MarSwitching.jl: A Julia package for Markov switching dynamic models

Mateusz Dadej [1¶]

1 Phd. student, University of Brescia, Italy ¶ Corresponding author

DOI: 10.21105/joss.06441

Software

- Review
- Repository
- Archive

Editor: Mehmet Hakan Satman

Reviewers:

- @y1my1
- @mkitti

Submitted: 08 February 2024
Published: 14 June 2024

Summary

`MarSwitching.jl` package allows users of the Julia programming language (Bezanson et al., 2017) to efficiently use Markov switching dynamic models. It provides a set of tools for estimation, simulation, and forecasting of Markov switching models. This class of models is the principal tool for modelling time series with regime changes. The time-variation of model parameters is governed by the limited memory Markov process. Given the non-trivial nature of the likelihood function and the amount of model parameters, Julia is an ideal language for implementing this class of models due to its computational performance.

Currently, the package provides model estimation with a combination of switching or non-switching intercept, error variance and exogenous variables. The transition matrix can be either constant or time-varying. The package also provides a set of functions for model diagnostics and forecasting. Further development of the package is considered, conditional on the interest in thereof.

Statement of need

The Markov switching regression (also referred to as regime switching) was first introduced in the seminal work of Hamilton (1989). Since then, it has been extensively used in empirical research. Although the model was introduced as an application to economic data, the range of applications has expanded significantly since the first publication. These fields include finance (Buffington & Elliott, 2002), political science (Brandt et al., 2014), hydrology (Wang et al., 2023), epidemiology (Shiferaw, 2021) and even bibliometrics (Delbianco et al., 2020).

The popularity of these models among applied scientists and industry professionals is reflected in the availability of implementations. There are several packages in R (R Core Team, 2017) such as `MSwM` (Josep A. Sanchez-Espigares, 2021) or `dynr` (Ou et al., 2019). For the Python language, the Markov switching model is implemented as part of the `statsmodels` package (Seabold & Perktold, 2010). MATLAB users may also estimate these models with the `MS_Regress` (Perlin, 2012) package. Most of the well-established closed-source statistical applications also have their own implementations of Markov switching models. These include EViews, Stata, and SAS.

Despite the popularity of the method, `MarSwitching.jl` is, at the moment, the only package that allows for estimation of Markov switching models with the Julia programming language by specifying a minimal set of regime switching parameters. At the same time, it is implemented purely in this language. For more general modeling with hidden Markov models, Julia users may find the `HiddenMarkovModels.jl` (Dalle, 2024) package useful as well. `HiddenMarkovModels.jl` offers a more generic approach to programming hidden Markov models, albeit requiring user-side development of certain estimation algorithms for Markov switching models, as well as model inference functions.

Background

Markov switching models are a class of regression models that allow for time variation of parameters in an otherwise linear model. More specifically, the current state is determined only by the state from the previous period, which is described in the transition matrix.

Consider a general model:

$$\mathbf{y}_t = \mathbf{X}_{t,i}\beta_{S,i} + \epsilon_t$$
$$\epsilon \sim f(0, \Sigma_s)$$

where $\mathbf{y}_t$ is N-vector of the dependent variable indexed by time t, $\mathbf{X}_{t,i}$ is $N \times M$ matrix of exogenous regressors, $\beta_{S,i}$ is K-vector of parameters, and ϵ_t is N-vector of errors. The errors are distributed according to some distribution $f(0, \Sigma_s)$ with zero mean and covariance matrix Σ_s. The state S is a latent (unobservable) variable that can take values from 1 to K. Parameters indexed by S are different for each state.

The state S_t is governed by the Markov process. The probability of transition from state i to state j is given by the $K \times K$ left-stochastic transition matrix $\mathbf{P}$:

$$\mathbf{P} = P(S_t = i | S_{t-1} = j) = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,k} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k,1} & p_{k,2} & \cdots & p_{k,k} \end{pmatrix}$$

with standard constraints: $0 < p_{i,j} < 1, \forall j, i \in \{1, \dots, K\}$ and $\sum_i^K p_{i,j} \forall j \in \{1, \dots, K\}$.

