18
Nonlinear Time Series Models

18.1 Introduction

Most of the time series models discussed in the previous chapters are linear time series models. Although they remain at the forefront of academic and applied research, it has often been found that simple linear time series models usually leave certain aspects of economic and financial data unexplained. Since economic and financial systems are known to go through both structural and behavioral changes, it is reasonable to assume that different time series models may be required to explain the empirical data at different times. This chapter introduces some popular nonlinear time series models that have been found to be effective at modeling nonlinear behavior in economic and financial time series data.

To model nonlinear behavior in economic and financial time series, it seems natural to allow for the existence of different *states of the world* or *regimes* and to allow the dynamics to be different in different regimes. This chapter focuses on models that assume in each regime the dynamic behavior of the time series is determined by an autoregressive (AR) model, such as threshold AR, self-exciting threshold AR and smooth transition AR models. This is because simple AR models are arguably the most popular time series model and are easily estimated using regression methods. By extending AR models to allow for nonlinear behavior, the resulting nonlinear models are easy to understand and interpret. In addition, this chapter also covers more general Markov switching models using state space representations. The types of models that can be cast into this form are enormous.

However, there are many other types of nonlinear time series models that are not covered in this chapter, such as bilinear models, k nearest neighbor methods and neural network models[1]. Book length treatment of nonlinear time series models can be found in Tong (1990), Granger and Teräsvirta (1993) and Franses and van Dijk (2000). Kim and Nelson (1999) provides a comprehensive account of different Markov switching models that have been used in economic and financial research.

Given the wide range of nonlinear time series models available and the inherent flexibility of these models, the possibility of getting a spuriously good fit to any time series data set is very high. Therefore it is usually recommended to perform a test of linearity against nonlinearity before building a possibly complex nonlinear model. Section 18.2 first introduces a popular test for nonlinearity, the BDS test, which has been found to have power against a wide range of nonlinear time series models. There are many other types of nonlinearity tests that are developed to test against specific nonlinear models. Some of these tests will be introduced together with the nonlinear models in later sections. For example, Section 18.3 introduces threshold AR models and two tests for threshold nonlinearity, and Section 18.4 introduces smooth transition AR (STAR) models and a test for STAR nonlinearity. Finally Section 18.5 describes the Markov switching state space models and Section 18.6 gives an extended example of how to estimate Markov switching models in `S+FinMetrics`.

18.2 BDS Test for Nonlinearity

The BDS test developed by Brock, Dechert and Scheinkman (1987) (and later published as Brock, Dechert, Scheinkman and LeBaron, 1996) is arguably the most popular test for nonlinearity. It was originally designed to test for the null hypothesis of independent and identical distribution (iid) for the purpose of detecting non-random chaotic dynamics.[2] However, many studies have shown that BDS test has power against a wide range of linear and nonlinear alternatives, for example, see Brock, Hsieh and LeBaron (1991) and Barnett, Gallant, Hinich, Jungeilges, Kaplan and Jensen (1997). In addition, it can also be used as a portmanteau test or mis-specification test when applied to the residuals from a fitted model. In particular, when applied to the residuals from a fitted *linear* time series model, the BDS test can be used to detect remaining dependence and the presence of omitted nonlinear structure. If the null hypothesis cannot be rejected, then the original linear model cannot be rejected; if the null

[1] A function to estimate single-hidden-layer neural network models is in the `nnet` library provided with S-PLUS.

[2] Loosely speaking, a time series is said to be "chaotic" if it follows a nonlinear deterministic process but looks random.

hypothesis is rejected, the fitted linear model is mis-specified, and in this sense, it can also be treated as a test for nonlinearity.

18.2.1 BDS Test Statistic

The main concept behind the BDS test is the *correlation integral*, which is a measure of the frequency with which temporal patterns are repeated in the data. Consider a time series x_t for $t = 1, 2, \cdots, T$ and define its m-history as $x_t^m = (x_t, x_{t-1}, \cdots, x_{t-m+1})$. The correlation integral at *embedding dimension* m can be estimated by:

$$C_{m,\epsilon} = \frac{2}{T_m(T_m - 1)} \sum_{m \le s < t \le T} \sum I(x_t^m, x_s^m; \epsilon) \tag{18.1}$$

where $T_m = T - m + 1$ and $I(x_t^m, x_s^m; \epsilon)$ is an indicator function which is equal to one if $|x_{t-i} - x_{s-i}| < \epsilon$ for $i = 0, 1, \cdots, m-1$ and zero otherwise. Intuitively the correlation integral estimates the probability that any two m-dimensional points are within a distance of ϵ of each other. That is, it estimates the joint probability:

$$\Pr(|x_t - x_s| < \epsilon, |x_{t-1} - x_{s-1}| < \epsilon, \cdots, |x_{t-m+1} < x_{s-m+1}| < \epsilon)$$

If x_t are iid, this probability should be equal to the following in the limiting case:

$$C_{1,\epsilon}^m = \Pr(|x_t - x_s| < \epsilon)^m$$

Brock, Dechert, Scheinkman and LeBaron (1996) define the *BDS statistic* as follows:

$$V_{m,\epsilon} = \sqrt{T} \frac{C_{m,\epsilon} - C_{1,\epsilon}^m}{s_{m,\epsilon}} \tag{18.2}$$

where $s_{m,\epsilon}$ is the standard deviation of $\sqrt{T}(C_{m,\epsilon} - C_{1,\epsilon}^m)$ and can be estimated consistently as documented by Brock, Dechert, Scheinkman and LeBaron (1997). Under fairly moderate regularity conditions, the BDS statistic converges in distribution to $N(0, 1)$:

$$V_{m,\epsilon} \xrightarrow{d} N(0, 1) \tag{18.3}$$

so the null hypothesis of iid is rejected at the 5% significance level whenever $|V_{m,\epsilon}| > 1.96$.

18.2.2 Size of BDS Test

`S+FinMetrics` provides the `BDSTest` function for performing the BDS test.[3] The arguments expected by `BDSTest` function are:

[3]The `BDSTest` function is implemented using the C source file provided by LeBaron (1997). The same test can also be performed by calling `nonlinearTest` function with the optional argument `method` set to `"BDS"`.

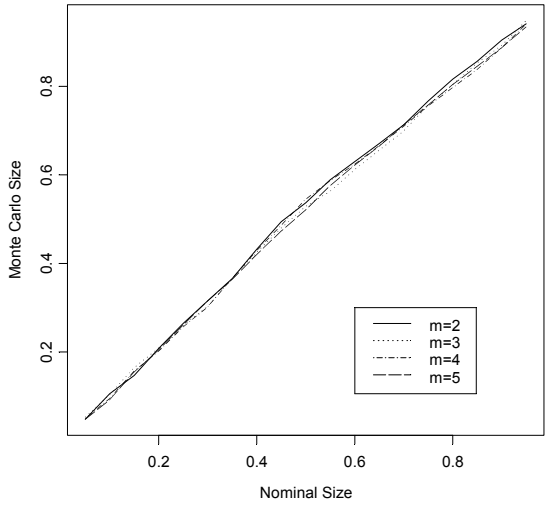

FIGURE 18.1. Size of BDS test statistics using t distribution.

```
> args(BDSTest)
function(x, m = 3, eps = NULL, variable.removal = T)
```

where `x` specifies the time series to be tested, `m` instructs the test to use the embedding dimensions from `2` to `m`, and `eps` specifies in units of sample standard deviations the distance threshold ϵ in (18.1). By default, `BDSTest` computes the BDS statistics with ϵ set to 0.5, 1, 1.5 and 2 standard deviations of the data set. When the optional argument `variable.removal` is set to `TRUE`, different numbers of points in the sample are removed for different values of m such that the test is always computed using all the sample observations available; if it is set to `FALSE`, the same points are removed for different values of m such that the test is always computed using the same sample observations.

Example 116 *Size of BDS test*

The following script illustrates how to use the `BDSTest` function in a Monte Carlo experiment to investigate the size of the BDS test:

```
set.seed(10)
size.mat = matrix(0, 1000, 4)
for (i in 1:1000) {
  if (i %% 100 == 0) {
    cat("i =", i, "\n")
  }
```

```
  test.dat = rt(500, df=8)
  size.mat[i,] = BDSTest(test.dat, m=5, eps=1)$stat[,1]
}
```

One advantage of the BDS test is that it is a statistic which requires no distributional assumption on the data to be tested. In fact, in the above Monte Carlo experiment, the data are simulated from a t distribution with 8 degrees of freedom. Each simulated sample has 500 observations, which is usually thought to be the minimal sample size for the BDS test to have reliable performance. The data are simulated 1000 times and BDS statistics using embedding dimensions from 2 to 5 are computed by setting ϵ to one standard deviation of the sample observations. The following commands plot the size of the "one-sided" test against its nominal value:[4]

```
> size.p = seq(0.05, 0.95, by=0.05)
> size.q = qnorm(size.p)
> size.bds = apply(size.mat, 2,
+      function(x) colMeans(outer(x, size.q, FUN="<=")))
> par(fty="s")
> matplot(matrix(size.p, nrow=length(size.p), ncol=4),
+         size.bds, type="l",
+         xlab="Nominal Size", ylab="Monte Carlo Size")
> legend(0.6, 0.3, paste("m=",2:5,sep=""), type="l", lty=1:4)
```

and the result is shown in Figure 18.1. Considering the Monte Carlo experiment is conducted using only 1000 replications, the plot shows the test has very good size behavior for all the chosen embedding dimensions.

18.2.3 BDS Test As a Nonlinearity Test and a Mis-specification Test

Another advantage of the BDS test is that when applied to model residuals, the first order asymptotic distribution of BDS statistic given in (18.3) is independent of estimation errors under certain sufficient conditions. In general, de Lima (1996) shows that for linear additive models, or models that can be transformed into that format, the BDS test is nuisance parameter free and does not require any adjustment when applied to fitted model residuals. Thus the BDS test can be used as a test for nonlinearity, or as a test for model mis-specification.

Example 117 *Nonlinearity in weekly returns of Dutch Guilder foreign exchange rates*

[4]The BDS test is actually a two-sided test. However, for the purpose of illustrating distributional properties of BDS statistics, the plots are generated using the "incorrect" one-sided test.

The "timeSeries" data set DFX.ts in S+FinMetrics contains weekly returns on the Dutch Guilder spot exchange rate from January 1980 to December 1998. To test for the existence of nonlinearity in this data set, use the following command:

```
> BDSTest(DFX.ts, m=5)

BDS Test for Independence and Identical Distribution

Null Hypothesis: DFX.ts is independently and identically
distributed.

Embedding dimension =  2 3 4 5

Epsilon for close points =  0.0073 0.0146 0.0219 0.0291

Test Statistics =
      [ 0.01 ] [ 0.01 ] [ 0.02 ] [ 0.03 ]
[ 2 ]   1.0802   1.5908   1.9991   2.6097
[ 3 ]   3.1661   3.0984   3.5817   4.1536
[ 4 ]   4.0523   3.9006   4.4871   5.1613
[ 5 ]   5.2798   4.7189   5.3238   5.9882

p-value =
      [ 0.01 ] [ 0.01 ] [ 0.02 ] [ 0.03 ]
[ 2 ]   0.2801   0.1117   0.0456   0.0091
[ 3 ]   0.0015   0.0019   0.0003   0.0000
[ 4 ]   0.0001   0.0001   0.0000   0.0000
[ 5 ]   0.0000   0.0000   0.0000   0.0000
```

In the above output, the default values of $\epsilon = (0.5, 1.0, 1.5, 2.0)$ used in the test are converted back to the units of the original data, and the null hypothesis that the data is iid is rejected for most combinations of m and ϵ at conventional significance levels. Since there is almost no discernible linear structure in the levels of DFX.ts, the results from the BDS test suggest that there may be nonlinear structure in the data.

