Smooth Transition ARMAX Models in Twinkle. (Version 0.9-1

Alexios Ghalanos

May 25, 2014

Contents

1 Introduction

The motivation for creating a package for the modelling of smooth transition ARMA models was the observation that many phenomena appear to go through different states which may be explained by some underlying and observable factors.[1] In financial markets, under different states of the business cycle, financial instruments have been observed to exhibit different characteristics with recessions (or the lead up to such) usually marked by increased volatility and lower or negative mean. Being able to model the evolution of the process driving such changes and hence the switch from one state to another must surely make for a better understanding of the underlying dynamics and perhaps lead to a better forecast model.

The precursor to smooth transition models appears to have been Carmichael (1928) who used the arctangent transformation to capture the change in trends of a number of common stocks, and even considered the possibility of a double transition well before the plethora of papers which came more than 30 years later. The D-method for estimating switching regressions by Quandt (1958) was another early foray into observed switching dynamics which proved quite popular. However, pioneering contributions to the literature on more general threshold autoregression (TAR) must surely have been Tong and Lim (1980) and Tong and Ghaddar (1981), with a more general class of nonlinear AR models introduced in a series of papers by Billings and Voon (1983), Billings and Voon (1986), and Zhu and Billings (1993). Many extensions to the basic TAR model have been considered in the literature , including the threshold moving average model in Gooijer (1998), threshold ARMA processes in Brockwell et al. (1992), smooth transition based on the Gaussian CDF in Chan and Tong (1986), the more widely adopted logistic (LSTAR) and exponential (ESTAR) models discussed in Teräsvirta (1994), nested transition models in Astatkie et al. (1997), multiple states in van Dijk and Franses (1999), and the inclusion of GARCH dynamics in Chan and McAleer(2002, 2003). Thresholds in the variance dynamics have been considered by Zakoian (1994) for GARCH models, and So et al. (2002) for stochastic volatility models. A more general class of hierarchical mixture time series models was proposed by Huerta et al. (2003), while Kalliovirta et al. (2012) recently proposed a Gaussian mixture AR type model. A type of mixture model is also available in the **twinkle** package and discussed in the next section. For a general review of 'recent' extensions and the state of research Dijk et al. (2002) appears to be the most cited paper.

A common theme among the vast majority of econometric research in the area of smooth transition AR models has been the use of the self-exciting model, where the lagged value of the dependent variable (or some simple transformation of the same) is used. In this author's opinion this is not likely to make for a good forecast model since there is only so much information contained in a series own history. The 'real' value is more likely to come from using a set of independent predictor variables either experimentally or better yet using a researcher's insights into the data and process. Unsuprisingly, a large part of the literature focuses on the representation of the model which only considers one switching variable in the state dynamics, perhaps the result of copy-paste propagation. The **twinkle** package departs from this represenation and considers a possibly linear combination of multivariate variables and the use of autoregressive first order dynamics in the state. The m-states model is also reparameterized to use the softmax function to more closely resemble the representation of the multinomial logistic regression model in the way the probabilities are summed and weighted across states. Further extensions include an m-state AR mixture model partially bridging the gap with the finite mixture models, the inclusion of GARCH dynamics, MA dynamics either inside or outside the states, and a large number of conditional distributions which follow from the rugarch package.

As in related packages by the author for econometric modelling, the package includes methods

[1]As opposed to unobservable factors which leads to a different class of models, such as the Markov switching models.

for model specification, estimation, filtering, forecasting and simulation, with useful extractor functions and a number of misspecification tests and inference methods.

It is important for the interested user to be aware from the start that such models are difficult to estimate, may contain local minima and may be generally hard to solve with confidence. While the package has made efforts to provide for a number of solvers and strategies to estimate these models, confident estimation may prove challenging depending on properties of the dataset used and choice of model options. The model is naturally greedy in requiring a substantial amount of data to confidently identify the optimal classification of states given the conditional mean equation. From experience, it is this author's opinion that these types of models may not be as forgiving as linear models when it comes to forecasting, depending on whether the actual forecast state contains the type of nonlinearities under which the model was estimated. Thus, unlike linear models, the misclassification of the forecast state may be more costly. As always, parsimony should be sought in model specification. In-sample fit will rarely translate 1-for-1 to the out-of-sample forecast gains.

This paper is organized as follows: Section 2 discusses the representation of the model in the **twinkle** package and how it can be specified. Section 3 discusses forecasting with a special emphasis on the different methods implemented for n-period ahead forecasts, followed by Section **??** on simulation. Examples may be found in the twinkle.tests folder (under the inst folder) of the source distribution.

While every possible effort has been made to test the model and its methods under different scenarios and squash any bugs, the package is still quite new and further testing is required. The package is available to download from the Bitbucket repository (*alexiosg*) which also contains the development versions of other packages by the author.

General questions on the package should be posted to the R-SIG-FINANCE mailing list, while bugs (with reproducible code and preferably a patch) and suggestions can be reported directly to me.

Finally I would like to acknowledge the valuable help of Eduardo Rossi who collaborated on the new representation and research publication in this area, and participants in the R/Finance 2014 conference where this package was presented and were kind enough to provide valuable feedback.

2 Smooth Transition ARMA Models Revisited

2.1 Dynamics and Extensions

Consider the standard representation of a 2-state STAR model (adapted from van Dijk and Franses (1999))

$$\begin{aligned}
&y_t = {\phi'}_1 y_t^{(p)}\left(F\left(z_{t-d};\gamma,\alpha,c\right)\right) + {\phi'}_2 y_t^{(p)}\left(1-F\left(z_{t-d};\gamma,\alpha,c\right)\right)+\varepsilon_t\\
&y_t^{(p)} = \left(1,\tilde{y}_t^{(p)}\right)', \tilde{y}_t^{(p)} = \left(y_{t-1},\ldots,y_{t-p}\right)'\\
&\phi_i = \left(\phi_{i0},\phi_{i1},\ldots,\phi_{ip}\right)'\\
&\alpha = \left(\alpha_1,\ldots,\alpha_k\right)'\\
&\varepsilon_t \sim ID\left(0,\sigma\right)\\
&i = 1,2 \quad \text{(states)}
\end{aligned} \tag{1}$$

where ε_t is a white noise zero mean error process with standard deviation σ. The state transition function $F\left(z_t-1;\gamma,\alpha,c\right)$ is a continuous function bounded on the unit interval and usually taken to be the logistic CDF[2] such that:

$$\begin{aligned}\text{(Logistic)}:F\left(z_{t-d};\gamma,\alpha,c\right) &= \left(1+\exp\left\{-\gamma\left(\alpha' z_{t-d}-c\right)\right\}\right)^{-1}, \gamma>0\\ \text{(Exponential)}:F\left(z_{t-d};\gamma,\alpha,c\right) &= \left(1-\exp\left\{-\gamma\left(\alpha' z_{t-d}-c\right)^2\right\}\right), \gamma>0\end{aligned} \tag{2}$$

where $z_{t-d} = (z_{1t-d},\dots,z_{jt-d})', j = 1,\dots,k$ is a vector of k observed variables which are assumed to explain the state transition with delay parameter(s)) d. These can be a set of explanatory variables or the lagged values of y_t in which case the model is called 'self-exciting'. Note that in the notation used no special attention has been paid to defining the delay parameter d for the case that z_t is a set of external explanatory variables which may had a variable lag structure. It is also possible that the variable is time in which case the model can be used to identify breaks in the mean as in Lin and Teräsvirta (1994), or a combination of time and other variables giving rise to the time varying STAR (TVSTAR) model discussed in Lundbergh et al. (2003). As correctly noted by van Dijk and Franses (1999), the vector of parameters α needs to be normalized in some way in order to achieve identification (i.e by setting α_1=1). The parameter γ is then a type of scaling factor which determines the smoothness (or speed) of the transition. As Figure 1 illustrates, the logistic function, which gives rise to the LSTAR model, nests the piecewise linear TAR model as $\gamma \to \infty$, and the linear case as $\gamma \to 0$, with different degrees of smoothness for the state transition between the 2 extremes. On the other hand, the exponential function shown in Figure 2 does not nest the TAR model, though it does nest the linear model between the 2 extremes. The symmetric behavior of the function is sometimes preferred for instance in exchange rate studies because of the perceived symmetric exchange rate adjustment behavior.

