .tab.out[order(rownames(.tab.out)),id]
> print(round(tab, 3))
```

```
                  var1   var2 var3   var4  loglik
StructTS-ab0-0-0 1.219  7.044    0  0.658 168.181
StructTS-ab0-0-1 0.000  7.567    0 14.070 162.706
StructTS-ab0-1-0 1.270  6.977    0  0.650 168.183
StructTS-ab0-1-1 0.000  7.695    0 13.999 162.709
StructTS-co0-0-0 0.000 14.921    0  0.000 159.347
StructTS-co0-0-1 0.000 12.368    0 15.510 160.496
StructTS-co0-1-0 1.295  6.994    0  0.642 168.183
StructTS-co0-1-1 0.000  7.718    0 13.969 162.709
```

Although not shown above, the main script stores the results in a csv file and reports the name of the output file on the screen.

```
> cat(paste("The results are stored in:\n",
+   paste(.file.out, ".csv", sep = ""), "\n"))
```

```
The results are stored in:
 BSM-restricted-0-steps-1.csv
```

Before storing the results it is checked whether a previous file with the same name exists in the results directory. If such file exists, the results for the new variations are added to that file ensuring that the results from previous variations are not overwritten. This is convenient since we may not want to run all the possible variations (for a model, restrictions and number of steps) in the same run. In addition, if during the computations the function `check.gonext` (not shown above, see original main file) detects that the results for the current variation are already available in the output file modified in a previous run, then it is skipped. This is also convenient, especially as the number of variations grow, since it helps us to make sure that the computations are done just once.

When the number of variations is relatively large, it is helpful to show the results clustered by groups of variations with common characteristics. For example, below the results are split into groups that share the same value for the options in the first, second and forth positions, (`ord = c(1, 2, 4)`), of the row names in the table of results that was printed before. The positions are separated by dashes and follow the same order in which the options were enumerated in Section 2 (starting from the third one, the Kalman filter interface, since the restrictions and the number of steps is common for all the variations selected here). Thus, passing `ord = c(1, 2, 4)` to `tab.split` split the results into groups that use the same Kalman filter interface, optimization algorithm as well as the same structure of the initial covariance matrix.

```
> tabs <- tab.split(tab, ord = c(1, 2, 4))
> for (i in seq(along = tabs))
+   print(round(tabs[[i]], 3))
```

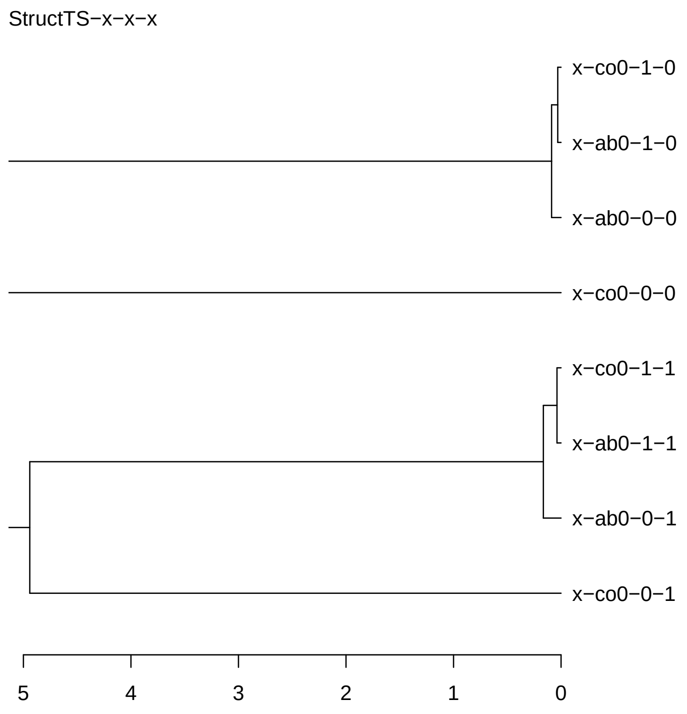
```
                  var1  var2 var3  var4  loglik
StructTS-ab0-0-0 1.219 7.044    0 0.658 168.181
StructTS-ab0-1-0 1.270 6.977    0 0.650 168.183
                 var1  var2 var3   var4  loglik
StructTS-ab0-0-1    0 7.567    0 14.070 162.706
StructTS-ab0-1-1    0 7.695    0 13.999 162.709
                  var1   var2 var3  var4  loglik
StructTS-co0-0-0 0.000 14.921    0 0.000 159.347
StructTS-co0-1-0 1.295  6.994    0 0.642 168.183
```