One possibility to model the nonlinear structure in DFX.ts is to use a GARCH(1,1) model:

```
> DFX.garch = garch(DFX.ts~1, ~garch(1,1), trace=F)
> summary(DFX.garch)$coef
                Value     Std.Error    t value      Pr(>|t|)
       C 0.00021084425 3.939145e-004  0.5352539 5.925817e-001
       A 0.00001942582 5.508377e-006  3.5265964 4.381551e-004
 ARCH(1) 0.10297320531 2.096693e-002  4.9112210 1.041116e-006
GARCH(1) 0.80686268689 3.798031e-002 21.2442379 0.000000e+000
```

All the estimated parameters in `DFX.garch` are highly significant except for the conditional mean parameter `C`. To evaluate if the GARCH(1,1) model adequately captures the nonlinear structure in `DFX.ts`, the BDS test can be used again on the standardized residuals of `DFX.garch` as a misspecification test. There are two ways to apply the BDS test to GARCH standardized residuals: one is to apply the BDS test directly to the standardized residuals:

```
> BDSTest(residuals(DFX.garch, standard=T), m=5,
+ eps=c(0.5, 1, 1.5))

BDS Test for Independence and Identical Distribution

Null Hypothesis: residuals(DFX.garch, standard = T) is
independently and identically distributed.

Embedding dimension =  2 3 4 5

Epsilon for close points =  0.5002 1.0004 1.5006

Test Statistics =
      [ 0.5 ]   [ 1 ] [ 1.5 ]
[ 2 ] -1.9487 -1.5430 -1.6035
[ 3 ] -1.4581 -1.1172 -1.2687
[ 4 ] -1.2832 -0.9735 -1.1355
[ 5 ] -0.8634 -0.6079 -0.8305

p-value =
      [ 0.5 ]  [ 1 ] [ 1.5 ]
[ 2 ]  0.0513 0.1228  0.1088
[ 3 ]  0.1448 0.2639  0.2045
[ 4 ]  0.1994 0.3303  0.2561
[ 5 ]  0.3879 0.5432  0.4062
```

and the other is to apply it to the logarithms of squared standardized residuals:[5]

```
> BDSTest(log(residuals(DFX.garch, standard=T)^2),
+ m=5, eps=c(0.5, 1, 1.5))

BDS Test for Independence and Identical Distribution
```

[5] When `BDSTest` function is applied to a fitted model object, it is currently always applied to the residuals of the fittd model, instead of standardized residuals or logarithms of squared standardized residuals.

```
Null Hypothesis: log(residuals(DFX.garch, standard = T)^2)
is independently and identically distributed.

Embedding dimension =  2 3 4 5

Epsilon for close points =  1.1218 2.2435 3.3653

Test Statistics =
      [ 1.12 ] [ 2.24 ] [ 3.37 ]
[ 2 ]  -0.6461  -0.5538  -0.5463
[ 3 ]  -0.8508  -0.9030  -0.9175
[ 4 ]  -0.7540  -0.9977  -1.0821
[ 5 ]  -0.9397  -0.8581  -1.0252

p-value =
      [ 1.12 ] [ 2.24 ] [ 3.37 ]
[ 2 ]   0.5182   0.5797   0.5849
[ 3 ]   0.3949   0.3665   0.3589
[ 4 ]   0.4509   0.3184   0.2792
[ 5 ]   0.3474   0.3909   0.3052
```

Here, both ways of applying the BDS test suggest that the GARCH(1,1) model provides an adequate fit to the original data and successfully removes the nonlinearity in the data. In general, when applied to standardized residuals from a fitted GARCH model, earlier studies (for example, see Brock, Hsieh and LeBaron, 1991) suggest that the BDS statistic needs to be adjusted to have the right size and Monte Carlo simulations are usually relied upon to derive the adjustment factor for specific GARCH models. However, following suggestions in Brock and Potter (1993) and de Lima (1996), recent studies (for example, see Caporale, Ntantamis, Pantelidis and Pittis, 2004 and Fernandes and Preumont, 2004) show that if applied to the logarithms of squared standardized residuals from a fitted GARCH model, the BDS test actually has correct size, because the logarithmic transformation casts the GARCH model into a linear additive model which satisfies the conditions in de Lima (1996) for the BDS test to be nuisance parameter free.[6]

Example 118 *Size of BDS mis-specification test for GARCH models*

The following script performs a Monte Carlo experiment to illustrate the different size behavior of the BDS test when applied to standardized residuals and logarithms of squared standardized residuals for the GARCH(1,1) model. The data sets are simulated using the GARCH fit in `DFX.garch` with

[6]Since GARCH models with leverage effects cannot be transformed into a linear additive model, BDS test may not have good size behavior for those models.

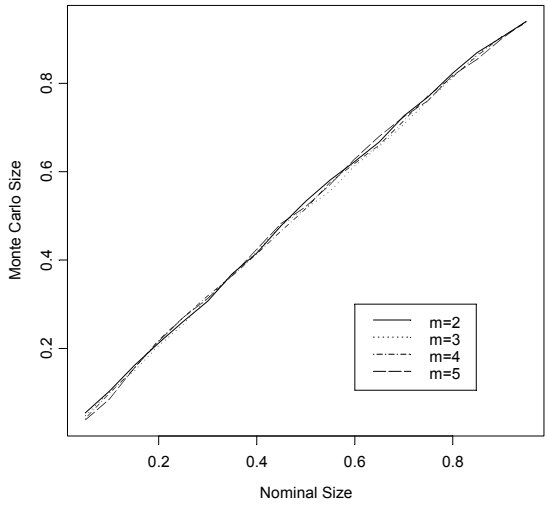

FIGURE 18.2. Size of BDS test when applied to logarithms of squared standardized GARCH residuals.

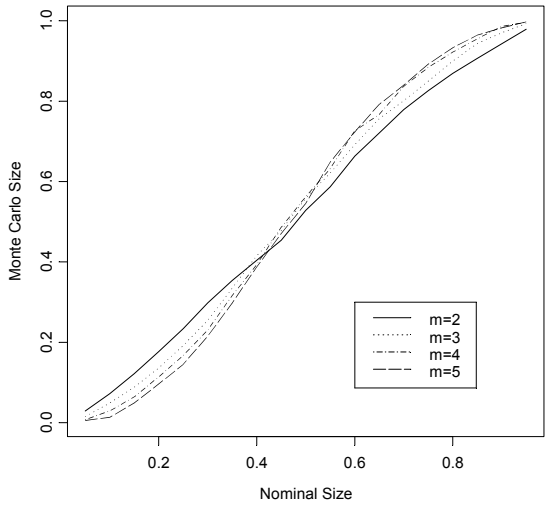

FIGURE 18.3. Size of BDS test when applied to standardized GARCH residuals.

1000 observations. The GARCH estimation and BDS test are repeated 1000 times.

```
set.seed(10)
sim.garch.dat = simulate(DFX.garch, sigma=F, n.start=500,
    n=1000, n.rep=1000)
size.garch.res = matrix(0, 1000, 4)
size.garch.log = matrix(0, 1000, 4)
for (i in 1:1000) {
  tmp = garch(sim.garch.dat[,i]~1, ~garch(1,1), trace=F)
  if (i %% 10 == 0)
    cat("Simulation No.", i, "\n")
  tmp.res = residuals(tmp, standardized=T)
  size.garch.res[i,] = BDSTest(tmp.res, m=5, eps=1)$stat[,1]
  size.garch.log[i,] = BDSTest(log(tmp.res^2), m=5,
                               eps=1)$stat[,1]
}

size.p = seq(0.05, 0.95, by=0.05)
size.q = qnorm(size.p)
size.garch.res = apply(size.garch.res, 2,
    function(x) colMeans(outer(x, size.q, FUN="<=")))
size.garch.log = apply(size.garch.log, 2,
    function(x) colMeans(outer(x, size.q, FUN="<=")))
```

As in Example 116, the sizes of the "one-sided" test applied to the standardized residuals and the logarithms of squared standardized residuals are plotted against the nominal sizes in Figure 18.3 and Figure 18.2, respectively. Obviously the sizes of the BDS test computed using standardized residuals are off and become more conservative for larger values of m, but those using logarithms of squared standardized residuals are reliable.

18.3 Threshold Autoregressive Models

As discussed in the previous section, when there is no prior knowledge about the type of nonlinearity a time series may have, the BDS test can be used to test for the existence of nonlinearity in either the time series itself or the residuals from a fitted linear time series model. However, sometimes economic or financial theory, or even stylized empirical facts, may suggest a specific form of nonlinearity for a time series. In these cases, it is usually preferred to perform the test for the specific form of nonlinearity and build a nonlinear time series model for the form of nonlinearity detected.

One popular class of nonlinear time series models is the *threshold autoregressive* (TAR) models, which is probably first proposed by Tong (1978) and discussed in detail in Tong (1990). The TAR models are simple and

easy to understand, but rich enough to generate complex nonlinear dynamics. For example, it can be shown that the TAR models can have limit cycles and thus be used to model periodic time series, or produce asymmetries and jump phenomena that cannot be captured by a linear time series model.

In spite of the simplicity of the TAR model form, there are many free parameters to estimate and variables to choose when building a TAR model, and this has hindered its early use. Recently, however, much progress has been made with regard to specification and estimation of TAR models. The next section introduces the general form of TAR models and a special class called SETAR models, and then illustrates how to perform tests for threshold nonlinearity and estimate unknown parameters in TAR models using ready-to-use functions in `S+FinMetrics`.

18.3.1 TAR and SETAR Models

Consider a simple AR(p) model for a time series y_t:[7]

$$y_t = \mu + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \sigma\epsilon_t \tag{18.4}$$

where ϕ_i $(i = 1, 2, \cdots, p)$ are the AR coefficients, $\epsilon_t \sim WN(0, 1)$ and $\sigma > 0$ is the standard deviation of disturbance term. The model parameters $\boldsymbol{\phi} = (\mu, \phi_1, \phi_2, \cdots, \phi_p)$ and σ are independent of time t and remain constant. To capture nonlinear dynamics, TAR models allow the model parameters to change according to the value of a weakly exogenous *threshold variable* z_t:

$$y_t = \mathbf{X}_t\boldsymbol{\phi}^{(j)} + \sigma^{(j)}\epsilon_t \text{ if } r_{j-1} < z_t \leq r_j \tag{18.5}$$

where $\mathbf{X}_t = (1, y_{t-1}, y_{t-2}, \cdots, y_{t-p})$, $j = 1, 2, \cdots, k$, and $-\infty = r_0 < r_1 < \cdots < r_k = \infty$. In essence, the $k-1$ non-trivial *thresholds* $(r_1, r_2, \cdots, r_{k-1})$ divide the domain of the threshold variable z_t into k different regimes. In each different regime, the time series y_t follows a different AR(p) model.[8]

When the threshold variable $z_t = y_{t-d}$, with the *delay parameter* d being a positive integer, the dynamics or regime of y_t is determined by its own lagged value y_{t-d} and the TAR model is called a *self-exciting* TAR or SETAR model. For the ease of notation, let SETAR(1) denote the one-regime linear AR model with $k = 1$, SETAR(2) denote the two-regime TAR model with $k = 2$, etc. For the one-regime SETAR(1) model, $-\infty = r_0 < r_1 = \infty$ and the unknown parameters are $\boldsymbol{\Theta} = (\boldsymbol{\phi}^{(1)}, \sigma^{(1)})$; for the

[7] See Chapter 3 and the references therein for basic concepts in linear time series analysis.

[8] Although the AR order p is assumed to be the same in different regimes throughout this chapter and in the related `S+FinMetrics` functions for the ease of illustration and programming, in theory the AR order can be different for different regimes.

two-regime SETAR(2) model, the unknown parameters include the single threshold $-\infty < r_1 < \infty$ and $\boldsymbol{\Theta} = (\boldsymbol{\phi}^{(1)}, \boldsymbol{\phi}^{(2)}, \sigma^{(1)}, \sigma^{(2)})$.

The next section introduces two approaches for testing threshold nonlinearity and estimating the unknown parameters in the associated SETAR models, following Tsay (1989) and Hansen (1997), respectively. Although the illustrations and examples focus on SETAR models, the theory and procedures can also be applied to TAR models in general. Finally, note that if only the intercept terms $\mu^{(j)}$ are different in different regimes, SETAR models can be used to capture *level shifts* in y_t; if only the variance terms $\sigma^{(j)}$ are different in different regimes, SETAR models can be used to capture *additive outliers* or *innovation outliers* in y_t. Chapter 17 provides a more comprehensive approach for analyzing time series models that are robust to level shifts and outliers.

18.3.2 Tsay's Approach

Before developing a SETAR model, it is preferred to test for the existence of threshold-type nonlinearity in the time series first. The null hypothesis is usually the time series y_t follows the SETAR(1) model, while the alternative hypothesis is that y_t follows a SETAR(j) model with $j > 1$. One complicating issue in testing for threshold nonlinearity is that the thresholds r_i for $i = 1, 2, \cdots, k-1$ are only identified under the alternative hypothesis. To avoid dealing with the thresholds directly, Tsay (1989) proposes a conventional F test based on an auxiliary regression.

Arranged Autoregression and Tsay's F Test

Tsay's approach centers on the use of an *arranged autoregression* with recursive least squares (RLS) estimation. Consider the SETAR model in (18.5) with $z_t = y_{t-d}$. Since the threshold values r_i are usually unknown, Tsay suggests to arrange the equations in (18.5) for $t = \max(d, p)+1, \cdots, n$, where n is the sample size, such that the equations are sorted according to the threshold variable y_{t-d} which may take any value in $\mathbf{Y}_d = (y_h, \cdots, y_{n-d})$ with $h = \max(1, p+1-d)$:

$$y_{\pi_i} = \mathbf{X}_{\pi_i}\hat{\boldsymbol{\phi}} + \hat{\sigma}\epsilon_{\pi_i} \tag{18.6}$$

where $i = 1, 2, \cdots, n'$, $n' = n - d - h + 1$ is the effective sample size for the above arranged autoregression, and π_i corresponds to the index in the original sample such that $y_{\pi_i - d}$ is the i-th smallest value in $\mathbf{Y}_d$. For example, if y_{10} is the smallest value in $\mathbf{Y}_d$, then $\pi_1 = 10 + d$; if y_{20} is the second smallest value in $\mathbf{Y}_d$, then $\pi_2 = 20 + d$, etc. So if the original time series is generated by a SETAR(2) model and there are $m < n$ values in $\mathbf{Y}_d$ that are smaller than the threshold r_1, then the first m equations in (18.6) correspond to the first regime and the remaining equations correspond to the second regime.