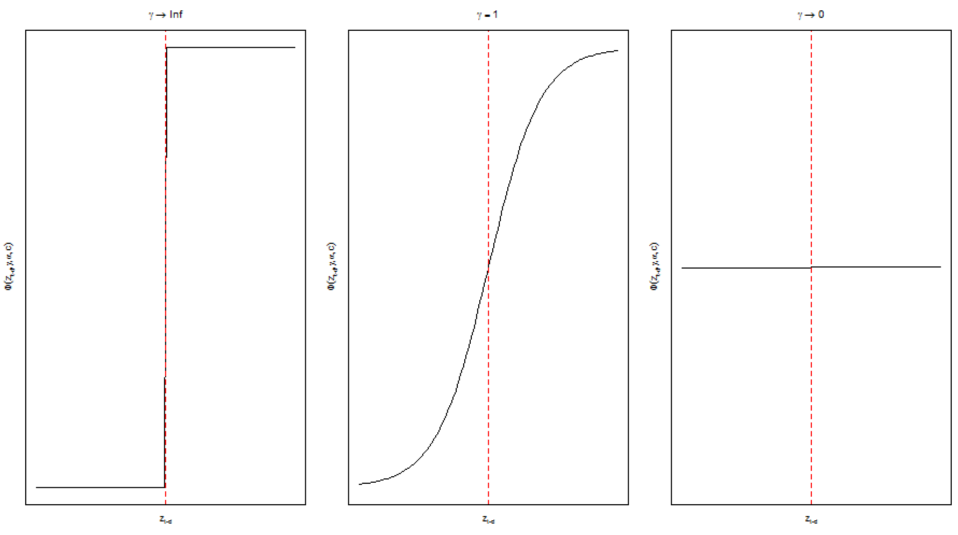

Figure 1: Logistic Transition

By far the most popular test of STAR nonlinearity is described in Luukkonen et al. (1988) using a Taylor series expansion around equation 2, effectively testing whether $\gamma = 0$ (via a auxilliary regression), which would in turn imply that the α vector is also zero and hence a rejection of STAR type non-linearity.

An interesting extension considered in this package is the introduction of autoregressive

[2]At present only the Logistic STAR model is entertained and it is not likely that the exponential STAR model will be considered at all.

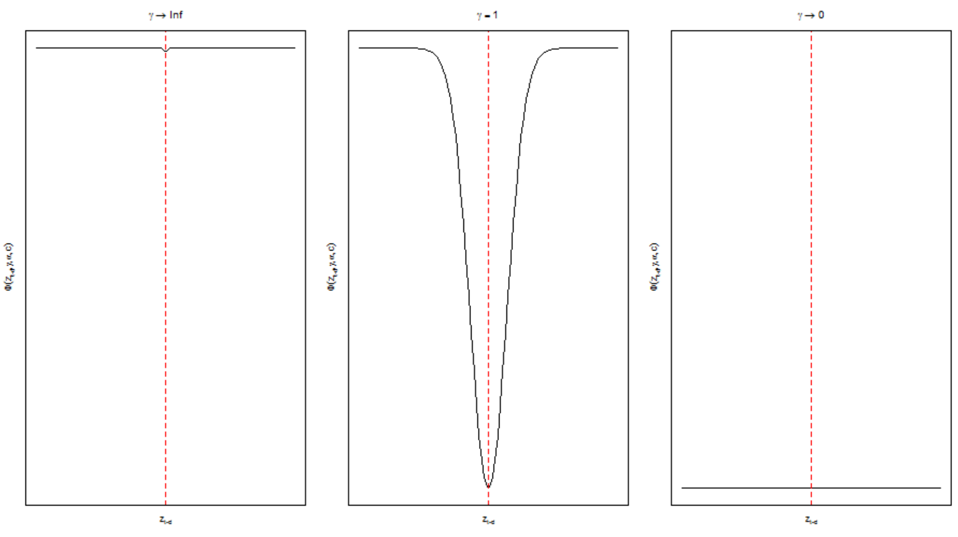

Figure 2: Exponential Transition

dynamics in the state equation as follows:

$$\begin{aligned} &F\left(z_{t-d};\gamma,\alpha,c,\beta\right)=\left(1+\exp\left\{-\pi_t\right\}\right)^{-1}\\ &\pi_t=\gamma\left(\alpha' z_{t-d}-c\right)+\beta'\pi_t^{(q)}\\ &\pi_t^{(q)}=\left(\pi_{t-1},\ldots,\pi_{t-q}\right)' \end{aligned} \tag{3}$$

where the unconstrained state dynamics π_t can be initialized by setting

$$\begin{aligned} \pi_0&=\frac{\gamma\left(\alpha'\bar{z}_{t-d}-c\right)}{1-\beta'\mathbf{1}}\\ \bar{z}&=\left(E\left[z_1\right],\ldots,E\left[z_k\right]\right)' \end{aligned} \tag{4}$$

with a stationarity constraint of $\left|\sum_{i=1}^{q}\beta_i\right|<1$.

This is not just another frivolous extension. Consider that the transition between states is determined by z_{t-d} which usually arrive as shocks, but the actual state we are trying to estimate (e.g. recessions) may be more persistent, in which case the use of AR dynamics may prove usefule. The use of autoregressive dynamics in the state equation follows related work in the area of dynamic binary response models of Kauppi and Saikkonen (2008) and Nyberg (2010). Generally, estimation becomes quite difficult for more than one autoregressive parameter in the state dynamics which is why at present only a lag-1 autoregressive state dynamics model is allowed in the package.

The conditional mean dynamics are not limited to AR terms but may include external regressors (ARX) and moving average (MA) terms in the states giving rise to a full STARMAX model specification:

$$\begin{aligned} y_t&=\left(\phi'_1 y_t^{(p)}+\xi'_1 x_t+\psi'_1 e_t^{(q)}\right)F\left(z_{t-d};\gamma,\alpha,c,\beta\right)\\ &\quad+\left(\phi'_2 y_t^{(p)}+\xi'_2 x_t+\psi'_2 e_t^{(q)}\right)\left(1-F\left(z_{t-d};\gamma,\alpha,c,\beta\right)\right)+\varepsilon_t\\ \varepsilon_t^{(q)}&=\left(\varepsilon_{t-1},\ldots,\varepsilon_{t-q}\right)',\quad \psi'_i=\left(\psi_{i1},\ldots,\psi_{iq}\right)'\\ x_t&=\left(x_1,\ldots,x_l\right)',\quad \xi'_1=\left(\xi_{i1},\ldots,\xi_{il}\right)' \end{aligned} \tag{5}$$

Once again, no special notation is used for the lag structure of the l external regressors x_t, noting only that these must be passed pre-lagged in the package's routines. It is also possible that the MA term enters outside of the states instead of inside giving rise to a STARX with Linear MA terms (STARXLMA):

$$y_t = \left(\phi'_1 y_t^{(p)} + \xi'_1 x_t\right) F\left(z_{t-d}; \gamma, \alpha, c, \beta\right) + \left(\phi'_2 y_t^{(p)} + \xi'_2 x_t\right)\left(1 - F\left(z_{t-d}; \gamma, \alpha, c, \beta\right)\right) + \psi' e_t^{(q)} + \varepsilon_t \tag{6}$$

