Description of the MS-Regress R package (Rmetrics)

Author: Marcelo Perlin
PhD Student / ICMA – Reading University
Email: marceloperlin@gmail.com / m.perlin@icmacentre.ac.uk

The purpose of this document is to show the general structure of the package. The mathematical notation behind markov switching models and Hamilton's filter is not going to be exposed here since it is not the point. This definitely isn't an introductory document on the markov switching filter. My advice to those who are just starting out is to first read through the papers and books at the references and then come back here to check the application of the algorithm and the general computational structure (input/output) of the package.

Also, be aware that this package is a direct translation of an existing Matlab package. It is available at:

http://www.mathworks.com/matlabcentral/fileexchange/15789

Broadly speaking there are no big differences in both packages but, the Matlab one has a couple of extra functionalities that are not available in the R package (e.g. choice of method for standard error calculation, use of C++ version of Hamilton's filter).

The Mean Equation

This particular R package is quite flexible when it comes to the specification of markov switching models. The central point of this flexibility resides in the input argument S, which controls for where to include markov switching effect.

For instance, if you have 2 explanatory variables (x_{1t} , x_{2t}) and if the input argument S, which is passed to the fitting function MS_Regress_Fit.R, is defined as S<-c(1,1), then the model for the mean equation is:

$$y_t = \beta_1^{S_t} x_{1t} + \beta_2^{S_t} x_{2t} + \sigma^{S_t} \varepsilon_t$$

S_t - represent the state at time *t*, that is, $S_t = 1....K$, where K is the number of states

σ^{S_t} - model's standard deviation at state S_t

$\beta_i^{S_t}$ - beta coefficient for explanatory variable *i* at state S_t where *i* goes from *1* to *n*

ε_t - residue which follows a particular distribution (in this case Normal or Student), with zero mean and std=var=1

Now, changing the input argument to S<-c(0,1), the model is:

$$y_t = \beta_1 x_{1t} + \beta_2^{S_t} x_{2t} + \sigma^{S_t} \varepsilon_t$$

Notes that, with this change in the input argument S, only the second coefficient and the model's standard deviation are switching according to the transition probabilities, that is, the first coefficient (β_1) doesn't change states. Also, you should notes that the order in S is the same order as in the mean equation. For instance, if S<-c(1 0), then the mean equation becomes:

$$y_t = \beta_1^{S_t} x_{1t} + \beta_2 x_{2t} + \sigma^{S_t} \varepsilon_t$$

As an example for the markov switching fitting function, the following script:

```
library(fMarkovSwitching)   # Assuming library is installed

data(indep)          # data from package
data(dep)            # data from package

S=c(1,0,0)           # switching in first indep variable (first column)
distrib<-"Normal"    # Assumed distribution for residue
k<-2                 # number of states

myModel<-MS_Regress_Fit(dep,indep,S,k,distrib)  # calling Fit function
```

reefers to the estimation, based on Gaussian maximum likelihood, of the equations:

$$y_t = \beta_1^{S_t=1} x_{1,t} + \beta_2 x_{2,t} + \beta_3 x_{3,t} + \sigma^{S_t=1} \varepsilon_t \qquad \text{if } S_t = 1$$

$$y_t = \beta_1^{S_t=2} x_{1,t} + \beta_2 x_{2,t} + \beta_3 x_{3,t} + \sigma^{S_t=2} \varepsilon_t \qquad \text{if } S_t = 2$$

with:

$$P = \begin{bmatrix} p_{11} & p_{21} \\ p_{12} & p_{22} \end{bmatrix}$$

as the transition matrix, which controls the probability of a switch from state *j* (column *j*) to state *i* (row *i*). The sum of each column in *P* is equal to one, since they represent full probabilities of the process for each state. Notes that these options to the fitting function are the same provided at the example script Example_MS_Regress_Fit.R (or alternatively with the command *example(MS_Regress_Fit)*).

Another flexibility of the package is that, since I wrote it for dealing with a generic type of regression, you can set any kind of explanatory variable in the model as long as they are observed (you have it available for the whole time period studied). This includes autoregressive components, constants or just plain regression on other variables. This means that you can set up your own econometric model without changing any part of the original code. Make sure you check the folder advScripts in the root directory of fMarkovSwitching.