To test for the existence of threshold-type nonlinearity, Tsay suggests to compute RLS estimates of $\hat{\boldsymbol{\phi}}$ in (18.6). If there is no threshold nonlinearity, the standardized predictive residuals $\hat{e}_{\pi_i}$ from RLS of (18.6) should be white noise asymptotically and orthogonal to $\mathbf{X}_{\pi_i}$. However, if y_t is a SETAR(j) process with $j > 1$, the RLS estimates of $\hat{\boldsymbol{\phi}}$ are biased and $\hat{\boldsymbol{\Psi}}$ in the following auxiliary regression will be statistically significant:

$$\hat{e}_{\pi_i} = \mathbf{X}'_{\pi_i}\boldsymbol{\Psi} + u_{\pi_i} \tag{18.7}$$

Thus the conventional F statistic for testing $\boldsymbol{\Psi} = \mathbf{0}$ the above regression can be used as a test for threshold nonlinearity.

Example 119 *SETAR nonlinearity in NASDAQ realized volatility*

To illustrate the usage of Tsay's F test for threshold nonlinearity, consider the weekly realized volatility of NASDAQ 100 index constructed as follows from the `S+FinMetrics` data set `ndx.dat`:

```
> ndx.ret2 = getReturns(ndx.dat[,"Close"])^2
> ndx.rvol = sqrt(aggregate(ndx.ret2, FUN=sum, by="weeks",
+ week.align=1))
> colIds(ndx.rvol) = "RVOL"
> par(mfrow=c(2,2))
> plot(ndx.rvol, reference.grid=F, main="RVOL")
> plot(log(ndx.rvol), reference.grid=F, main="Log RVOL")
```

The levels and the logarithms of the weekly realized volatility series are shown in the top half of Figure **??**. The time series plots suggest that the volatility may have switched to a different regime after the first quarter of 2000. Before testing for threshold nonlinearity, the ACF and PACF plots can be used to help identify the autoregressive order to use:

```
> ndx.acf = acf(log(ndx.rvol))
> ndx.pacf = acf(log(ndx.rvol), type="partial")
```

The resulting plots are shown in the bottom half of Figure 18.4. The ACF function decays very slowly and remains significant even after 30 lags, while the PACF function is significant for the first six lags. This suggests that an AR model with order from 2 to 6 may be considered as a starting point for modeling the logarithms of realized volatility `log(ndx.rvol)`.[9]

The `S+FinMetrics` function `nonlinearTest` can now be used to test for threshold nonlinearity:

```
> nonlinearTest(log(ndx.rvol), method="threshold", p=6, d=1:6)
```

[9]Hereinafter the logarithms of `ndx.rvol` are used because usually the logarithms of realized volatility tend to be normally distributed. See Andersen, Bollerslev, Diebold and Ebens (2001) for a detailed analysis of properties of realized volatility for stock returns.

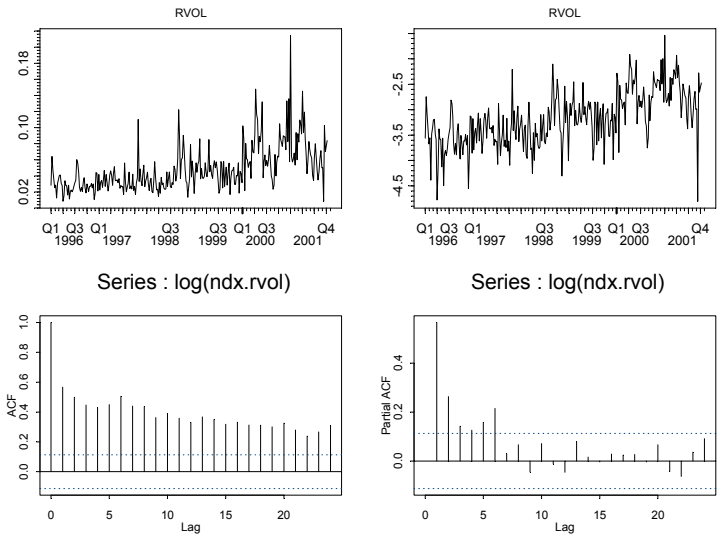

FIGURE 18.4. Weekly realized volatility of NASDAQ 100 index.

```
Nonlinearity Test: Threshold Nonlinearity

Null Hypothesis: no threshold nonlinearity

    F-stat     dof  P-val
d=1 1.2568 (7,253) 0.2724
d=2 1.4203 (7,253) 0.1974
d=3 1.2586 (7,253) 0.2714
d=4 0.5104 (7,253) 0.8264
d=5 0.5224 (7,253) 0.8173
d=6 0.1179 (7,253) 0.9971
```

Note that the optional argument `p` specifies the AR order to use in the arranged autoregression, and the optional argument `d` is used to select the delay parameters from 1 to 6. The output gives the F statistics and their corresponding p-values for all chosen values of delay parameter d, and shows that the evidence for threshold nonlinearity is not strong with the AR(6) specification. Since a high order AR model may actually approximate nonlinear dynamics relatively well, a lower order AR(2) specification may also be tried:

```
> nonlinearTest(log(ndx.rvol), method="threshold", p=2, d=1:2)
```

```
Nonlinearity Test: Threshold Nonlinearity

Null Hypothesis: no threshold nonlinearity

    F-stat     dof  P-val
d=1 4.4468 (3,265) 0.0046
d=2 4.0010 (3,265) 0.0082
```

Now the null hypothesis of no threshold nonlinearity is actually rejected for both $d = 1$ and $d = 2$ with an AR(2) specification!

Choice of Delay Parameter and Thresholds

After rejecting the null hypothesis of no threshold nonlinearity, one proceeds to the next stage of estimating a SETAR model. Tsay (1989) suggests to identify the delay parameter d and the thresholds r_i for $i = 1, \cdots, k-1$ first, and then use least squares (LS) to estimate the unknown parameters $\boldsymbol{\Theta}$ in (18.5) with given values of d and thresholds. As long as there are enough observations in each regime, the LS estimates are consistent.

For a given AR order p, Tsay suggests to choose the delay parameter d such that

$$d = \underset{v \in S}{\operatorname{argmax}} F(p, v)$$

where $F(p, v)$ is the F statistic of the auxiliary regression (18.7) with AR order p and the delay parameter equal to v, and S is a set of values of d to consider. For the NASDAQ realized volatility series, d can be set to 1 according to the nonlinearity test output using this rule.

Tsay (1989) also proposes to use two graphical tools for identifying the threshold values: (1) the scatter plot of standardized predictive residuals $\hat{e}_{\pi_i}$ from the arranged autoregression versus the ordered threshold variable; (2) the scatter plot of the t-statistics of the RLS estimates of $\hat{\boldsymbol{\phi}}$ from the arranged autoregression versus the ordered threshold variable. Both plots may exhibit structural breaks at the threshold values. To produce such plots for the nonlinearity test, set the optional argument `save.RLS` to `TRUE` when calling `nonlinearTest`:

```
> ndx.test = nonlinearTest(log(ndx.rvol), method="threshold",
+ p=2, d=1, save.RLS=T)
> names(ndx.test)
[1] "stat"      "df"        "threshold" "residuals"
[4] "tRatios"   "yd"        "method"
```

The returned object `ndx.test` includes the following components: `yd` is the ordered threshold variable, `residuals` is the standardized predictive residuals and `tRatios` is the t-statistics of RLS estimates of the AR coefficients. To produce the scatter plot of t-statistics versus the ordered threshold variable, for example, use the following commands:

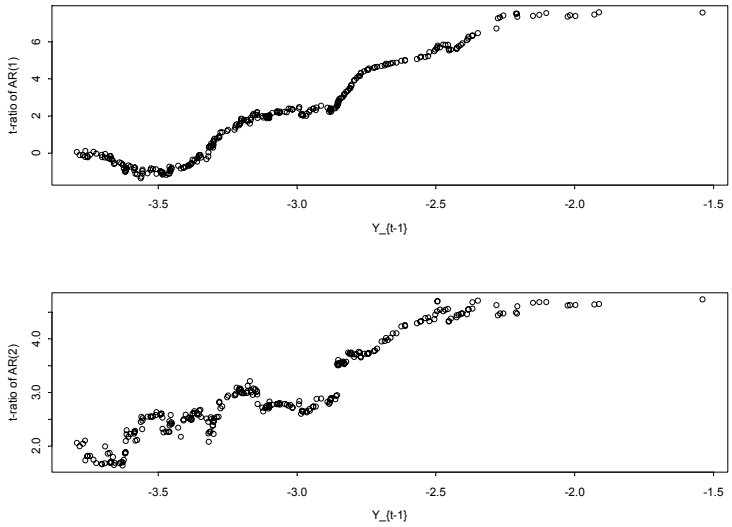

FIGURE 18.5. Scatter plot of t-statistics of RLS estimates of AR coefficients versus ordered threshold variable.

```
> par(mfrow=c(2,1))
> plot(ndx.test$yd, ndx.test$tRatio[,1], xlab="Y_{t-1}",
+      ylab="t-ratio of AR(1)")
> plot(ndx.test$yd, ndx.test$tRatio[,2], xlab="Y_{t-1}",
+      ylab="t-ratio of AR(2)")
```

The plots in Figure 18.5 show that both estimates are significant with t-statistics greater than 2 in absolute values in most cases. In addition, the trend in the t-statistics seems to have two breaks: one occurs when the threshold variable is around -2.8; and the other occurs when the threshold variable is around -2.4. This suggests a SETAR(3) model with two non-trivial threshold values: $r_1 = -2.8$ and $r_2 = -2.4$.

LS Estimates of SETAR Model

After choosing the delay parameter d and the thresholds, other unknown parameters in $\boldsymbol{\Theta}$ of the SETAR model may be simply estimated by LS using the `S+FinMetrics` function `SETAR`, which takes the following arguments:

```
> args(SETAR)
function(x, threshold, p = 1, d = NULL)
```

where the first argument specifies the data to be used, the second argument gives the vector of threshold values, and the optional arguments

`p` and `d` specify the AR order and delay parameter, respectively. To estimate the SETAR(3) model with thresholds $(-2.8, -2.4)$, use the following command:

```
> ndx.setar = SETAR(log(ndx.rvol), c(-2.8, -2.4), p=2, d=1)
> summary(ndx.setar)

Call:
SETAR(x = log(ndx.rvol), threshold = c(-2.8, -2.4), p = 2,
d = 1)

Coefficients:
          regime.1 regime.2 regime.3
Intercept -1.5043  -2.4463  -3.2661
(std.err)  0.2778   1.1323   0.8676
 (t.stat) -5.4157  -2.1605  -3.7643

     lag1  0.2866  -0.0373  -0.6283
(std.err)  0.0776   0.4400   0.3795
 (t.stat)  3.6942  -0.0848  -1.6555

     lag2  0.2573   0.1381   0.2191
(std.err)  0.0687   0.1305   0.1279
 (t.stat)  3.7449   1.0577   1.7138


Std. Errors of Residuals:
 regime.1 regime.2 regime.3
 0.4291   0.3794   0.3583

Information Criteria:
      logL       AIC       BIC        HQ
 -157.5830  333.1659  366.5000  346.5063

                  total regime.1 regime.2 regime.3
Degree of freedom:   300      228       44       19
Time period: from 01/15/1996 to 10/08/2001
```

Note that the AR coefficients for the first regime are estimated to be $(0.29, 0.26)$ which appear to be significant, while the AR coefficients for the second and third regimes are estimated to be $(-0.03, 0.14)$ and $(-0.63, 0.22)$, respectively, and are not very significant. The estimated regime indices can be plotted as follows:

```
> plot(timeSeries(ndx.setar$regime,
+ pos=positions(ndx.rvol)[-(1:2)]), reference.grid=F,
```

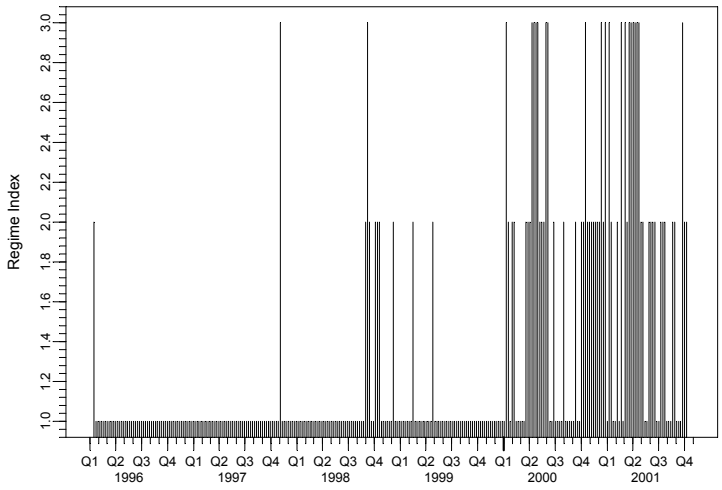

FIGURE 18.6. Estimated regime indices of `ndx.setar`.

```
+ ylab="Regime Index", plot.args=list(type="h"))
```

and the plot is shown in Figure 18.6. It can be seen that most of the observations prior to 2000 fall into the first regime, and the third regime observations usually follow the second regime observations.