The STARMAX model therefore encompasses a very wide range of sub-models based on the type of restrictions placed in the conditional mean and state dynamics, and choice of switching variables in the latter.
A natural question which arises from the representation is whether it is reasonable to assume that the conditional variance is the same in both states. Consider the STARMAX 2-state model:

$$\begin{aligned} y_t &= \left(\phi'_1 y_t^{(p)} + \xi'_1 x_t + \psi'_1 e_t^{(q)}\right)\left(F\left(z_{t-d}; \gamma, \alpha, c, \beta\right)\right) \\ &+ \left(\phi'_2 y_t^{(p)} + \xi'_2 x_t + \psi'_2 e_t^{(q)}\right)\left(1 - F\left(z_{t-d}; \gamma, \alpha, c, \beta\right)\right) + \varepsilon_t \\ \varepsilon_t &= y_t - (\mu_{1t}) p_t - (\mu_{2t})(1 - p_t), d > 0 \end{aligned} \tag{7}$$

Add and subtract $y_t p_t$, and re-arrange:

$$\begin{aligned} \varepsilon_t &= +y_t p_t - (\mu_{1t}) p_t + y_t - y_t p_t - (\mu_{2t})(1 - p_t) \\ \varepsilon_t &= +y_t p_t - (\mu_{1t}) p_t + y_t (1 - p_t) - (\mu_{2t})(1 - p_t) \\ \varepsilon_t &= (y_t - \mu_{1t}) p_t + (y_t - \mu_{2t})(1 - p_t) \\ \varepsilon_t &= (\varepsilon_{1,t}) p_t + (\varepsilon_{2,t})(1 - p_t) \\ \varepsilon_{1,t} &\sim N\left(0, \sigma_1^2\right) \qquad \varepsilon_{2,t} \sim N\left(0, \sigma_2^2\right) \\ \varepsilon_t &\sim N\left(0, \sigma_1^2 p_t + \sigma_2^2 (1 - p_t)\right) \end{aligned} \tag{8}$$

Thus, the model can naturally be re-formulated as a mixture of Normals[3] with mixing probabilities derived from the state dynamics. This provides for a more parsimonious and clear extension than using GARCH dynamics on the mixed state residuals. Alternatively, it can be thought of as the time-invariant version of a STARX-STGARCH model with common transition dynamics (and restriction ARCH=GARCH=0).

2.2 Multiple States

van Dijk and Franses (1999) considered an extension of the 2-state STAR model in equation 1 to a 4-state models as follows:

$$\begin{aligned} y_t = &\left[\phi'_1 y_t^{(p)}\left(1 - F\left(z_{t-d}; \gamma_1, \alpha, c\right)\right) + \phi'_2 y_t^{(p)}\left(1 - F\left(z_{t-d}; \gamma_1, \alpha, c\right)\right)\right]\left(1 - F\left(z_{t-d}; \gamma_2, b, d\right)\right) \\ + &\left[\phi'_3 y_t^{(p)}\left(1 - F\left(z_{t-d}; \gamma_1, \alpha, c\right)\right) + \phi'_4 y_t^{(p)}\left(1 - F\left(z_{t-d}; \gamma_1, \alpha, c\right)\right)\right] F\left(z_{t-d}; \gamma_2, b, d\right) + \varepsilon_t \end{aligned} \tag{9}$$

This effectively models 2 unique states and one interaction state:

$$\begin{aligned} \mu_1 &= \phi'_1 y_t^{(p)}\left(1 - F\left(z_{t-d}; \gamma_1, \alpha, c\right) - F\left(z_{t-d}; \gamma_2, b, d\right) + F\left(z_{t-d}; \gamma_1, \alpha, c\right) F\left(z_{t-d}; \gamma_2, b, d\right)\right) \\ \mu_2 &= \phi'_2 y_t^{(p)}\left(1 - F\left(z_{t-d}; \gamma_1, \alpha, c\right) - F\left(z_{t-d}; \gamma_2, b, d\right) + F\left(z_{t-d}; \gamma_1, \alpha, c\right) F\left(z_{t-d}; \gamma_2, b, d\right)\right) \\ \mu_3 &= \phi'_3 y_t^{(p)}\left(F\left(z_{t-d}; \gamma_2, b, d\right) - F\left(z_{t-d}; \gamma_1, \alpha, c\right) F\left(z_{t-d}; \gamma_2, b, d\right)\right) \\ \mu_4 &= \phi'_4 y_t^{(p)}\left(F\left(z_{t-d}; \gamma_2, b, d\right) - F\left(z_{t-d}; \gamma_1, \alpha, c\right) F\left(z_{t-d}; \gamma_2, b, d\right)\right) \end{aligned} \tag{10}$$

[3]Or any other location-scale invariant distribution

Alternatively, the implementation followed in this package takes a page out of multinomial regression and models multiple states using the softmax function:

$$y_t = \sum_{i=1}^{s} \left[\left({\phi'}_i y_t^{(p)} + {\xi'}_i x_t + {\psi'}_i e_t^{(q)} \right) F_i \left(z_{t-d}; \gamma_i, \alpha_i, c_i, \beta_i \right) \right] + \varepsilon_{i,t} \tag{11}$$

with s-1 distinct states modelled:

$$\begin{aligned} F_i \left(z_{t-d}; \gamma_i, \alpha_i, c_i, \beta_i \right) &= \frac{e^{\pi_{i,t}}}{1 + \sum\limits_{i=1}^{s-1} e^{\pi_{i,t}}} \\ F_s \left(z_{t-d}; \gamma_i, \alpha_i, c_i, \beta_i \right) &= \frac{1}{1 + \sum\limits_{i=1}^{s-1} e^{\pi_{i,t}}} \end{aligned} \tag{12}$$

where the s states are weighted to sum to unity. This appears, at least to this author, to be a more natural representation for a multi-state setup. In the **twinkle** package, upto 4 states may be modelled[4]

2.3 Estimation

Estimation of the STARMAX models is done by maximizing the likelihood without imposing any particular inequality restrictions on the state dynamic intercepts or any parameter bound restrictions (except for positivity bounds on the variance and γ state scaling parameter).[5] Since unconstrained optimizers appear to do quite well for hard nonlinear/non-smooth problems, the main solver in the twinkle package is the BFGS solver from the optim function. It is possible to include parameter bounds in which case a logistic transformation is used with the unconstrained solvers. Additional solvers included are 'nlminb' (bound constrained), 'solnp' (nonlinear SQP solver with nonlinear constraints), 'cmaes' (bound constrained global solver) and 'deoptim' (bound constrained global solver). However, it is suggested that either a multi-start strategy is followed (by choosing 'msoptim') or an iterative search strategy ('strategy') which cycles between fixing the state parameters to estimate the conditional mean parameters (linear), fixing the conditional mean parameters to estimate the state parameters (nonlinear) and a random start estimation. As with general nonlinear optimization problems, scaling of the data prior to estimation may help, an ica or pca transformation if they are highly correlated, or hinge basis transformation of the dataset via the earth package is another interesting option in the case of relevant feature extraction.

3 Forecasting

Consider a general nonlinear first order autoregressive model:

$$y_t = F\left(y_{t-1}; \theta \right) + \varepsilon_t \tag{13}$$

where $F\left(y_{t-1}; \theta \right)$ is some nonlinear function mapping y_{t-1} to y_t given the parameter set θ. The optimal h-step ahead point forecast, using a least squares criterion, of y_{t+h} at time t is given by:

$$\hat{y}_{t+h|t} = E\left[y_{t+h} \left| \Im_t \right. \right] \tag{14}$$

[4]For the case of only a single state, it is also possible to pass a set of 'time' weights.