In a standard model, the transition matrix is assumed to be constant over time. However, it is possible to allow for time variation of the transition matrix itself, as described in (Filardo, 1994) (and as implemented in the package). In this case, each of the transition probabilities is modeled as a function of the exogenous variables $\mathbf{Z}_t$:

$$p_{i,j,t} = \frac{\exp(\mathbf{Z}_t \delta_{i,j})}{\sum_{j=1} \exp(\mathbf{Z}_t \delta_{i,j})}$$

where $\delta_{i,j}$ is a vector of coefficients. The exponentiation and sum division of the coefficients ensure that the probabilities are non-negative and sum up to one. For this model, the expected duration of the state is time-varying as well.

Quick start

The package allows for simulation of data from the Markov switching model. The user can specify the number of states, observations, and model parameters (both transition and regression parameters). The package will return a simulated dataset and the standardized exogenous variables.

```julia
using MarSwitching
using Random
import Statistics: quantile

k = 2               # number of regimes
T = 400             # number of generated observations
μ = [1.0, -0.5]     # regime-switching intercepts
β = [-1.5, 0.0]     # regime-switching coefficient for β
```

```
σ = [1.1,  0.8]  # regime-switching standard deviation
P = [0.9 0.05    # transition matrix (left-stochastic)
     0.1 0.95]

Random.seed!(123)

# generate artificial data with given parameters
y, s_t, X = generate_msm(μ, σ, P, T, β = β)
```

The model is estimated using `MSModel()` function. The user needs to specify the dependent variable `y`, the number of states `k`. The exogenous variables are passed to either the `exog_vars` or the `exog_switching_vars` argument, depending wether the variable is expected to have a switching parameter. In a similar vein the user may pass an exogenous variable for the time-varying transition matrix into `exog_tvtp`. However, in order to have an intercept the column of ones needs to be added explicitly.

```
# estimate the model
model = MSModel(y, k, intercept = "switching", exog_switching_vars = X[:,2])
```

Thanks to Julia's multiple dispatch, the `generate_msm()` function works by either providing the parameters as in the first code chunk or using the previously estimated model. This is useful, e.g., for assessing the statistical properties of the model by Monte Carlo simulation.

```
quantile(generate_msm(model, 1000)[1], 0.05)
```

There are several functions for printing statistics of the estimated model. Each of the functions has a `digits` argument specifying a rounding number. `state_coeftable()` shows model coefficients' statistics for a given state and the expected duration of the state. For a standard model with constant transition matrix, the function `transition_mat()` prints a formatted matrix of estimated transition probabilities. For models with time-varying transition probabilities, the coefficients can be inspected with `coeftable_tvtp()`. The function `summary_mars()` prints all the relevant information about the model for each of the states. Additionally, it shows basic information about the model and fitness statistics.

The package also provides functions for filtered transition probabilities $P(S_t = i|\Psi_t)$ and smoothed ones $P(S_t = i|\Psi_T)$ (Kim, 1994). The former is estimated using the data available up to time t, while the latter is estimated using the full sample information in a backward fashion, starting from T, $T-1$, ... and so on. The functions to obtain these probability vectors are `filtered_probs()` and `smoothed_probs()`, respectively.

```
using Plots

plot(filtered_probs(model),
     label     = ["Regime 1" "Regime 2"],
     title     = "Regime probabilities",
     linewidth = 2)
```

Figure Figure 1 presents the output of the code above.

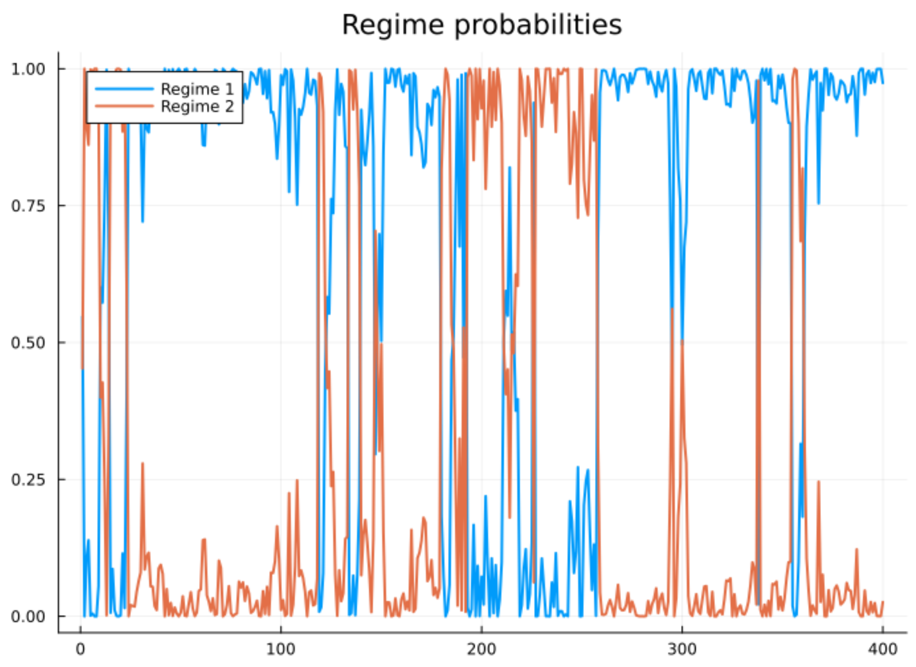

Figure 1: Filtered probabilites.