Predictions from SETAR Models

After estimating a SETAR model, sometimes a more important task is to generate forecasts of future values of the time series that is of interest. Predictions from SETAR models can be easily computed using Monte Carlo simulations, by following the same principle used for VAR forecasting (see Section 11.3 for details). For example, to generate 1-step-ahead to 100-step-ahead forecasts from the fitted model `ndx.setar`, use the following command:

```
> class(ndx.setar)
[1] "SETAR"
> ndx.pred = predict(ndx.setar, n.predict=100, CI.alpha=0.6,
+ n.sim=10000)
```

Note that the fitted object `ndx.setar` has class `"SETAR"`. By calling the generic `predict` function on `"SETAR"` objects, the simulation-based forecasting method implemented in `predict.SETAR` is automatically applied on the `"SETAR"` objects. The optional argument `n.predict` is used to spec-

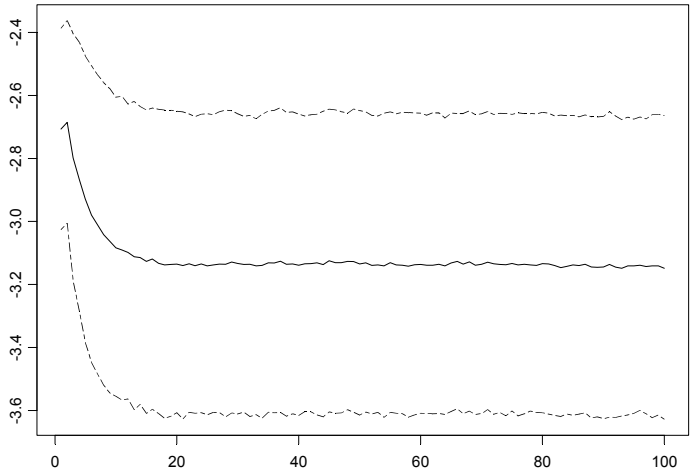

FIGURE 18.7. Predicted realized volatility (in logarithm scale) from `ndx.setar`.

ify the number of forecasts to obtain in the future, the argument `CI.alpha` is used to specify 60% pointwise confidence intervals for the forecasts based on Monte Carlo simulations, and the argument `n.sim` is used to specify the number of simulations to be used for computing the forecasts. The forecasts and their pointwise confidence intervals can be plotted as follows:

```
> tsplot(cbind(ndx.pred$values, ndx.pred$CI), lty=c(1,6,6))
```

and the plot is shown Figure 18.7. After less than 20 steps, the forecasts settle down to the asymptotic mean of the SETAR process.

18.3.3 Hansen's Approach

Although the procedure introduced in the above subsection for identifying and estimating SETAR models is easy to perform, it requires some human decisions especially for choosing the threshold values. This subsection introduces another test for threshold nonlinearity and another procedure for estimating SETAR models as proposed by Hansen (1997). The advantage of this procedure is that the thresholds can be estimated together with other model parameters and valid confidence intervals can be constructed for the estimated thresholds. The disadvantage is that the current imple-

mentation only supports the two-regime SETAR model and thus only one threshold can be estimated.[10]

Hansen's sup-LR Test

Hansen (1997) considers the following two-regime variant of (18.5):

$$y_t = \mathbf{X}_t\boldsymbol{\phi}^{(1)}(1 - I(y_{t-d} > r_1)) + \mathbf{X}_t\boldsymbol{\phi}^{(2)}I(y_{t-d} > r_1) + \epsilon_t \tag{18.8}$$

where $I(A)$ is the indicator function that is equal to 1 if A is true and 0 otherwise, $\epsilon_t \sim iid(0, \sigma^2)$, and there is only one non-trivial threshold r_1. As discussed in the previous subsection, if d and r_1 are known, then the model parameters $\boldsymbol{\Theta} = (\boldsymbol{\phi}^{(1)}, \boldsymbol{\phi}^{(2)}, \sigma^2)$ can be estimated by least squares:

$$\hat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\phi}^{(1)},\boldsymbol{\phi}^{(2)}}{\operatorname{argmin}}\ \hat{\sigma}^2(r_1) = \underset{\boldsymbol{\phi}^{(1)},\boldsymbol{\phi}^{(2)}}{\operatorname{argmin}}\ \frac{1}{n'}\sum_{t=h}^{n}\hat{\epsilon}_t^2 \tag{18.9}$$

where $h = \max(1, p + 1 - d)$ and $n' = n - d - h + 1$ is the effective sample size after adjusting for starting values and the delay parameter.

To test the null hypothesis of SETAR(1) against the alternative hypothesis of SETAR(2), the likelihood ratio test assuming normally distributed errors can be used:

$$F(r_1) = \frac{\text{RSS}_0 - \text{RSS}_1}{\hat{\sigma}_1^2(r_1)} = n'\frac{\hat{\sigma}_0^2 - \hat{\sigma}_1^2(r_1)}{\hat{\sigma}_1^2(r_1)} \tag{18.10}$$

where RSS_0 is the residual sum of squares from SETAR(1), RSS_1 is the residual sum of squares from SETAR(2) given the threshold r_1, and $\hat{\sigma}_0^2$ is the residual variance of SETAR(1). The above test is also the standard F test since (18.8) is a linear regression. However, since the threshold r_1 is usually unknown, Hansen (1997) suggests to compute the following *sup-LR test*:

$$F_s = \sup_{r_1 \in \mathbf{Y}_d} F(r_1) \tag{18.11}$$

by searching over all the possible values of the threshold variable y_{t-d}. In practice, to ensure each regime has a non-trivial proportion of observations, a certain percentage of $\mathbf{Y}_d$ at both ends are usually trimmed and not used.

The sup-LR test has near-optimal power as long as the error term ϵ_t is iid. If ϵ_t is not iid, the F test needs to be replaced by heteroskedasticity-consistent Wald or Lagrange multiplier test. One complicating issue is that since r_1 is only identified under the alternative, the asymptotic distribution of F_s is not χ^2 and non-standard. Hansen (1996) shows that the asymptotic distribution may be approximated by a bootstrap procedure in general, and

[10] Hansen (1999) has generalized this procedure to SETAR models with more than two regimes.

Hansen (1997) gives the analytic form of the asymptotic distribution for testing against SETAR(2) models.

The `nonlinearTest` function in `S+FinMetrics` can also be used to produce Hansen's sup-LR test, simply by setting the optional argument `method` to `"sup-LR"`. For example, to test for threshold nonlinearity in weekly realized volatility of NASDAQ 100 index using the same AR(2) specification and choosing the threshold variable to be $z_t = y_{t-1}$ as in Tsay's F test, use the following command:[11]

```
> nonlinearTest(log(ndx.rvol), method="sup-LR", p=2, d=1,
+ trim.pct=0.1, n.boot=1000)

Nonlinearity Test: Hansen sup-LR Nonlinearity

Null Hypothesis: no threshold with the specified threshold
variable
Under Maintained Assumption of Homoskedastic Errors --

Number of Bootstrap Replications  1000
Trimming percentage               0.1
Threshold Estimate                -2.8768
F-test for no threshold           22.9687
Bootstrap P-Value                 0
```

Note that the optional argument `trim.pct` is used to trim 10% observations at both ends of $\mathbf{Y}_d$, and `n.boot` is used to set the number of bootstrap simulations for computing the p-value of the test. Again, the null hypothesis of no threshold nonlinearity is strongly rejected. To produce the test robust to heteroskedastic errors, simply set the optional argument `hetero` to `TRUE`:

```
> nonlinearTest(log(ndx.rvol), method="sup-LR", p=2, d=1,
+ trim.pct=0.1, n.boot=1000, hetero=T)

Nonlinearity Test: Hansen sup-LR Nonlinearity

Null Hypothesis: no threshold with the specified threshold
variable
Allowing Heteroskedastic Errors using White Correction --

Number of Bootstrap Replications  1000
Trimming percentage               0.1
Threshold Estimate                -2.8768
```

[11] General TAR alternatives with arbitrary threshold variable can also be tested by using setting the optional argument `q` instead of `d`.

```
F-test for no threshold          18.7357
Bootstrap P-Value                0
```

Sequential Estimation of SETAR Models

After confirming the existence of threshold nonlinearity, Hansen (1997) suggests to estimate the threshold value r_1 together with $\boldsymbol{\phi}$ using least squares methods:

$$\hat{r}_1 = \underset{r_1 \in \mathbf{Y}_d}{\operatorname{argmin}} \hat{\sigma}^2(r_1, d) \tag{18.12}$$

where $\hat{\sigma}^2(r_1, d)$ is the residual variance of the LS estimate of (18.8) given the threshold r_1 and the delay parameter d. So the threshold value r_1 can be estimated sequentially by searching over the possible values of r_1. If the delay parameter is not known, it can be estimated similarly by expanding the search to another dimension:

$$(\hat{r}_1, \hat{d}) = \underset{r_1, d}{\operatorname{argmin}} \hat{\sigma}^2(r_1, d) \tag{18.13}$$

One thing to note is that for the asymptotic inference on SETAR models to work correctly, each regime must have a non-trivial proportion of observations in the limit. Therefore, just as in computing Hansen's sup-LR test, a certain percentage of $\mathbf{Y}_d$ at both ends are usually trimmed and not used when searching for the value of r_1.

The `TAR` function in `S+FinMetrics` implements the above sequential estimation approach.[12] For example, to estimate a two-regime SETAR model with $d = 1$ and AR(2) components, use the following command:

```
> ndx.setar.r = TAR(log(ndx.rvol), p=2, d=1, trim.pct=0.1)
> ndx.setar.r

Call:
TAR(x = log(ndx.rvol), p = 2, d = 1, trim.pct = 0.1)

Coefficients:
          regime.1 regime.2
intercept -2.0356  -1.4614
     lag1  0.1903   0.2183
     lag2  0.2056   0.2435

Std. Errors of Residuals:
```

[12] As its name suggests, `TAR` function actually supports general TAR models, in addition to SETAR models. A general threshold variable can be used by specifying the optional argument `q`. In addition, `TAR` function also allows for the use of some popular functions of a variable as the threshold variable. See the online help file for `TAR` for details.

```
 regime.1 regime.2
 0.4233   0.3828

Information Criteria:
      logL       AIC       BIC        HQ
 -155.7369  323.4739  345.6966  332.3674

                   total regime.1 regime.2
Degree of freedom:   300      207       87
Time period: from 01/15/1996 to 10/08/2001
```

Note that the optional argument `trim.pct` is used to set the trimming percentage for $\mathbf{Y}_d$ to 10%. Compared with the three-regime SETAR fit in the previous subsection, this two-regime SETAR model actually gives a better fit in terms of log-likelihood value and BIC, which is probably due to the fact the threshold value is also optimized in this fit. The estimated threshold value is given as a component in the returned object `ndx.setar.r`:

```
> ndx.setar.r$qhat
[1] -2.876807
```

which is quite close to the first threshold identified using Tsay's t-statistics plot in the previous subsection.