```
                  var1   var2 var3   var4  loglik
StructTS-co0-0-1     0 12.368    0 15.510 160.496
StructTS-co0-1-1     0  7.718    0 13.969 162.709
```

Comparing the results across groups, we can see that when a non-diagonal covariance matrix for the initial state vector is used, the value of the variance in the seasonal component is relatively larger. Within the third group of solutions as split above, we can see that using the regularization on the parameters helps the L-BFGS-B to converge to a reliable solution when a diagonal initial covariance matrix is used. The second variation in the last group follows the same design as the function `StructTS` in the stats core package. The documentation of `StructTS` claims that this solution is superior to the parameter estimates reported in other source.

The solutions within each group can be plot as a hierarchical cluster. For example, for the whole table (not the groups in which it was split) we obtain the plot shown below. The variance parameter estimates are taken as reference to compute the distance between each solution (`hc.vars = id[1:4]`). The argument `inf.val = 99` indicates that values greater than 99 are taken as infinite (we don't expect so large variance parameters and hence if that value arises it should be located as far as possible from the others).

```
> plothc(tab, ii = 1, hc.vars = id[1:4], inf.val = 99,
+   hc.method = "average", save = FALSE)
```



Notice that, in the plot, the labels of each item takes on an `x` for those elements that are common to all the items (variations). The title of the plot reveals the value of those elements, while an `x` is given to those elements that change across the variations. This labelling is more relevant when we plot the cluster for a group defined in `tab.split`. As in this example those groups are small it is more interesting to display the cluster for all the variations that we considered.

When there are more than one file containing the results, that is, we have run the main scritp for different choices of the restrictions and/or number of steps (given a model BSM oor LLM), it may be necessary to retrieve a selection of results from all the output files. The function `tab.gather.filter` collects the results from all the files found in the directory of results and filters the rows according to the directions passed through the argument `include` as shown below.

```
> tab.gather.filter(path.results = "results",
+   extra.cols = c("loglik"), expand.names = TRUE, fvar = 1,
+   include = list(res = c(0), steps = c(1, 3),
+     kf = c("dlm", "KFAS", "StructTS"),
+     opt = c("co"), trans = c(0, 1), p0 = c(0, 1)))
```

In this case the row names are expanded to identify the value of the restrictions and number of steps, since they are gathered in the same table and a unique label is required. The expanded names for output table shown above are:

```
[1] "r0-s1-StructTS-ab0-0-0" "r0-s3-StructTS-ab0-0-1" "r0-s1-StructTS-ab0-1-0"
[4] "r0-s3-StructTS-ab0-1-1" "r0-s1-StructTS-co0-0-0" "r0-s3-StructTS-co0-0-1"
[7] "r0-s1-StructTS-co0-1-0" "r0-s3-StructTS-co0-1-1"
```

In addition to the variance parameter estimates, extra variables can be retrieved (e.g. `loglik`, `aic`, `bic`). See the output file generated by the main script to identify other variables by columns.

The table returned by `tab.gather.filter` can be split using `tab.split` as shown above. It is also helpful to use the function `summaryPanel`. The arguments of this function are similar to `tab.split`. This function creates a panel of hierarchical plots. It displays a graphical interface to select a plot and to locate and print the results attached to a variation by doing clik on the plot with the mouse.

```
> summaryPanel(tab, ord = c(4, 3, 2), hc.method = "average",
+   inf.val = 99, maxval = 1000, dec = 3, hc.vars)
```

4 Tests

The directory bsm.airp/inst/tests in the source file of the package contains some scripts intended for debugging purposes. The script compare-mll-values.R compares the value of the log-likelihood function obtained with different Kalman filter interfaces. Constant terms that are independent of the parameters of the model are sometimes removed. This code allows us to check whether differences between interfaces are the same, up to a constant. It is a straightforward way to see the expression that is used in each case for the likelihood function (easier than reading the source code since it is not always reported in the documentation).

The reamaining scripts contain sample code about the usage of the packages employed to run the Kalman filter. These interfaces are wrapped in the package KFKSDS. Thus, the code of the main script is simplified using this unified interface. The scripts in this directory are useful for debugging.

5 Results reported in the paper

The directory bsm.airp/inst/article-tables contains the code that prints the tables and summary results presented in the reference paper. Some of the output files follow an older labelling where a 0 or 1 is not added to the label of the optimization procedure to distinguish whether the standard or the concentrated likelihood function was used.