The package also provides a function for forecasting the dependent variable. However, for the Markov switching models, the prediction is not as intuitive as in less complex models. The reason is that the model also requires a forecast of state at time $t+1$.

`predict()` function returns the forecasted values either calculated in the instantaneous way:

$$\hat{y}_t = \sum_{i=1}^{k} \hat{\xi}_{i,t} X_t' \hat{\beta}_i$$

or as a one step ahead forecast, where the states are predicted themselves:

$$\hat{y}_{t+1} = \sum_{i=1}^{k} (P\hat{\xi}_{i,t}) X_{t+1}' \hat{\beta}_i$$

For more details, the user is referred to the package documentation. Alternatively, in order to inspect the description of a particular function, the help operator - ? in Julia's REPL may come in handy (e.g., `?MSModel`).

Acknowledgements

This open-source research software project received no financial support.

References

Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2017). Julia: A fresh approach to numerical computing. *SIAM Review*, *59*(1), 65–98. https://doi.org/10.1137/141000671

Brandt, P. T., Freeman, J. R., & Schrodt, P. A. (2014). Evaluating forecasts of political conflict dynamics. *International Journal of Forecasting*, *30*(4), 944–962. https://doi.org/10.1016/j.ijforecast.2014.03.014

Buffington, J., & Elliott, R. J. (2002). American options with regime switching. *International Journal of Theoretical and Applied Finance*, *05*(05), 497–514. https://doi.org/10.1142/S0219024902001523

Dalle, G. (2024). HiddenMarkovModels.jl: Generic, fast and reliable state space modeling. *Journal of Open Source Software*, *9*(96), 6436. https://doi.org/10.21105/joss.06436

Delbianco, F., Fioriti, A., Hernandez-Chanto, A., & Tohmé, F. (2020). A Markov-switching approach to the study of citations in academic journals. *Journal of Informetrics*, *14*(4), 101081. https://doi.org/10.1016/j.joi.2020.101081

Filardo, A. J. (1994). Business-cycle phases and their transitional dynamics. *Journal of Business & Economic Statistics*, *12*(3), 299–308. https://doi.org/10.2307/1392086

Hamilton, J. D. (1989). A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, *57*(2), 357–384. https://doi.org/10.2307/1912559

Josep A. Sanchez-Espigares, A. L.-M. (2021). *MSwM: Fitting Markov switching models*. https://doi.org/10.32614/cran.package.mswm

Kim, C.-J. (1994). Dynamic linear models with Markov-switching. *Journal of Econometrics*, *60*(1), 1–22. https://doi.org/10.1016/0304-4076(94)90036-1

Ou, L., Hunter, M. D., & Chow, S.-M. (2019). What's for dynr: A package for linear and nonlinear dynamic modeling in R. *The R Journal*, *11*, 1–20. https://doi.org/10.32614/rj-2019-012

Perlin, M. (2012). *MS_regress - the MATLAB package for Markov regime switching models*. https://doi.org/10.2139/ssrn.1714016

R Core Team. (2017). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Seabold, S., & Perktold, J. (2010). Statsmodels: Econometric and statistical modeling with Python. *9th Python in Science Conference*. https://doi.org/10.25080/majora-92bf1922-011

Shiferaw, Y. A. (2021). Regime shifts in the COVID-19 case fatality rate dynamics: A Markov-switching autoregressive model analysis. *Chaos, Solitons & Fractals: X*, *6*, 100059. https://doi.org/10.1016/j.csfx.2021.100059

Wang, H., Song, S., Zhang, G., & Ayantoboc, O. O. (2023). Predicting daily streamflow with a novel multi-regime switching ARIMA-MS-GARCH model. *Journal of Hydrology: Regional Studies*, *47*, 101374. https://doi.org/10.1016/j.ejrh.2023.101374