Confidence Interval for the Threshold

Using the generic `summary` function on the fitted model object `ndx.setar.r` displays more details of the model fit:

```
> summary(ndx.setar.r)

Call:
TAR(x = log(ndx.rvol), p = 2, d = 1, trim.pct = 0.1)

Minimized SSE for all threshold variable candidates:
 RVOL.lag1
  49.84288

Threshold estimate for the threshold variable chosen with
smallest minimized SSE:
  CI.lower     point  CI.upper
 -3.826435 -2.876807 -2.828314

Coefficients and standard errors:
          regime.1   (se) regime.2   (se)
intercept -2.036    0.325 -1.461    0.372
```

```
     lag1  0.190    0.103  0.218    0.150
     lag2  0.206    0.073  0.244    0.099

Coefficient confidence intervals:
          regime.1.lower regime.1.upper
intercept -2.700         -1.075
     lag1 -0.020          0.417
     lag2  0.055          0.412

          regime.2.lower regime.2.upper
intercept -2.435         -0.454
lag1      -0.093          0.600
lag2      -0.003          0.472

Std. Errors of Residuals:
 regime.1 regime.2
 0.423    0.383

Information Criteria:
     logL      AIC      BIC       HQ
 -155.737  323.474  345.697  332.367

                   total regime.1 regime.2
Degree of freedom:   300      207       87
Time period: from 01/15/1996 to 10/08/2001
```

Note that standard inference statistics as well as confidence intervals for both the coefficients and the threshold are given. In particular, as proposed by Hansen (1997), an asymptotically valid confidence interval for the threshold is constructed by inverting the likelihood ratio test for testing the null hypothesis that the threshold is equal to a given value r:

$$LR(r) = n' \frac{\hat{\sigma}^2(r) - \hat{\sigma}^2(\hat{r}_1)}{\hat{\sigma}^2(\hat{r}_1)} \tag{18.14}$$

The $100 \cdot \alpha\%$ confidence interval for the threshold r_1 is given by the set of values of r for which the above LR test cannot be rejected at significance level $1 - \alpha$:

$$CI(\alpha) = \{r : \ LR(r) \leq \mathrm{Z}_\alpha\} \tag{18.15}$$

where Z_α is the $100 \cdot \alpha\%$ quantile of the asymptotic distribution of the LR statistic given in Hansen (1997). A graphical tool to help locate the confidence interval for the threshold is to plot the above LR statistics against different values of r, and choose the region of r close to r_1 where the LR statistics are smaller than the critical value Z_α. The necessary information to generate such a plot is contained in the `LR.q` component of the fitted

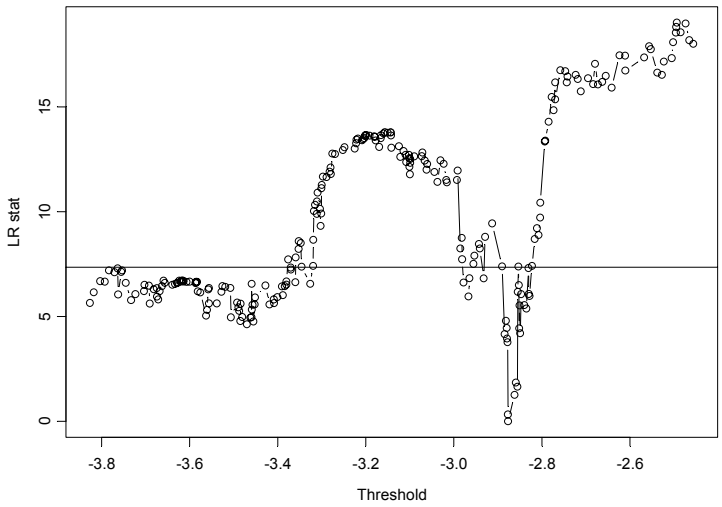

FIGURE 18.8. Confidence interval for threshold value by inverting likelihood ratio statistics.

model object. For example, to produce the plot using the fitted model object `ndx.setar.r`, use the following commands:

```
> names(ndx.setar.r$LR.q)
[1] "LR"        "Threshold" "Critical"
> plot(ndx.setar.r$LR.q$Threshold, ndx.setar.r$LR.q$LR,
+      type="b", xlab="Threshold", ylab="LR stat")
> abline(h=ndx.setar.r$LR.q$Critical)
```

and the plot is shown in Figure 18.8. This plot can also be generated directly and applying the generic `plot` function on the fitted model object `ndx.setar.r`.

Predictions From TAR Models

Just like with SETAR models, predictions from general TAR models can be computed using Monte Carlo simulations, as long as the future values of the threshold variable are known. In fact, the objects returned by the `TAR` function have class `"TAR"`, which inherits from the `"SETAR"` class. For example,

```
> class(ndx.setar.r)
[1] "TAR"
> inherits(ndx.setar.r, "SETAR")
```

```
[1] T
```

Thus, when the generic `predict` function is called on `"TAR"` objects, the simulation-based forecasting procedure in `predict.SETAR` is also used to produce the forecasts. For example, to generate forecasts from the fitted model object `ndx.setar.r`, use the following command:

```
> ndx.pred.2 = predict(ndx.setar.r, n.predict=100,
+ CI.alpha=0.6, n.sim=10000)
```

which are very similar to the forecasts produced earlier using a three-regime model.

18.4 Smooth Transition Autoregressive Models

In the TAR models introduced in the previous section, a regime switch happens when the threshold variable crosses a certain threshold. Although the model can capture many nonlinear features usually observed in economic and financial time series, sometimes it is counter-intuitive to suggest that the regime switch is abrupt or discontinuous. Instead, in some cases it is reasonable to assume that the regime switch happens gradually in a smooth fashion. If the discontinuity of the thresholds is replaced by a smooth transition function, TAR models can be generalized to smooth transition autoregressive (STAR) models.

In this section two main STAR models – logistic STAR and exponential STAR – are introduced. After illustrating how to test for STAR nonlinearity, examples will be given to show how to estimate STAR models in `S+FinMetrics`. A systematic modeling cycle approach for STAR models is proposed by Teräsvirta (1994), and van Dijk, Teräsvirta and Franses (2002) provides a survey of recent development for STAR models.

18.4.1 Logistic and Exponential STAR Models

In the SETAR model (18.8) considered in the previous section, the observations y_t are generated either from the first regime when y_{t-d} is smaller than the threshold, or from the second regime when y_{t-d} is greater than the threshold. If the binary indicator function is replaced by a smooth transition function $0 < G(z_t) < 1$ which depends on a *transition variable* z_t (like the threshold variable in TAR models), the model becomes a *smooth transition autoregressive* (STAR) model :

$$y_t = \mathbf{X}_t\boldsymbol{\phi}^{(1)}(1 - G(z_t)) + \mathbf{X}_t\boldsymbol{\phi}^{(2)}G(z_t) + \epsilon_t \tag{18.16}$$

Now the observations y_t switch between two regimes smoothly in the sense that the dynamics of y_t may be determined by both regimes, with one

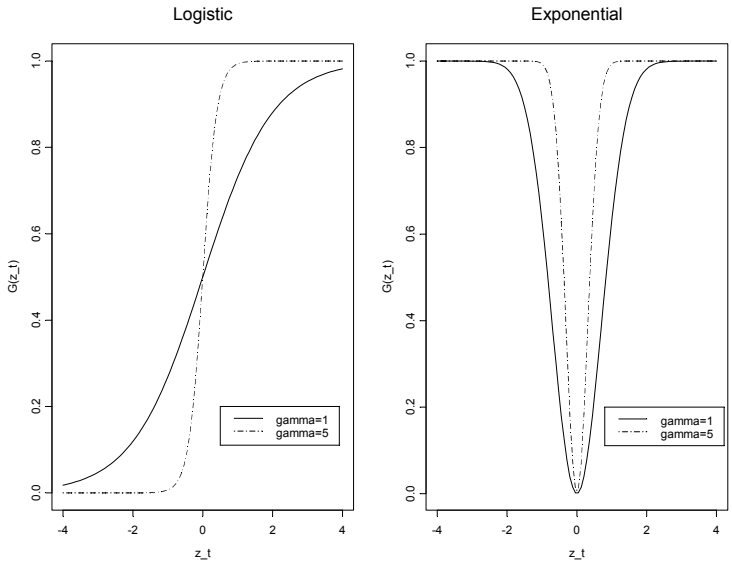

FIGURE 18.9. Logistic and exponential transition functions.

regime having more impacts in some times and the other regime having more impacts in other times. Another interpretation is that STAR models actually allow for a "continuum" of regimes, each associated with a different value of $G(z_t)$.

Two popular choices for the smooth transition function are the *logistic* function and the *exponential* function. Using the logistic function, the transition function can be specified as:

$$G(z_t;\gamma,c) = \frac{1}{1+e^{-\gamma(z_t-c)}},\ \gamma>0 \tag{18.17}$$

and the resulting model is referred to as *logistic STAR* or LSTAR model. The parameter c can be interpreted as the threshold, as in TAR models, and γ determines the speed and smoothness of transition. Using the exponential function, the transition function can be specified as:

$$G(z_t;\gamma,c) = 1-e^{-\gamma(z_t-c)^2},\ \gamma>0 \tag{18.18}$$

and the resulting model is referred to as *exponential STAR* or ESTAR model. As in LSTAR models, c can be interpreted as the threshold, and γ determines the speed and smoothness of transition.

In spite of the similarity between LSTAR and ESTAR models, they actually allow for different types of transitional behavior. To illustrate this point, Figure **??** plots the logistic and exponential transition functions with $c=0$ and $\gamma=1$ and 5. The following properties can be readily observed:

1. If γ is small, both transition functions switch between 0 and 1 very smoothly and slowly; if γ is large, both transition functions switch between 0 and 1 more quickly.

2. As $\gamma \to \infty$, both transition functions become binary. However, the logistic function approaches the indicator function $I(z_t > c)$ and the LSTAR model reduces to a TAR model; while the exponential function approaches the indicator function $I(z_t = c)$ and the model does not nest the TAR model as a special case.

3. The logistic function is monotonic and the LSTAR model switches between two regimes smoothly depending on how much the transition variable z_t is smaller than or greater than the threshold c. The exponential function is symmetrical and the ESTAR model switches between two regimes smoothly depending on how far the transition variable z_t is from the threshold c. For the LSTAR model, both the distance between z_t and c and its sign matter; for the ESTAR model, only the distance between z_t and c matters, but not the sign.

18.4.2 Test for STAR Nonlinearity

Testing for the existence of STAR-type nonlinearity is usually the first step toward building a STAR model. However, just like the test for threshold type nonlinearity, tests for the null hypothesis of a simple AR model against the alternative of a STAR model have non-standard asymptotic distributions, because some parameters in the STAR model are not identified under the null hypothesis, such as the AR coefficients $\boldsymbol{\phi}^{(2)}$ in the second regime, the transition parameter γ and the threshold c.

STAR Nonlinearity Test with Homoskedastic Errors

To avoid complicated issues caused by the unidentified STAR model parameters under the null hypothesis of a linear AR model, Luukkonen, Saikkonen and Teräsvirta (1988) propose to replace the transition function $G(z_t; \gamma, c)$ by a suitable Taylor series approximation around $\gamma = 0$. It turns out that if the transition function $G(z_t; \gamma, c)$ in the LSTAR model is replaced by its third order Taylor series approximation, the LSTAR model in (18.16) can be written as:[13]

$$y_t = \mathbf{X}_t\boldsymbol{\beta}_0 + \mathbf{X}_t z_t\boldsymbol{\beta}_1 + \mathbf{X}_t z_t^2\boldsymbol{\beta}_2 + \mathbf{X}_t z_t^3\boldsymbol{\beta}_3 + e_t \tag{18.19}$$

where the coefficient vectors $\boldsymbol{\beta}_i$ for $i = 0, 1, 2, 3, 4$ are functions of original model parameter $\boldsymbol{\phi}$. Similarly, if the transition function $G(z_t; \gamma, c)$ in the

[13] See Franses and van Dijk (2002) for details.

ESTAR model is replaced by its second order Taylor series approximation, the ESTAR model in (18.16) can be written as:

$$y_t = \mathbf{X}_t\boldsymbol{\beta}_0 + \mathbf{X}_t z_t\boldsymbol{\beta}_1 + \mathbf{X}_t z_t^2\boldsymbol{\beta}_2 + \mathbf{X}_t z_t^3\boldsymbol{\beta}_3 + \mathbf{X}_t z_t^4\boldsymbol{\beta}_4 + e_t \tag{18.20}$$

Now testing the null hypothesis of a linear AR model against a nonlinear STAR model is equivalent to testing the null hypothesis $H_0 : \boldsymbol{\beta}_j = 0$ for $j = 1, 2, 3, 4$ in the above auxiliary regressions, which is a conventional Lagrange multiplier (LM) test with an asymptotic χ^2 distribution.

In practice, it has been found that the LM test based on (18.19) for LSTAR models also has power against ESTAR alternatives. Thus, for reasons of parsimony, usually only the LM test based on (18.19) is computed for testing STAR-type nonlinearity in general. Also, instead of using the asymptotic χ^2 distribution, in small samples it is usually preferred to use the F version of the LM test which tends to have better size and power properties. Finally, since TAR models are special cases of LSTAR models when the transition parameter $\gamma \to \infty$, it can be shown that the LM test also has power against threshold type nonlinearity. Granger and Teräsvirta (1993) discuss these issues in more details.