[5]Constraints on the autoregressive and moving average parameters may also be placed to impose some type of stationarity constraint per state.

where $\Im_t$ is the information set upto time t. Given that $E\left[\varepsilon_{t+1}\left|\Im_t\right.\right] = 0$, then the 1-step-ahead optimal forecast is:

$$\hat{y}_{t+1|t} = E\left[y_{t+1}\left|\Im_t\right.\right] = F\left(y_t;\theta\right) \tag{15}$$

which is the same as when $F(.)$ is linear. However, for horizons greater than 1, this is not the case since $E\left[F\left(.\right)\right] \neq F\left(E\left[.\right]\right)$, which means that simple recursive relationship found in the linear case do not exist in the nonlinear case. Instead, consider the h-step-ahead point forecast using the following closed form representation:[6]

$$E\left[y_{t+h}\left|\Im_t\right.\right] = \int_{-\infty}^{\infty} E\left[y_{t+h}\left|y_{t+h-1}\right.\right] g\left(y_{t+h-1}\left|\Im_t\right.\right) dy_{t+h-1} \tag{17}$$

where $g\left(y_{t+h}\left|\Im_t\right.\right) = f\left(y_{t+h} - F\left(y_{t+h-1};\theta\right)\right)$, is the distribution of the shock ε_{t+h} with mean $F\left(y_{t+h-1}\right)$, though the distribution ε_t is never known with certainty. A number of approaches have been used in the literature to estimate this integral. It is simple to see that the conditional distribution of $g\left(y_{t+h-1}\left|\Im_t\right.\right)$ can be obtained recursively starting at h=2 and noting that $g\left(y_{t+1}\left|\Im_t\right.\right) = f\left(y_{t+1} - F\left(y_t;\theta\right)\right)$. To obtain the forecasts, numerical integration can be used (applied recursively) or monte carlo methods. In the former case, the form of the conditional distribution $f\left(.\left|\Im_t\right.\right)$ can be replaced by a kernel estimator, whereas in the latter case one has an option of using an empirical bootstrap, simulating from the conditional distribution $f\left(.\left|\Im_t\right.\right)$ or a kernel estimator. For instance, the 2-step ahead monte carlo forecast is given by:

$$\hat{y}_{t+2|t} = \frac{1}{T}\sum_{i=1}^{T} F\left(\hat{y}_{t+1|t} + \varepsilon_i;\theta\right) \tag{18}$$

However, in the case when a GARCH model is used for the modelling of the conditional variance, then the monte carlo forecast needs to be adjusted as follows:

$$\hat{y}_{t+2|t} = \frac{1}{T}\sum_{i=1}^{T} F\left(\hat{y}_{t+1|t} + z_i\hat{\sigma}_{t+2|t};\theta\right) \tag{19}$$

where z_i represent draws from either the parametric standardized distribution of the model or the standardized in-sample innovations (or draws from a kernel estimated density of the standardized in-sample innovations), which are then multiplied by the forecast GARCH volatility $\hat{\sigma}_{t+2|t}$ to obtain the forecast residuals ε_i. One benefit of using a monte-carlo or bootstrap approach is that they immediately give rise to the density of each point forecast thus allowing for the creation of interval forecasts.

[6]This is based on the Chapman-Kolmogorov relation:

$$g\left(y_{t+h}\left|\Im_t\right.\right) = \int_{-\infty}^{\infty} g\left(y_{t+h}\left|y_{t+h-1}\right.\right) g\left(y_{t+h-1}\left|\Im_t\right.\right) dy_{t+h-1} \tag{16}$$

which leads to Equation 17 after taking conditional expectations from both sides.

4 Software Implementation

4.1 Specification

The entry point to defining and estimating a STARMAX model in the twinkle package is the starspec function:

```
>starspec

function(
mean.model = list(states=2, include.intercept=c(1,1), arOrder=c(1,1),
        maOrder=c(0, 0), matype="linear", statevar=c("y","s"), s=NULL,
        statear=FALSE, ylags=1, xreg=NULL, yfun=NULL, transform="log"),
variance.model=list(dynamic=FALSE, model="sGARCH", garchOrder=c(1,1),
        submodel=NULL, vreg=NULL, variance.targeting=FALSE),
distribution.model="norm", start.pars=list(), fixed.pars=list(),
fixed.prob=NULL, ...)
```

The **mean.model** defines the equation for the conditional mean dynamics including the state dynamics. Upto 4 **states** are allowed, with the 1-state option having a special implementation in that the **fixed.probs** list is an xts matrix (alined to the dataset which will be passed to the estimation routine) of weights. By default this is set to a vector of ones in this case but may be any other 'time-weighting' scheme the user wishes. The options for **intercept**, **arOrder** and **maOrder** should be integer vectors of length equal to the number of **states**.

The **matype** denotes whether the moving average terms enter inside the states ('states') or outside ('linear').

The **statevar** indicates whether the model will switch based on its own value ('y') or an exogenouse set of regressors ('s'), in which case an xts matrix (aligned to the index of the dataset and appropriately lagged) is passed to **s**. In the case that **statevar** is 'y', then **ylags** is an integer vector of the unique lags to use as a linear combination.

The **yfun** option allows the user to pass a function to transform the value of y[7] prior to being used in the state dynamics equation. While it may appear at first that the same can be achieved by passing a pre-transformed value and using 's' as the **statevar**, consider that simulation and n-ahead forecasts (which depend on simulation methods) on transformed values of 'y' can then be used directly, where it would have been impossible to do so otherwise (because of the path dependency).

The **xreg** is an optional xts matrix of external regressors which needs to be aligned to the index of the dataset (and appropriately lagged). The **statear** indicates whether to include lag-1 autoregression in the state dynamics as discussed previously in equation 3.

Tthe **transform** is currently fixed to use only the logistic transformation, and there are no plans to extend to the exponential at present.

The variance model can be **dynamic**, in which case a choice of 'mixture','sGARCH','gjrGARCH' and 'eGARCH' are implemented, else by default a static variance model is used. For the GARCH flavors, the rest of the options follow from the rugarch package, whilst the 'mixture' model is based on equation 8.

All distributions implemented in rugarch are included as options in **distribution.model**, while fixed and starting parameters can be passed directly via **fixed.pars** and **start.pars** respectively, else later on via the **setfixed<-** and **setpars<-** methods on the star specification (note that there is also a **setbounds<-** methods for setting and enforcing parameter bounds).

[7]The function must return the same length as the value it receives without any NAs or NaNs.

Finally, the **fixed.probs** list allows the user to pass an xts matrix of fixed probabilities for each state (aligned to the index of the dataset and with columns equal to **states**). This could for instance be the forecast probabilities from another model (e.g. logistic regression) representing market up and down periods, recessions etc. In this case the state equation is not used and the model is effectively linear and extremely fast to estimate for the conditional mean dynamics.

The returned specification object is of class **STARspec** which may be passed to the estimation routine **starfit**. If the object has been assigned fixed parameters for the complete parameter set, then it may instead be passed to the **starfilter**, **starforecast** or the **starpath** routines.