If you run the script Example_MS_Regress_Fit.R, this is the output (from print()) you should be getting if everything is working. Notes that I've also added some clarifications for the output (in red).

```
***** Numerical Optimization for MS Model Converged *****

Final log Likelihood: 2486.471          (maximum log likelihood found)
Number of parameters: 8                  (number of parameters in the model)
Distribution Assumption -> Normal       (the distribution assumption. You can change it at
                                         advOpt.distrib)

***** Final Parameters *****

---> Non Switching Parameters <---      (the parameters that don't switch (the 0 options at input S)

 Non Switching Parameter at Indep  Column  2
    Value:     0.4797                   (value of non switching coefficient for indep column 2)
    Std error: 0.0269 (0.00)            (standard error and p value for coefficient)
 Non Switching Parameter at Indep  Column  3
    Value:     0.1564                   (value of non switching coefficient for indep column 3)
    Std error: 0.0333 (0.00)            (standard error and p value for coefficient)

--->   Switching Parameters   <---      (the parameters that do switch (the 1 options at input S)

  State 1
     Model Standard Deviation: 0.0288   (the model's standard deviation (the sigma) at state 1)
     Std Error:            0.0014 (0.00)(standard error and p value for corresponding sigma)
  State 2
     Model Standard Deviation: 0.0163   (the model's standard deviation (the sigma) at state 2)
     Std Error:            0.0005 (0.00)(standard error and p value for corresponding sigma)

  Switching Parameters for Indep  Column  1

  State  1
    Value:      0.0006                  (value of the switching variable Indep column 1 at state 1)
    Std error:  0.0017 (0.74)           (associated standard error and p value)
  State  2
    Value:      0.0003                  (value of the switching variable Indep column 1 at state 2)
    Std error:  0.0007 (0.65)           (associated standard error and p value)
```

---> Transition Probabilities Matrix <--- (the matrix that controls the probability of each regime switch, along with standard errors and p-values)[1]

0.98 0.01
0.02 0.99

---> Expected Duration of Regimes <--- the expected duration of each regime, calculated as 1/(1-p_ii))

Expected duration of Regime #1: 50.56 time periods
Expected duration of Regime #2: 94.08 time periods

If this isn't the output you're getting then something is wrong with your copy of R. Fell free to contact me by email, including the error message you get and the code (the main script) you're running.

Using it for your own Data

You probably want to apply the package to you own time series. This topic will set some advices of how you can do this if the model is not converging to a solution.

- Try to make your explained and explanatory series to behave around zero. For instance, if the model is not converging, then try diminishing the mean from the original time series. This will help the optimizing function to find the solution. As you probably know, this is a gentle transformation.

- Always try to estimate simple models. For instance, don't try to estimate any model with $k>3$ and number of explanatory variables higher than 4. The model's size (number of parameters) grows exponentially as n and k grows. For instance, if $k=5$ and $n=4$ (4 explanatory variables), then the model has 55 parameters to be estimated from data, which is definitely too much for Rdonlp2. Don't get me wrong, the package will try to estimate it, but the solution is probably a local maximum and you can't really trust the output you get.

[1] You're probably wondering why no standard errors for the transition probabilities. The reason is that I coded the estimation part of the package as a linear inequality problem. This reduces the problem of finding the higher log likelihood conditional on k*k parameters of the transition matrix to k*(k-1), that is, you solve the optimization problem with less parameters. This means that the values of the transition matrix for the last row are not directly estimated as they are just one minus the sum of the other probabilities in the same column. Given that, those parameters are not included in the Hessian matrix and therefore no standard errors are available. For the transition probabilities outside the last row, the standard errors are available, but I choose not to print them for matter of elegancy

If after those steps, you're still having problems converging to a solution, send a message to my email (**marceloperlin@gmail.com**) with a nice personal introduction and an attached zip file containing:

1) the scripts you're running (the main .R file[2])

2) the error message (if there is one) (could be in .txt or just in the email space)

3) the full dataset (explained and explanatory in .xls or .txt). Hopefully the data is already being loaded to R in your main script

and I'll take a look over it. I'll try to reply in less than 48 hours if the problem is simple (usually is). Also, if you have any question regarding the package, feel free to contact me at my previously cited email.

References

ALEXANDER, C. (2008) 'Market Risk Analysis: Practical Financial Econometrics' Wiley.

HAMILTON, J., D. (1994). Time Series Analysis. Princeton University Press.

HAMILTON, J., D. (2005) 'Regime Switching Models' Palgrave Dictionary of Economic.

KIM, C., J., NELSON, C., R. (1999) 'State Space Model with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications' The MIT press.

[2] The file should run without any serious modification. (I don't mind changing a few things like the working directory or anything simple as that)