The LM test for STAR nonlinearity can be performed in `S+FinMetrics` using the `nonlinearTest` function, by setting the optional argument `method` to `"STAR-LM"`. For example, to test for STAR-type nonlinearity in NASDAQ realized volatility `ndx.rvol`, use the command:

```
> nonlinearTest(log(ndx.rvol), method="STAR-LM", p=2, d=1:2)

Nonlinearity Test: STAR Nonlinearity

Null Hypothesis: no smooth threshold nonlinearity
Under Maintained Assumption of Homoskedastic Errors --

           ChiSq-stat ChiSq-dof ChiSq.pv-val
RVOL.lag1    21.3008         6       0.0016
RVOL.lag2    13.6974         6       0.0332

             F-stat   F-dof F.pv-val
RVOL.lag1    3.7068 (6,291)   0.0014
RVOL.lag2    2.3204 (6,291)  0.0333
```

In the above example, the transition variable is set to y_{t-d} by specifying the optional argument `d`.[14] More than one value of `d` can be specified and `nonlinearTest` automatically computes the LM test for all the given values of `d`. If the null hypothesis of a linear AR model is rejected, the test

[14] A weakly exogenous variable can also be used as the transition variable by setting the optional argument `q` instead of `d`. See the online help file for `nonlinearTest` for details.

statistics based on different values of `d` can be used to choose the appropriate value of d in the final STAR model. In the output shown above, the null hypothesis of no STAR-type nonlinearity is rejected at 5% significance level for both $d = 1$ and $d = 2$. In addition, the p-values from both the χ^2 test and F test prefer $d = 1$, which is consistent with the results of threshold-type nonlinearity tests presented in the previous section.

STAR Nonlinearity Test with Heteroskedastic Errors

The LM test presented above assumes the error term in (18.16) has constant variance. However, economic and financial time series are often heteroskedastic, and neglected heteroskedasticity may lead to spurious rejection of the null hypothesis. Based on Davidson and MacKinnon (1985), Granger and Teräsvirta (1993) summarize the following LM test for nonlinearity which is robust toward heteroskedastic errors:

1. Regress y_t on $\mathbf{X}_t$ to obtain the OLS residuals $\hat{\epsilon}_t$.
2. Regress $\mathbf{X}_t z_t^j$ for $j = 1, 2, 3$ on $\mathbf{X}_t$ to obtain the residuals $\hat{\mathbf{R}}_t$.
3. Regress the unit vector on $\hat{\mathbf{R}}_t \hat{\epsilon}_t$ and compute the LM statistic as the explained sum of squares from this regression.

This test can be performed just as before by setting the optional argument `hetero` to `TRUE`:

```
> nonlinearTest(log(ndx.rvol), method="STAR-LM", p=2, d=1:2,
+ hetero=T)

Nonlinearity Test: STAR Nonlinearity

Null Hypothesis: no smooth threshold nonlinearity
Allowing Heteroskedastic Errors using White Correction --

          ChiSq-stat ChiSq-dof ChiSq.pv-val
RVOL.lag1    15.0731         6       0.0197
RVOL.lag2    10.8287         6       0.0938

             F-stat   F-dof F.pv-val
RVOL.lag1    2.5657 (6,291)   0.0195
RVOL.lag2    1.8162 (6,291)   0.0957
```

Now the null hypothesis cannot be rejected at 5% significance level when $d = 2$, but it is still rejected at 5% level when $d = 1$. However, based on some simulation evidence, Lundbergh and Teräsvirta (1998) suggest that in some cases this robustification may not be desirable because it removes most of the power of the test.

18.4.3 Estimation of STAR Models

After confirming the existence of STAR-type nonlinearity in a time series, one can proceed to the next stage of building a STAR model. This usually involves choosing the transition variable and the form of transition function. As mentioned in the previous subsection, the test for STAR-type nonlinearity can be computed for a range of transition variables, and the p-values of the test statistics can be used to help choose the appropriate transition variable. The choice between the LSTAR model and the ESTAR model can usually be made by considering the specific transitional behavior under investigation, or by comparing different information criteria. This subsection first shows how to estimate LSTAR models using the `STAR` function in `S+FinMetrics`, and then walks through an example of estimating an ESTAR model using the `S-PLUS` function `nlregb`.

LSTAR Model

Once the AR order and the transition variable have been chosen, LSTAR models can be estimated by *nonlinear least squares* (NLS):

$$\hat{\boldsymbol{\Theta}} = \underset{\gamma,c}{\operatorname{argmin}} \sum_t \hat{\epsilon}_t^2 \tag{18.21}$$

where

$$\begin{aligned}
\hat{\epsilon}_t &= y_t - \tilde{\mathbf{X}}_t \hat{\boldsymbol{\phi}} \\
\tilde{\mathbf{X}}_t &= \begin{bmatrix} \mathbf{X}_t(1 - G(z_t;\gamma,c)) \\ \mathbf{X}_t G(z_t;\gamma,c) \end{bmatrix} \\
\hat{\boldsymbol{\phi}} &= \begin{bmatrix} \hat{\boldsymbol{\phi}}^{(1)} \\ \hat{\boldsymbol{\phi}}^{(2)} \end{bmatrix} = \left[\sum_t (\tilde{\mathbf{X}}_t' \tilde{\mathbf{X}}_t)\right]^{-1} \left[\sum_t \tilde{\mathbf{X}}_t' y_t\right]
\end{aligned}$$

Note that the minimization of the NLS objective function is only performed over γ and c because $\hat{\boldsymbol{\phi}}^{(1)}$ and $\hat{\boldsymbol{\phi}}^{(2)}$ can be estimated by least squares once γ and c are known. Under the additional assumption that the errors are normally distributed, NLS is equivalent to maximum likelihood estimation. Otherwise, the NLS estimates can be interpreted as quasi maximum likelihood estimates.

Example 120 *LSTAR model for NASDAQ realized volatility*

The following command fits an LSTAR model to the logarithms of weekly realized volatility of NASDAQ 100 index, with the same AR order and delay parameter used in the previous examples:

```
> ndx.lstar = STAR(log(ndx.rvol), p=2, d=1)
> summary(ndx.lstar)
```

```
Call:
STAR(X = log(ndx.rvol), p = 2, d = 1)

Parameter estimates:
          Values Std.Error Std.Error.white
    gamma  1.608  1.113     1.282
threshold -2.845  0.398     0.309


Coefficient estimates and standard errors:

Lower regime:
                 Values Std.Error Std.Error.white
intercept(lower) -3.729  1.832     2.696
     lag1(lower) -0.221  0.404     0.632
     lag2(lower)  0.205  0.092     0.092

Upper regime:
                 Values Std.Error Std.Error.white
intercept(upper) -2.668  1.904     1.497
     lag1(upper) -0.396  1.076     0.896
     lag2(upper)  0.216  0.134     0.131

Std. Errors of Residuals:
[1] 0.415

Information Criteria:
     logL      AIC      BIC       HQ
 -158.863  329.727  351.950  338.620

Degrees of freedom:
 total residuals
   300       294
Time period: from 01/15/1996 to 10/08/2001
```

Note that the threshold estimate -2.85 is very close to the SETAR estimate of -2.88 given by the TAR estimate `ndx.setar.r`. However, by allowing for smooth transition between two regimes, the AR coefficients in both regimes are quite different from those estimated by `ndx.setar.r`.

Predictions from LSTAR Model

Simulation based forecasts from the LSTAR model can be easily generated using the same principle for generating forecasts from VAR models and SETAR models. The fitted model objects returned by the `STAR` function

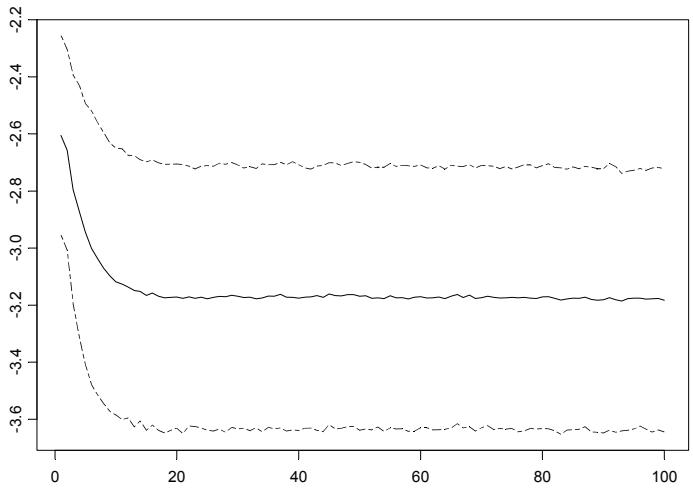

FIGURE 18.10. Predicted realized volatility (in logarithmic scale) from `ndx.lstar`.

have class `"STAR"`. By calling the generic `predict` function on fitted model objects, the method function `predict.STAR` is automatically invoked. For example, the following command generates 100-step-ahead forecasts from `ndx.lstar`:

```
> ndx.pred.3 = predict(ndx.lstar, n.predict=100,
+ CI.alpha=0.6, n.sim=10000)
> tsplot(cbind(ndx.pred.3$values, ndx.pred.3$CI),
+ lty=c(1,6,6))
```

and Figure 18.10 shows the forecasts with 60% pointwise confidence intervals. The forecasts are very similar to those generated by the SETAR model object `ndx.setar`, except they do not have the initial small peak exhibited by the SETAR forecasts.

ESTAR Model

Currently the `STAR` function in `S+FinMetrics` only supports LSTAR models but not ESTAR models. However the estimation of ESTAR models follows essentially the same procedure in (18.21) with the transition function given by (18.18). Here an example is given to show how to estimate ESTAR models using the `S-Plus` function `nlregb` for nonlinear least squares estimation.

The arguments expected by `nlregb` are as follows:

```
> args(nlregb)
function(nres, start, residuals, jacobian=NULL, scale=NULL,
control = NULL, lower = -Inf, upper = Inf, ...)
```

where the first argument `nres` specifies the number of observations or residuals to be used, the second argument `start` specifies the starting values for the unknown parameters, and the third argument `residuals` is an `S-PLUS` function which takes the parameter values and computes the residual vector with length equal to `nres`. The optional arguments `lower` and `upper` can be used to specify lower and upper bounds on the unknown parameters.

One general issue in estimating STAR models is that the transition parameter γ can get large and cause numerical problems in the optimization procedure. To alleviate the potential numerical problems in estimating ESTAR models, it is usually preferred to estimate the following transition function instead of the original exponential function in (18.18):

$$G(z_t;\tilde{\gamma},c) = 1 - \exp\{-e^{\tilde{\gamma}}\frac{(z_t - c)^2}{\sigma_z^2}\} \tag{18.22}$$

where σ_z^2 is the sample variance of the transition variable z_t. The new parameter $\tilde{\gamma}$ can be transformed to the original parameter γ as follows:

$$\gamma = \frac{e^{\tilde{\gamma}}}{\sigma_z^2} \tag{18.23}$$

This transformation has the following numerical properties:

1. The squared distance between z_t and the threshold c is now scaled by the variance of z_t which makes it scale-free.
2. The original parameter γ lies in $(0,\infty)$ which requires a constrained optimization in terms of γ. The new parameter $\tilde{\gamma}$ lies in $(-\infty,\infty)$ and is unconstrained.
3. The new parameter $\tilde{\gamma}$ is a linear function of the logarithm of γ which is more dampened than γ.

Using the new formulation in (18.22), the following `S-PLUS` function takes the unknown parameter values $(\tilde{\gamma},c)$ and returns the residual vector:

```
ESTAR.res = function(theta, g.scale, x, y, q)
{
  k = ncol(x)
  G = 1 - exp( - exp(theta[1])/g.scale * (q - theta[2])^2)
  X = cbind(x * (1 - G), x * G)
  m = crossprod(t(backsolve(chol(crossprod(X)), diag(2 * k))))
  beta = m %*% t(X) %*% y
  y - X %*% beta
}
```

Now to estimate an ESTAR model with an AR(2) specification and transition variable $z_t = y_{t-1}$ using the NASDAQ realized volatility series, use the following commands:

```
> ndx.LHS = log(ndx.rvol)[3:length(ndx.rvol)]@data
> ndx.RHS = cbind(1, tslag(log(ndx.rvol), 1:2, trim=T)@data)
> ndx.estar = nlregb(length(ndx.rvol)-2,
+    start=c(0,mean(ndx.RHS[,2])),
+    residuals=ESTAR.res,
+    lower=c(-Inf, min(ndx.RHS[,2])),
+    upper=c( Inf, max(ndx.RHS[,2])),
+    g.scale=var(ndx.RHS[,2]),
+    x=ndx.RHS, y=ndx.LHS, q=ndx.RHS[,2]))
```

Note that the regressors `ndx.RHS` include a constant term and two lagged values of y_t, and the transition variable y_{t-1} is given by the second column of `ndx.RHS`. In the call to the `nlregb` function, the starting values of $\tilde{\gamma}$ is set to zero, which corresponds to setting $\gamma = 1$, and the starting value of c is simply set to the mean of the transition variable y_{t-1}. Other arguments `g.scale`, `x`, `y` and `q` to the residual function `ESTAR.res` are passed as optional arguments to `nlregb`. The NLS estimates of $(\tilde{\gamma}, c)$ are given by:

```
> ndx.estar$parameters
[1] -1.239878 -2.774638
```

Note that the threshold estimate of -2.77 is close to the threshold estimates obtained in earlier examples. The transition parameter γ in the original exponential function can be obtained as follows:

```
> exp(ndx.estar$parameters[1])/var(ndx.RHS[,2])
[1] 1.013556
```

18.5 Markov Switching State Space Models

The nonlinear time series models introduced so far all allow for different regimes, with each regime represented by a simple AR model. For TAR and SETAR models, the regimes are solely determined by the magnitude of an *observable* weakly exogenous variable, while for STAR models the regimes are allowed to switch smoothly according to the magnitude of a weakly exogenous variable relative to a threshold value. This section introduces another type of regime switching model – the Markov switching model – where the regimes are determined by an *unobserved* state or regime variable that follows a discrete state Markov process. Discrete state Markov processes, also called Markov chains, are very popular choices for modeling state-dependent behavior. Since Hamilton (1989) proposed to use a simple Markov switching AR process to model the U.S. real GNP, Markov

switching time series models have seen extraordinary growth and become extremely popular for modeling economic and financial time series. They have been applied to model and forecast business cycles, the term structure of interest rates, volatility in economic and financial variables, foreign exchange rate dynamics, inflation rate dynamics, etc.