4.2 Estimation

Once the model has been specified, it may be estimated by maximum likelihood using the **starfit** routine:

```
>starfit

(spec, data, out.sample=0, solver="optim", solver.control=list(),
fit.control=list(stationarity=0, fixed.se=0, rec.init="all"),
cluster=NULL, n=25, ...)
```

The **data** must be an xts object with the same time indices as any data already passed to the **STARspec** object and contain only numeric data without any missing values. The **out.sample** is used to indicate how many data points to optionally leave out in the estimation (from the end of the dataset) for use in out-of-sample forecasting later on when the estimated object is passed to the **starfilter** routine. Perhaps the most important choice to be made is the type of **solver** to use and it's control parameters (**solver.control**). The following solvers and 'strategies' are included:

- optim. The preferred choice is the BFGS solver. The choice of solver is controll by the *method* option in the **solver.control** list.
- nlminb. Have had little luck getting the same performance as the BFGS solver.
- solnp. Will most likely find a local solution.
- cmaes. Even though it is a global solver, it requires careful tweaking of the control parameters (and there are many). This is the parma package version of the solver.
- deoptim. Another global solver. May be slow and require tweaking of the control parameters.
- msoptim. A multistart version of optim with option for using the **cluster** option for parallel evaluation. The number of multi-starts is controlled by the *n.restarts* option in the **solver.control** list.
- strategy. A special purpose optimization strategy for STAR problems using the BFGS solver. It cycles between keeping the state variables fixed and estimating the linear variables (conditional mean, variance and any distribution parameters), keeping the linear variables fixed and estimating the state variables, and a random re-start optimization to control for possibly local solutions. The argument **n** in the routine controls the number of times to cycle through this strategy. The **solver.control** list should pass control arguments for the BFGS solver. This is somewhat related to concentrating the sum of squares methodology in terms of the estimation strategy, but does not minimize the sum of squares.

The *strategy* and *msoptim* solver strategies should be the preferred options when estimating STARMA models.

The resulting object of class **STARfit** has a number of methods including an S4 summary (*show*) and a number of S4 extractor methods such as **coef**, **likelihood**, **vcov**, **infocriteria**, **modelmatrix**, **quantile**, **pit**, **fitted**, **residuals** and **sigma**. A new method **states** can be used for extracting the conditional state probabilities, with an extra argument **type** with options for 'prob' (probabilties), 'condm' (conditional mean dynamics per state) and anything else will return the untransformed (raw) state dynamics. The methods are documented in the help page of **STARfit-class**. A default **plot** method is also available which plots the states and fitted values of the model. Finally, note that unlike other implementations which start the iteration (and summation of the likelihood) of the model at the maximum of the AR or MA lags, the **twinkle** package starts at T=1 and builds up the likelihood by using all information available at that point (e.g. for T=1, if an intercept is included then that is used to obtain the residuals, then at T=2, the AR1 parameter is used and so on). For the **nonlinearTest** in section 5 the lags are first substracted before carrying out the test in order to be consistent with published results.

4.3 Filtering

An object of class **STARspec** with pre-assigned fixed parameter values (for the complete model parameter set) may be passed to the **starfilter** routine with a new or augmented dataset (to that which was used to estimate the original parameters). In this way, new data can be filtered using existing parameters which is equivalent to performing rolling 1-step ahead forecasts (without re-estimation).

```
>starfilter

(spec, data, out.sample=0, n.old=NULL, rec.init="all", ...)
```

It is probably best to provide an augmented dataset for filtering since the model may depend on the complete history (particularly as regards use of the autoregressive parameter in the state dynamics), in which case the **n.old** option should also be used to denote the size of the original dataset.

4.4 Forecasting

Forecasting can be carried out either from an estimated object of class **STARfit** or a specification with fixed parameters of class **STARspec** (in which case the **data** argument must also be used[8]).

```
>starforecast

function(fitORspec, data=NULL, n.ahead=1, n.roll=0, out.sample=0,
external.forecasts=list(xregfor=NULL, vregfor=NULL, sfor=NULL,
probfor=NULL), method=c("an.parametric", "an.kernel", "mc.empirical",
"mc.parametric", "mc.kernel"), mc.sims=NULL, ...)
```

As discussed in Section 3, for n-step (n>1) ahead forecasts, there are a number of options available based on recursive quadrature integration ('an') of the integral in Equation 17 else monte carlo ('mc') integration. The *parametric* method uses the density from the estimated object while the *kernel* method fits a kernel density to the residuals. Finally, and only available for the monte carlo method, the *empirical* option samples from the empirical distribution of

[8]Effectively, the data is filtered with the fixed parameter specification prior to the forecast being carried out

the residuals. Clearly with a limited history it may be optimal to use either the *parametric* or *kernel* methods. For the monte carlo integration, the **mc.sims** argument denotes the number of samples to use per period.

Some care should be taken when passing **external.forecasts** for the conditional mean regressors (xregfor), the conditional variance regressors (vregfor), the conditional state dynamics regressors (sfor) and the conditional probability (probfor) in the case that the state probabilities where passed as fixed in the specification. These xts matrices should be pre-lagged in the same way as the input matrices where in the specification.

The resulting object is of class **STARforecast** with a number of extractor functions documented in the help page of the class and similar to those available for the **STARfit** class. Of particular interest in the case of monte carlo integration is the estimated density of each point forecast which may be extracted from the object.

4.5 Simulation

Simulation can be carried out either directly on a **STARfit** object using the **starsim** routine, else on a **STARspec** object with fixed parameters using the **starpath** method.

```
>starsim

function(fit, n.sim=1000, n.start=0, m.sim=1, presigma=NA, prereturns=NA,
preresiduals=NA, rseed=NA, custom.dist=list(name=NA, distfit=NA),
xregsim=NULL, vregsim=NULL, ssim=NULL, probsim=NULL, ...)
```

The arguments follow similar convention as in related packages. In particular, the **ssim** argument should be a list of matrices for the simulated values of the external regressors (**s**) in the state dynamics, while **probsim** should be a list of matrices of the simulated state probabilities in the case that fixed probabilties were used in the original specification.
Similarly, the **starpath** routine has the following arguments:

```
>starpath

function(spec, n.sim=1000, n.start=0, m.sim=1, presigma=NA, prereturns=NA,
preresiduals=NA, rseed=NA, custom.dist=list(name=NA, distfit=NA),
xregsim=NULL, vregsim=NULL, ssim=NULL, probsim=NULL, ...)
```

Where the **prereturns** are now required as depending on the model, so is **presigma** and **preresiduals**. The length of these initialization matrices is determined by the maximum of the conditional state, mean and variance dynamic lags.

5 Mispecification Test

5.1 STAR type nonlinearity

The key test of STAR type nonlinearity is that of Luukkonen et al. (1988) which is implemented in the **nonlinearTest** method. This effectively tests the null hypothesis that the model is linear against STAR type nonlinearity. Because of the presence of unidentified[9] STAR model parameters under the null, the test replaces the logistic transition function with a third order Taylor series approximation around $\gamma = 0$ leading to the following equation:

$$y_t = \text{const} + \mathbf{X}_t\beta_0 + \mathbf{X}_t z_t \beta_1 + \mathbf{X}_t z_t^2 \beta_1 + \mathbf{X}_t z_t^3 \beta_1 + \varepsilon_t \qquad (20)$$

[9]Under the null, γ is zero and hence the transition probabilities collapse to 0.5, in which case c and *alpha* (and *beta* if that is used) are not identified.

where $\mathbf{X}_t = \left(\tilde{y}_t^{(p)}, \tilde{x}_t\right)'$ is the vector of lagged series and external variables as in equation 5, and z_t is the state transition variable. In the case that this is a vector then it is not clear what one should do since only under a single variable does the identification restriction allow the test to be carried out without the model having to be estimated. In the package this case is dealt with by multiplying the z_t vector of k transition variables by the estimated coefficient vector $\alpha = (1, \alpha_2, \ldots, \alpha_k)$, which means that the model needs to be estimated first. Additionally, in the case that this vector is made up of external variables rather than a lagged vector of the series (y_t), then a constant is added to the $\mathbf{X}$ vector in the cases that it is multiplied by z_t so that $\mathbf{X}_t = \left(1, \tilde{y}_t^{(p)}, \tilde{x}_t\right)'$. This is not possible in the case of z_t being the lagged series of y_t since a constant would induce almost perfect multicollinearity. The test can be run on either an estimated object of class **STARfit** or a specification object of class **STARspec** in which case a data object of class **xts** is also required. The test returns both the χ^2 and F distribution statistics and p-values, and there is also an option for computing *robust* (to heteroskedasticity) statistics based on the procedure described in Granger and Terasvirta (1993).

6 Examples

In this section, a small number of examples are considered in order to illustrate the working of the package. A detailed testing suite is available in the src distribution under the **inst** folder.

6.1 The Dutch Gilder Benchmark (Franses and van Dijk (2000))

The package contains the **forex** dataset which is an xts matrix consisting of the daily value of 8 currency pairs spanning the period 31-12-1979 to 31-12-1998 from the book of Franses and van Dijk (2000), and originally sourced from the Federal Reserve Bank of New York. In Chapter 3 of their book, they provide a model for the Dutch Gilder weekly (Wednesday) series using a STAR model with switching based on the 4-week moving average of the lagged absolute value of this return series. In this example we replicate their results, making use of the custom transformation function *yfun*.

```
require(quantmod)
data(forex)
# to replicate the results of van Dijk and Franses use:
# next observation carried backward (i.e. fromLast=TRUE)
# for missing values
fx = na.locf(forex, fromLast = TRUE)
fxw = fx[which(weekdays(index(forex))=="Wednesday"),4]
dx = ROC(fxw, na.pad=FALSE)*100
# threshold variable a running mean of the absolute returns
# period considered
idx = 1:521
fun = function(x){
x = as.numeric(x)
y = runMean(abs(x), n=4)
y[1:3] = c(abs(x[1]), mean(abs(x[1:2])), mean(abs(x[1:3])))
return(y)
}
spec = starspec(mean.model = list(states=2, include.intercept=c(1,1),
arOrder=c(0,2), statevar="y",ylags=1,yfun=fun))
control=list(alpha=1, beta=0.4, gamma=1.2, reltol=1e-12, trace=1,
method="BFGS",n.restarts=3)
fit = starfit(spec, data=dx[idx], solver="msoptim", solver.control=control)
twinklepars = coef(fit)
vdfpars = c(-0.18019807, 0.05980984, 0.28705073, 0.21335697, 4.3161180, 1.3554227)
vdfse   = c( 0.13140430, 0.10138187, 0.10719107, 0.10026741, 1.0771948, 0.1521352)
names(vdfpars) = names(vdfse) = c("s1.phi0", "s2.phi0", "s2.phi1","s2.phi2",
"s1.gamma", "s1.c") twinklepars = coef(fit1)
twinklese = sqrt(diag(vcov(fit, robust=FALSE)))
twinkleSSE = sum(residuals(fit)^2)
vdfSSE = 1233.321352

>print(data.frame(vdfpars = vdfpars, twinklepars = twinklepars[1:6]))
             vdfpars twinklepars
s1.phi0  -0.18019807 -0.17894764
s2.phi0   0.05980984  0.07028044
s2.phi1   0.28705073  0.29665768
s2.phi2   0.21335697  0.21808028
s1.gamma  4.31611800  7.96823137
s1.c      1.35542270  1.34296993

>print(data.frame(vdfse = vdfse, twinklese = twinklese[1:6]))
             vdfse  twinklese
s1.phi0  0.1314043 0.14132956
s2.phi0  0.1013819 0.09597175
s2.phi1  0.1071911 0.11585311
s2.phi2  0.1002674 0.10842124
s1.gamma 1.0771948 5.54362697
s1.c     0.1521352 0.19176212

>show(fit)
*---------------------------------*
*          STAR Model Fit         *
*---------------------------------*
states       : 2
statevar     : y
statear      : FALSE
variance     : static
distribution : norm


Optimal Parameters (Robust Standard Errors)
------------------------------------
           Estimate  Std. Error  t value Pr(>|t|)
s1.phi0    -0.17895    0.192527 -0.92947 0.352647
s2.phi0     0.07028    0.082826  0.84854 0.396140
s2.phi1     0.29666    0.119232  2.48806 0.012844
s2.phi2     0.21808    0.119800  1.82037 0.068703
s1.gamma    7.96823    3.967261  2.00850 0.044590
s1.c        1.34297    0.260680  5.15180 0.000000
s1.alpha1   1.00000          NA       NA       NA
```

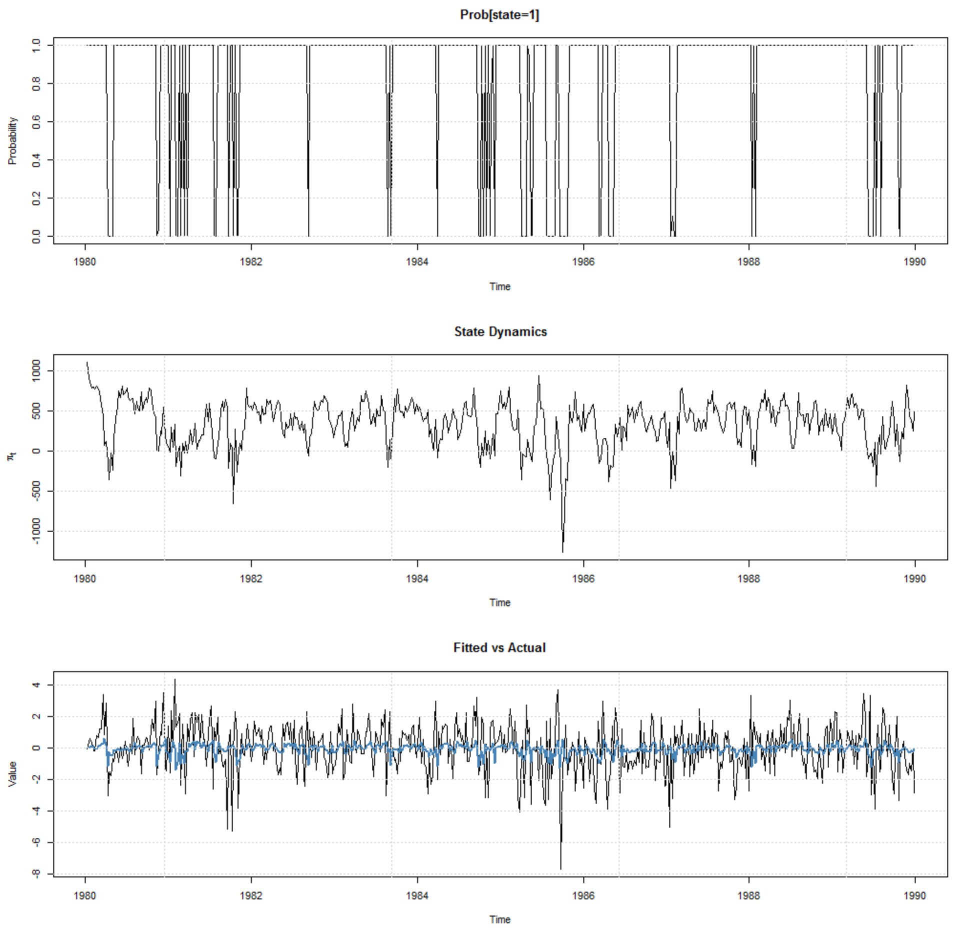

Figure 3: Dutch Gilder STAR Model

```
sigma        1.54011    0.057506 26.78179 0.000000

LogLikelihood : -964.2643

Akaike       3.7285
Bayes        3.7856
Shibata      3.7281
Hannan-Quinn 3.7509

r.squared         :  0.0418
r.squared (adj)   :  0.0287
RSS               :  1235.787
skewness (res)    :  -0.47002
ex.kurtosis (res) :  0.88495

AR roots
         Moduli1 Moduli2
state_1       NA      NA
state_2 1.566636 2.92695
```