This section first introduces the discrete state Markov process which is used to model the hidden state variable, then illustrates how the discrete state Markov process can be combined with an AR model to produce the Markov switching AR process. To allow for Markov switching dynamics in a much broader context, Markov switching state space models are then introduced and examples will be given to illustrate the estimation of these models using `S+FinMetrics` functions.

18.5.1 Discrete State Markov Process

Discrete state Markov processes are very popular choices for modeling state-dependent behavior in natural phenomena, and are natural candidates for modeling the hidden state variables in Markov switching models. A discrete state Markov process classifies the state of the world S_t at any time t into a few discrete regimes. The state switches between different regimes according to its previous value and transition probabilities given by:[15]

$$\Pr(S_t = j|S_{t-1} = i) = P_{ij} \geq 0 \tag{18.24}$$

where $i, j = 1, 2, \cdots, k$ with k different possible states or regimes, and

$$\sum_{j=1}^{k} \Pr(S_t = j|S_{t-1} = i) = 1 \tag{18.25}$$

It is usually convenient to collect the transition probabilities into a *transition matrix*:

$$\mathcal{P} = \begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1k} \\ P_{21} & P_{22} & \cdots & P_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ P_{k1} & P_{k2} & \cdots & P_{kk} \end{bmatrix}$$

where each row sums up to one. For example, at time t the state of the economy S_t can be classified as either recessionary ($S_t = 1$) or expansionary ($S_t = 2$). Using quarterly observations of the U.S. real GNP from 1952 to

[15] A discrete state Markov process which only depends on its most recent observation is called the first order Markov process. Since higher order Markov processes can always be rewritten as a first order Markov process, it usually suffices to consider only the first order Markov process.

1984, Kim (1994) estimates the transition matrix to be:

$$\mathcal{P} = \begin{bmatrix} 47\% & 53\% \\ 5\% & 95\% \end{bmatrix} \tag{18.26}$$

These transition probabilities imply that if the economy is in an expansion, it tends to stay in expansion with a very high probability of 95%; if the economy is in a recession, it has 47% chance of staying in recession and 53% chance of getting out of recession. These numbers also reflect the common observation that the transition from an expansion to a recession is usually very quick, whereas the recovery from a recession is relatively slow.

Suppose at time t the probability of each state or regime is given by the vector $\boldsymbol{\pi}_t = (P_1, P_2, \cdots, P_k)$, then the probability of each state at time $t+1$ is given by:

$$\boldsymbol{\pi}_{t+1} = \mathcal{P}'\boldsymbol{\pi}_t \tag{18.27}$$

For a stationary discrete state Markov process, the *ergodic probability* vector $\boldsymbol{\pi}$ exists such that

$$\boldsymbol{\pi} = \mathcal{P}'\boldsymbol{\pi} \tag{18.28}$$

The ergodic probability vector can also be treated as the *steady state*, or the *unconditional* probability of each state of the world. `S+FinMetrics` provides a convenience function `mchain.p0` to compute the ergodic probability vector for a stationary Markov chain.[16] For example, the following command computes the ergodic probabilities for the state of the economy using the transition matrix in (18.26):

```
> mchain.p0(matrix(c(0.47, 0.05, 0.53, 0.95), 2, 2))
[1] 0.0862069 0.9137931
```

So the unconditional probability of the economy being in a recession is about 9%, and the unconditional probability of the economy being in an expansion is about 91%.

The transition probabilities can also be used to infer the duration of each state or regime. For example, using the transition matrix in (18.26), the average duration of an economic expansion can be computed as:[17]

$$\frac{1}{1 - P_{22}} = 20 \text{ quarters} = 5 \text{ years}$$

and the average duration of an economic recession can be computed as:

$$\frac{1}{1 - P_{11}} = 2 \text{ quarters}$$

which is consistent with the fact that a recession is usually defined as a drop in real GDP for two consecutive quarters.

---

[16] See Hamilton (1994) for the analytic formula for computing the ergodic probabilities for a stationary Markov chain.

[17] See Kim and Nelson (1999), for example, for the derivation of this result.

18.5.2 Markov Switching AR Process

If the model parameters in the simple AR(p) model in (18.4) are relaxed to be dependent on a *latent* or hidden state variable S_t, it becomes:

$$y_t = \mu_{S_t} + \mathbf{X}_t \boldsymbol{\phi}_{S_t} + u_t \text{ for } t = 1, 2, \cdots, n \tag{18.29}$$

where $\mathbf{X}_t = (y_{t-1}, y_{t-2}, \cdots, y_{t-p})$, $\boldsymbol{\phi}_{S_t}$ is the $p \times 1$ vector of AR coefficients, $u_t \sim N(0, \sigma^2_{S_t})$, and the hidden state variable S_t follows a k-regime Markov chain given by (18.24)-(18.25). This is usually referred to as the *Markov switching* AR(p) *process*. The Markov switching AR(p) model has proved to be effective at modeling nonlinear dynamics usually observed in economic and financial time series. For example, Hamilton (1989) uses a two-state Markov switching AR(4) model with constant σ^2 to capture the different dynamics observed in the U.S. real GNP during economic recessions and expansions.

In general, if the states $\mathbf{S} = (S_{p+1}, \cdots, S_n)$ are known, the unknown parameters $\boldsymbol{\Theta}$ of the Markov switching AR(p) model, which include the intercept terms, the AR coefficients and the error variance in different regimes, can be estimated by maximizing the following log-likelihood function:

$$L(\boldsymbol{\Theta}|\mathbf{S}) = \sum_{t=p+1}^{n} \log f(y_t|\mathcal{Y}_{t-1}, S_t)$$

where $\mathcal{Y}_{t-1}$ denotes all the information available at time $t-1$ and includes all the observations in $\mathbf{X}_j$ for $j \leq t$, and

$$f(y_t|\mathcal{Y}_{t-1}, S_t) \propto \exp\{-\frac{1}{2}\log\sigma^2_{S_t} - \frac{(y_t - \mu_{S_t} - \mathbf{X}_t\boldsymbol{\phi}_{S_t})^2}{2\sigma^2_{S_t}}\} \tag{18.30}$$

However, the states $\mathbf{S}$ are usually unobserved and must be inferred from the data. When $\mathbf{S}$ are unknown, the parameters of the Markov switching AR(p) model are expanded to include the transition probabilities $\mathcal{P}$. By applying the law of total probability, the log-likelihood function can now be written as:

$$\begin{aligned} L(\boldsymbol{\Theta}) &= \sum_{t=p+1}^{n} \log f(y_t|\mathcal{Y}_{t-1}) \\ &= \sum_{t=p+1}^{n} \log \left\{ \sum_{j=1}^{k} f(y_t|\mathcal{Y}_{t-1}, S_t = j) \Pr(S_t = j|\mathcal{Y}_{t-1}) \right\} \end{aligned} \tag{18.31}$$

where $f(y_t|\mathcal{Y}_{t-1}, S_t = j)$ is given in (18.30), and by Bayes theorem the *predictive* probability $\Pr(S_t = j|\mathcal{Y}_{t-1})$ can be shown to be:

$$\Pr(S_t = j|\mathcal{Y}_{t-1}) = \sum_{i=1}^{k} \Pr(S_t = j|S_{t-1} = i, \mathcal{Y}_{t-1}) \Pr(S_{t-1} = i|\mathcal{Y}_{t-1})$$

$$= \sum_{i=1}^{k} P_{ij} \frac{f(y_{t-1}|\mathcal{Y}_{t-2}, S_{t-1} = i) \Pr(S_{t-1} = i|\mathcal{Y}_{t-2})}{\sum_{m=1}^{k} f(y_{t-1}|\mathcal{Y}_{t-2}, S_{t-1} = m) \Pr(S_{t-1} = m|\mathcal{Y}_{t-2})} \tag{18.32}$$

So given an estimate of the initial probability of each state $\Pr(S_{p+1} = i|\mathcal{Y}_p)$ for $i = 1, 2, \cdots, k$, the log-likelihood function of the Markov switching AR(p) model can be computed iteratively using (18.31)-(18.32) and the unknown parameters $\boldsymbol{\Theta}$ can be estimated by maximum likelihood estimation (MLE).

The evaluation of the above log-likelihood function for the Markov switching AR(p) model can be easily programmed in `S-PLUS`. However, since it involves an iterative process which prevents the use of vectorized operations in `S-PLUS`, the optimization process of obtaining the MLE can be slow and computationally inefficient. In order to be able to estimate a broad range of Markov switching models using the same code, the following subsection introduces Markov switching state space models which includes the Markov switching AR(p) model as a special case. The Markov switching state space models utilize optimized C code for fast calculation.

18.5.3 Markov Switching State Space Models

As shown in Chapter 14, most linear time series regression models can be cast into a state space form, and the state space representation provides a convenient framework for obtaining filtered and smoothed estimates of the unobserved state variables. In this subsection the state space representation in Chapter 14 is generalized to allow for Markov switching dynamics so that a vast number of Markov switching models can be easily estimated using the same framework.

Using the notation in Chapter 14, a state space model can be represented as follows:

$$\boldsymbol{\alpha}_{t+1} = \mathbf{d}_t + \mathbf{T}_t \cdot \boldsymbol{\alpha}_t + \mathbf{H}_t \cdot \boldsymbol{\eta}_t \tag{18.33}$$

$$\mathbf{y}_t = \mathbf{c}_t + \mathbf{Z}_t \cdot \boldsymbol{\alpha}_t + \mathbf{G}_t \cdot \boldsymbol{\varepsilon}_t \tag{18.34}$$

where $\boldsymbol{\alpha}_{t+1}$ is the $m \times 1$ state vector, $\mathbf{y}_t$ is the $N \times 1$ vector of observed variables, $\boldsymbol{\eta}_t \sim iid\ N(0, \mathbf{I}_r)$ is the $r \times 1$ vector of disturbance terms in the *transition equation* governing the dynamics of the state vector $\boldsymbol{\alpha}_{t+1}$, $\boldsymbol{\varepsilon}_t \sim iid\ N(\mathbf{0}, \mathbf{I}_N)$ is the $N \times 1$ vector of disturbance terms in the *measurement equation* governing the dynamics of the observed variables $\mathbf{y}_t$, and $\mathbf{d}_t$, $\mathbf{T}_t$,

$\mathbf{H}_t$, $\mathbf{c}_t$, $\mathbf{Z}_t$ and $\mathbf{G}_t$ are conformable *hyperparameter matrices* or *system matrices*. More compactly, the above representation can be rewritten as:

$$\begin{pmatrix} \boldsymbol{\alpha}_{t+1} \\ \mathbf{y}_t \end{pmatrix} = \boldsymbol{\delta}_t + \boldsymbol{\Phi}_t \cdot \boldsymbol{\alpha}_t + \mathbf{u}_t, \tag{18.35}$$

where $\mathbf{u}_t \sim iid\ N(\mathbf{0}, \boldsymbol{\Omega}_t)$ and

$$\boldsymbol{\delta}_t = \begin{pmatrix} \mathbf{d}_t \\ \mathbf{c}_t \end{pmatrix}, \boldsymbol{\Phi}_t = \begin{pmatrix} \mathbf{T}_t \\ \mathbf{Z}_t \end{pmatrix}, \mathbf{u}_t = \begin{pmatrix} \mathbf{H}_t\boldsymbol{\eta}_t \\ \mathbf{G}_t\boldsymbol{\varepsilon}_t \end{pmatrix}, \boldsymbol{\Omega}_t = \begin{pmatrix} \mathbf{H}_t\mathbf{H}_t' & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_t\mathbf{G}_t' \end{pmatrix}$$

For *Markov switching state space models*, the hyperparameter matrices are assumed to be dependent on a *latent* or unobserved discrete state variable S_t:

$$\begin{aligned} \boldsymbol{\delta}_t &= \boldsymbol{\delta}_{S_t} \\ \boldsymbol{\Phi}_t &= \boldsymbol{\Phi}_{S_t} \\ \boldsymbol{\Omega}_t &= \boldsymbol{\Omega}_{S_t} \end{aligned}$$

and the discrete state variable S_t follows a k-regime Markov chain given in (18.24)-(18.25). For example, by setting the continuous state vector to $\boldsymbol{\alpha}_t = (y_t, y_{t-1})$, the Markov switching AR(2) model can be put into the above state space representation with:

$$\boldsymbol{\delta}_t = \begin{bmatrix} \mu_{S_{t+1}} \\ 0 \\ 0 \end{bmatrix},\ \boldsymbol{\Phi}_t = \begin{bmatrix} \boldsymbol{\phi}'_{S_{t+1}} \\ \mathbf{I}_{2\times 2} \end{bmatrix},\ \mathbf{I}_{2\times 2} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \end{bmatrix}$$

and $\boldsymbol{\Omega}_t$ is a 3×3 matrix with $\sigma^2_{S_{t+1}}$ being the $(1,1)$ element and zero elsewhere.