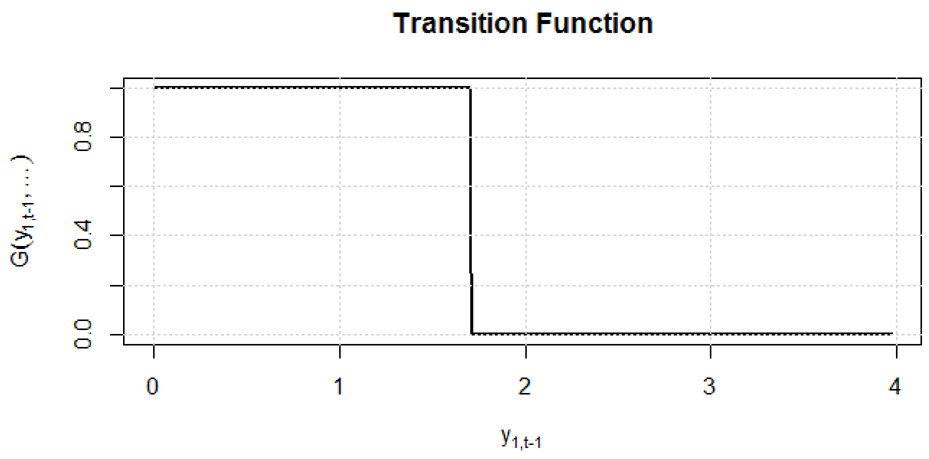

Figure 4: Dutch Gilder Transition

There is also a default plot method which creates 3 panels to show the estimated state probabilities, raw state dynamics and fitted versus actual as shown in figure 3. There is also a transition probability plot *trans2fun2d* as shown in figure 4. The threshold value at which state switching occurs is about 1.8. Given the scaling of the original input data, and the coefficient summary, this means that when average 4 week absolute returns are above 1.8% the states switch from a high (positive) to a low (negative) state.

6.2 Nasdaq 100 Index Benchmark

The package has a second dataset of the OHLCV of the Nasdaq 100 index spanning the period *02-01-1996* to *12-10-2001* from Yahoo finance. This is the same dataset used in a number of examples in the excellent book of Zivot and Wang (2007). The example here is taken from chapter 18 of that book based on a 'realized' weekly variance measure.

```
load(ndx)
require(quantmod)
ndx.ret2 = ROC(Cl(ndx), na.pad=FALSE)^2
ndx.rvol = sqrt(apply.weekly(ndx.ret2, FUN=sum))
colnames(ndx.rvol) = "RVOL"
par(mfrow=c(2,2))
plot(ndx.rvol, main="RVOL")
plot(log(ndx.rvol), main="Log RVOL")
ndx.acf = acf(log(as.numeric(ndx.rvol)), 25)
ndx.pacf = pacf(log(as.numeric(ndx.rvol)), 25)
spec = starspec(mean.model=list(states=2,arOrder=c(2,2),
statevar="y",ylags=1))
nt1 = nonlinearTest(spec, data=log(ndx.rvol))
nt2 = nonlinearTest(spec, data=log(ndx.rvol), robust=TRUE)
testmat = matrix(NA, ncol=4, nrow=2)
colnames(testmat) =c("F-stat","p-value","Chisq-stat","pvalue")
rownames(testmat) =c("standard","robust")
testmat[1,]=c(nt1$F.statistic,nt1$F.pvalue,nt1$chisq.statistic,nt1$chisq.pvalue)
testmat[2,]=c(nt2$F.statistic,nt2$F.pvalue,nt2$chisq.statistic,nt2$chisq.pvalue)
```

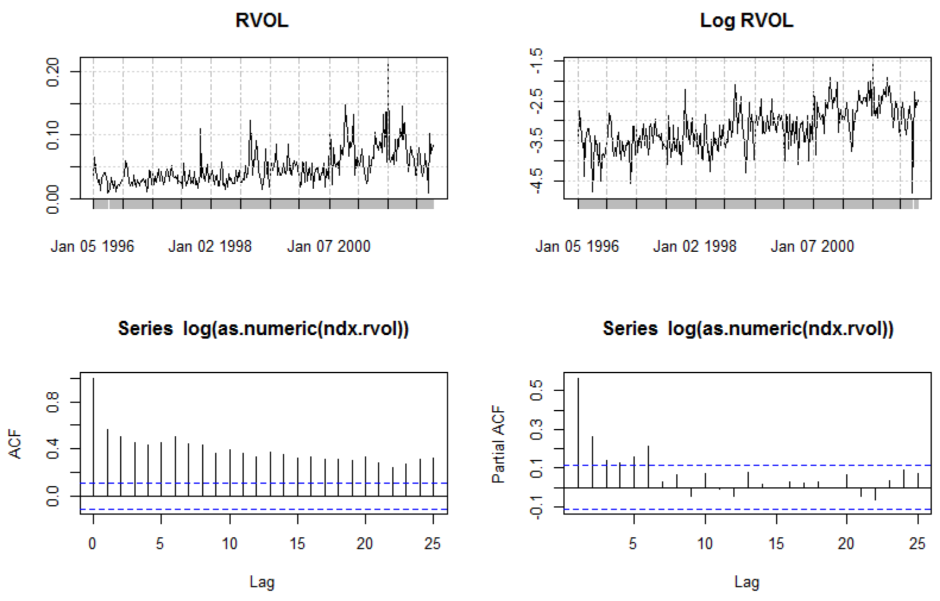

Figure 5: Nasdaq 100 Realized Volatility

```
print(testmat)
```

```
           F-stat     p-value Chisq-stat      pvalue
standard 3.708895 0.001436576   21.31186 0.001612269
robust   2.569438 0.019308720   15.09379 0.019539641
```

This is the same results as given on P.670 and 680 of the Zivot and Wang (2007) book. Finally, estimate the model:

```
ctrl=list(maxit=15000,alpha=1,beta=0.4,gamma=1.2,reltol=1e-12,method="BFGS")
mod = starfit(spec, data = log(ndx.rvol),
solver="strategy",solver.control=ctrl, n=8)
show(mod)
plot(mod)
```

```
*---------------------------------*
*          STAR Model Fit         *
*---------------------------------*
states       : 2
statevar     : y
statear      : FALSE
variance     : static
distribution : norm

Optimal Parameters (Robust Standard Errors)
------------------------------------
          Estimate  Std. Error   t value Pr(>|t|)
s1.phi0   -3.55151    0.052459 -67.70121 0.000000
```

```
s1.phi1    -0.86597    0.296898  -2.91672 0.003537
s1.phi2     0.19690    0.138362   1.42305 0.154721
s2.phi0    -3.71423    2.412406  -1.53964 0.123648
s2.phi1    -0.21872    0.572678  -0.38193 0.702512
s2.phi2     0.21283    0.062668   3.39616 0.000683
s1.gamma    2.74428    1.449708   1.89298 0.058360
s1.c       -2.65726    0.208624 -12.73707 0.000000
s1.alpha1   1.00000          NA        NA       NA
sigma       0.40946    0.021635  18.92585 0.000000

LogLikelihood : -158.8607

Akaike       1.1117
Bayes        1.2222
Shibata      1.1100
Hannan-Quinn 1.1559

r.squared         :  0.4114
r.squared (adj)   :  0.3932
RSS               :  50.63322
skewness (res)    :  -0.36457
ex.kurtosis (res) :  1.33252

AR roots
          Moduli1  Moduli2
state_1 0.9497038 5.347780
state_2 2.7415442 1.713851
```

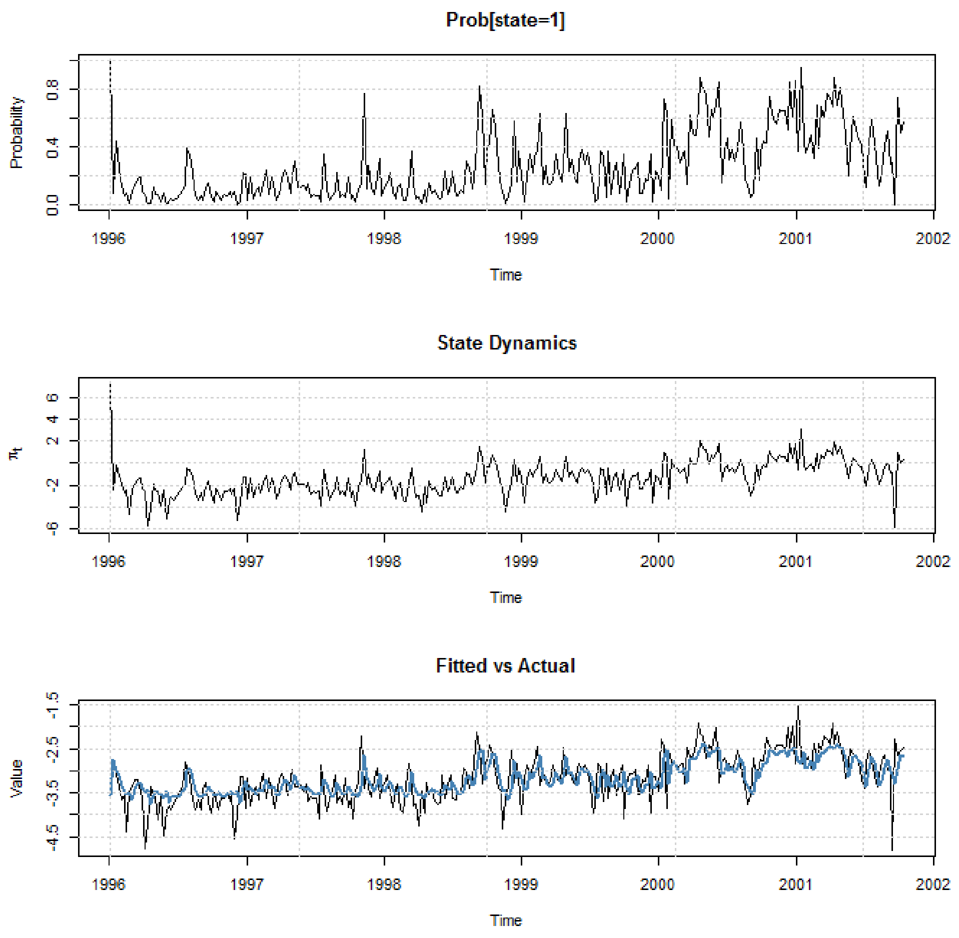

Figure 6: Nasdaq 100 Estimated 2-state STAR model

References

T Astatkie, DG Watts, and WE Watt. Nested threshold autoregressive (netar) models. *International Journal of Forecasting*, 13(1):105–116, 1997.

SA Billings and WSF Voon. Structure detection and model validity tests in the identification of nonlinear systems. In *IEE Proceedings D (Control Theory and Applications)*, volume 130, pages 193–199. IET, 1983.

SA Billings and WSF Voon. Correlation based model validity tests for non-linear models. *International journal of Control*, 44(1):235–244, 1986.

Peter J Brockwell, Jian Liu, and Richard L Tweedie. On the existence of stationary threshold autoregressive moving-average process. *Journal of Time Series Analysis*, 13(2):95–107, 1992.

FL Carmichael. The arc tangent in trend determination. *Journal of the American Statistical Association*, 23(163):253–262, 1928.

Felix Chan and Michael McAleer. Maximum likelihood estimation of star and star-garch models: theory and monte carlo evidence. *Journal of Applied Econometrics*, 17(5):509–534, 2002.

Felix Chan and Michael McAleer. Estimating smooth transition autoregressive models with garch errors in the presence of extreme observations and outliers. *Applied Financial Economics*, 13 (8):581–592, 2003.

K. S. Chan and H. Tong. On estimating thresholds in autoregressive models. *Journal of Time Series Analysis*, 7:178–190., 1986.

Dick van Dijk, Timo Teräsvirta, and Philip Hans Franses. Smooth transition autoregressive modelsŮa survey of recent developments. *Econometric Reviews*, 21(1):1–47, 2002.

Philip Hans Franses and Dick van Dijk. *Non-linear time series models in empirical finance.* Cambridge University Press, 2000.

Jan De Gooijer. On threshold moving-average models. *Journal of Time Series Analysis*, 19(1): 1–18, 1998.

Clive WJ Granger and Timo Terasvirta. Modelling non-linear economic relationships. *OUP Catalogue*, 1993.

Gabriel Huerta, Wenxin Jiang, and Martin A Tanner. Time series modeling via hierarchical mixtures. *Statistica Sinica*, 13(4):1097–1118, 2003.

Leena Kalliovirta, Mika Meitz, and Pentti Saikkonen. A gaussian mixture autoregressive model for univariate time series. *Working Paper*, 2012.

Heikki Kauppi and Pentti Saikkonen. Predicting us recessions with dynamic binary response models. *The Review of Economics and Statistics*, 90(4):777–791, 2008.

Chien-Fu Jeff Lin and Timo Teräsvirta. Testing the constancy of regression parameters against continuous structural change. *Journal of Econometrics*, 62(2):211–228, 1994.

Stefan Lundbergh, Timo Teräsvirta, and Dick van Dijk. Time-varying smooth transition autoregressive models. *Journal of Business & Economic Statistics*, 21(1):104–121, 2003.

Ritva Luukkonen, Pentti Saikkonen, and Timo Teräsvirta. Testing linearity against smooth transition autoregressive models. *Biometrika*, 75(3):491–499, 1988.

Henri Nyberg. Dynamic probit models and financial variables in recession forecasting. *Journal of Forecasting*, 29(1-2):215–230, 2010.

Richard E Quandt. The estimation of the parameters of a linear regression system obeying two separate regimes. *Journal of the american statistical association*, 53(284):873–880, 1958.

Mike KP So, WK Li, and K Lam. A threshold stochastic volatility model. *Journal of Forecasting*, 21(7):473–500, 2002.

T. Teräsvirta. Specification, estimation, and evaluation of smooth transition autoregressive models. *Journal of the American Statistical Association*, 89:208–218, 1994.

Howell Tong and DK Ghaddar. Data transformation and self-exciting threshold autoregression. *Journal of the Royal Statistical Society, series C*, 30:238–248, 1981.

Howell Tong and K.S Lim. Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society, Series B*, 42(3):245–292, 1980.

Dick van Dijk and Philip Hans Franses. Modeling multiple regimes in the business cycle. *Macroeconomic Dynamics*, 3:311–340, 1999.

Jean-Michel Zakoian. Threshold heteroskedastic models. *Journal of Economic Dynamics and Control*, 18(5):931–955, 1994.

Q.M. Zhu and S. Billings. Parameter estimation for stochastic nonlinear rational models. *International Journal of Control*, 57(2):309–333, 1993.

Eric Zivot and Jiahui Wang. *Modeling Financial Time Series with S-PLUS®*, volume 191. Springer, 2007.