Example 121 *State space representation of Markov switching AR(2) model*

`S+FinMetrics` uses a `"list"` object with some required components to represent a state space model in `S-PLUS`, and Chapter 14 has many examples showing how to create such objects for some popular time series regression models. In order for Markov switching state space models to be represented by an `S-PLUS` object, the `"list"` object is expanded to allow for the following components: `mTrans`, `mDelta.other`, `mPhi.other` and `mOmega.other`. The `mTrans` component is required for a Markov switching state space representation and specifies the transition matrix $\mathcal{P}$ for the underlying Markov chain, and at least one of `mDelta.other`, `mPhi.other` and `mOmega.other` must be specified so that at least some hyperparameter of the model is Markov switching. The usual components `mDelta`, `mPhi` and `mOmega` specify the hyperparameter matrices for the first regime, and the new components `mDelta.other`, `mPhi.other` and `mOmega.other` specify the hyperparameter matrices for other regimes if necessary. If there

are $k > 2$ regimes for the discrete state variable S_t, the components `mDelta.other`, `mPhi.other` and `mOmega.other` store the hyperparameter matrices for regimes 2 to k stacked column-wise.

For example, the unknown parameters of a two-regime Markov switching AR(2) model can be collected in the vector:

$$v = (\mu_1, \mu_2, \phi_{11}, \phi_{12}, \phi_{21}, \phi_{22}, \sigma_1, \sigma_2, P_{11}, P_{22}) \tag{18.36}$$

where μ_1, ϕ_{11}, ϕ_{12} and σ_1 are the intercept term, the AR coefficients and error standard deviation for the first regime; μ_2, ϕ_{21}, ϕ_{22} and σ_2 are the counterparts for the second regime; P_{11} and P_{22} are the diagonal elements of the transition matrix $\mathcal{P}$. Note that since each row of $\mathcal{P}$ sums up to one, only two transition probabilities are required to identify $\mathcal{P}$. The following `S-PLUS` function takes the vector (18.36) and returns a `"list"` object giving the state space representation of the two-regime Markov switching AR(2) model:

```
GetSsfMSAR = function(parm)
{
  mDelta = mDelta.other = rep(0, 3)
  mDelta[1] = parm[1]
  mDelta.other[1] = parm[2]
#
  mPhi = mPhi.other = matrix(0, 3, 2)
  mPhi[1,] = c(parm[3], parm[4])
  mPhi.other[1,] = c(parm[5], parm[6])
  mPhi[2:3,1] = mPhi.other[2:3,1] = 1
#
  mOmega = mOmega.other = matrix(0, 3, 3)
  mOmega[1,1] = parm[7]
  mOmega.other[1,1] = parm[8]
#
  mSigma = matrix(0, 3, 2)
  mSigma[1:2, 1:2] = diag(1e+6, 2)
#
  mTrans = matrix(0, 2, 2)
  mTrans[1,1] = parm[9]
  mTrans[1,2] = 1 - mTrans[1,1]
  mTrans[2,2] = parm[10]
  mTrans[2,1] = 1 - mTrans[2,2]
#
  list(mDelta=mDelta, mDelta.other=mDelta.other,
       mPhi=mPhi, mPhi.other=mPhi.other,
       mOmega=mOmega, mOmega.other=mOmega.other,
       mSigma=mSigma, mTrans=mTrans)
}
```

Note that a diffuse prior on the initial state vector is specified by setting the first 2×2 block of `mSigma` to a diagonal matrix with large values on the diagonal, and the last row of `mSigma` to zero.

Approximate MLE of Markov Switching State Space Models

Since Markov switching state space models allow for nonlinear dynamics, the traditional Kalman filtering and smoothing algorithms for Gaussian linear state space models can no longer be applied to obtain valid inference on the unobserved state vector. In particular, given the initial estimate $\mathbf{a}_{t|t}^{(i)}$ and $\mathbf{P}_{t|t}^{(i)}$ for $S_t = i$ with $i = 1, \cdots, k$, the prediction equations for the Gaussian linear state space model in (14.39)-(14.40) now become:

$$\mathbf{a}_{t+1|t}^{(i,j)} = \mathbf{T}_t \mathbf{a}_{t|t}^{(i)} \tag{18.37}$$

$$\mathbf{P}_{t+1|t}^{(i,j)} = \mathbf{T}_t \mathbf{P}_{t|t}^{(i)} \mathbf{T}_t' + \mathbf{H}_t \mathbf{H}_t' \tag{18.38}$$

where the superscript (i, j) denotes the case of $S_t = i$ and $S_{t+1} = j$ for $i, j = 1, \cdots, k$. The updating equations for the Gaussian linear state space model in (14.34)-(14.35) now become:

$$\mathbf{a}_{t|t}^{(i,j)} = \mathbf{a}_{t|t-1}^{(i,j)} + \mathbf{K}_t^{(i,j)} \mathbf{v}_t^{(i,j)} \tag{18.39}$$

$$\mathbf{P}_{t|t}^{(i,j)} = \mathbf{P}_{t|t-1}^{(i,j)} - \mathbf{P}_{t|t-1}^{(i,j)} \mathbf{Z}_t' (\mathbf{K}_t^{(i,j)})' \tag{18.40}$$

where

$$\begin{aligned}
\mathbf{v}_t^{(i,j)} &= \mathbf{y}_t - \mathbf{c}_t - \mathbf{Z}_t \mathbf{a}_{t|t-1}^{(i,j)} \\
\mathbf{F}_t^{(i,j)} &= \mathbf{Z}_t \mathbf{P}_{t|t-1}^{(i,j)} \mathbf{Z}_t' + \mathbf{G}_t \mathbf{G}_t' \\
\mathbf{K}_t^{(i,j)} &= \mathbf{P}_{t|t-1}^{(i,j)} \mathbf{Z}_t' (\mathbf{F}_t^{(i,j)})^{-1}
\end{aligned}$$

So at each step the set of statistics that needs to be computed and stored will increase by the order of k. Obviously, even for a relatively small sample, the Kalman filtering algorithm will become computationally infeasible.

To make the filtering algorithm manageable, Kim (1994) proposes to collapse the set of statistics in the updating equations (18.39)-(18.40) as follows:

$$\mathbf{a}_{t|t}^{(j)} = \frac{\sum_{i=1}^{k} \Pr(S_t = j, S_{t-1} = i | \mathcal{Y}_t) \mathbf{a}_{t|t}^{(i,j)}}{\Pr(S_t = j | \mathcal{Y}_t)} \tag{18.41}$$

$$\mathbf{P}_{t|t}^{(j)} = \frac{\sum_{i=1}^{k} \Pr(S_t = j, S_{t-1} = i | \mathcal{Y}_t) [\mathbf{P}_{t|t}^{(i,j)} + (\mathbf{a}_{t|t}^{(j)} - \mathbf{a}_{t|t}^{(i,j)})(\mathbf{a}_{t|t}^{(j)} - \mathbf{a}_{t|t}^{(i,j)})']}{\Pr(S_t = j | \mathcal{Y}_t)} \tag{18.42}$$

where the *filtered* probability $\Pr(S_t = j|\mathcal{Y}_t)$ can be updated similarly as in (18.32) given an initial estimate. Now at each step, only k sets of statistics need to be stored, which can be fed into the prediction equations (18.37)-(18.38) to complete the filtering algorithm. This algorithm is sometimes referred to as *Kim's filtering algorithm*.

Just like the Kalman filtering algorithm for Gaussian linear state space models, Kim's filtering algorithm can be used to provide the prediction error decomposition for computing the log-likelihood function of Markov switching state space models. However, the drawback of the above filtering algorithm is that the filtered estimates $\mathbf{a}_{t|t}^{(j)}$ now follow normal mixture distributions instead of normal distributions as in Gaussian linear state space models. As a result, the MLE estimates obtained using Kim's algorithm are only approximate and not optimal, but empirical evidence seems to suggest that MLE estimates obtained using Kim's filtering algorithm are very reliable.[18]

The `SsfLoglikeMS` function in `S+FinMetrics` implements Kim's filtering algorithm to compute the log-likelihood function for arbitrary Markov switching state space models, and the `SsfFitMS` function uses it to obtain approximate MLE estimates of the unknown parameters in Markov switching state space models. However, Markov switching state space models can be difficult to fit due to various numerical issues. Here, a few guidelines are provided for using the `SsfFitMS` function for MLE estimation of Markov switching state space models:

1. Make sure that the model to be fitted is actually identified. It can be very easy to specify a Markov switching model which is not identified or poorly identified. Over-identification or poor identification can cause the optimization procedure to fail.

2. Start from a small model. If the estimation of the small model does not pose any problem, extend the model to allow for more features.

3. Provide good starting values to `SsfFitMS`. Good starting values can be found by calling `SsfLoglikeMS` with different sets of parameter values, and choosing the one with largest log-likelihood value.

4. Although the `SsfFitMS` function allows lower and upper bound constraints on the parameters, sometimes better convergence can be obtained by transforming the parameters so that the parameters to be estimated are unconstrained.

[18] In recent years more computationally-intensive Bayesian methods have also been developed to analyze Markov switching state space models or non-Gaussian state space models on a case-by-case basis, for example, see Kim and Nelson (1998), Kim, Shephard and Chib (1998) and Aguilar and West (2000) for some examples.

Example 122 *Markov switching AR(2) model for NASDAQ realized volatility*

Earlier examples in this chapter show that the logarithms of weekly realized volatility of NASDAQ 100 index can be modeled by a switching AR(2) process, with the switching determined by either a TAR model or a STAR model. It is interesting to see if the Markov switching AR(2) model can provide a better or equivalent characterization of the nonlinear dynamics observed in the data.

Instead of directly estimating the unknown parameters for the Markov switching AR(2) model as given in (18.36), it is usually better to transform these parameters so that they are unconstrained. For example, the following monotonic transformations are usually adopted:

1. If x lies within $(0, \infty)$, then $y = \log x$ is unconstrained and $x = e^y$.
2. If x lies within $(0, 1)$, then $y = \log[x/(1-x)]$ is unconstrained and $x = 1/(1+e^{-y})$.
3. If x lies within $(-1, 1)$, then $y = \log[(1+x)/(1-x)]$ is unconstrained and $x = 2/(1+e^{-y}) - 1$.
4. For the AR(2) process $y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + u_t$ to be stationary, the roots z_1 and z_2 of the characteristic equation $z^2 - \phi_1 z - \phi_2 = 0$ must lie within the unit circle, with $z_1 + z_2 = \phi_1$ and $z_1 \cdot z_2 = -\phi_2$.

The following `S-PLUS` function modifies the `GetSsfMS` function given earlier in this subsection by employing the above transformations. It now takes an unconstrained parameter vector and returns the state space representation of Markov switching AR(2) model:

```
GetSsfMSAR2 = function(parm)
{
  parm = as.vector(parm)
#
  mDelta = mDelta.other = rep(0, 3)
  mDelta[1] = parm[1]
  mDelta.other[1] = parm[1] + exp(parm[2])
#
  AR11 = 2/(1+exp(-parm[3])) - 1
  AR12 = 2/(1+exp(-(parm[3]+exp(parm[4])))) - 1
  AR21 = 2/(1+exp(-parm[5])) - 1
  AR22 = 2/(1+exp(-(parm[5]+exp(parm[6])))) - 1
#
  mPhi = mPhi.other = matrix(0, 3, 2)
  mPhi[1,] = c(AR11+AR12, -AR11*AR12)
  mPhi.other[1,] = c(AR21+AR22, -AR21*AR22)
```

```
  mPhi[2:3,1] = mPhi.other[2:3,1] = 1
#
  mOmega = matrix(0, 3, 3)
  mOmega[1,1] = exp(parm[7])
#
  mSigma = matrix(0, 3, 2)
  mSigma[1:2, 1:2] = diag(1e+6, 2)
#
  mTrans = matrix(0, 2, 2)
  mTrans[1,2] = 1/(1+exp(-parm[8]))
  mTrans[1,1] = 1 - mTrans[1,2]
  mTrans[2,1] = 1/(1+exp(-parm[9]))
  mTrans[2,2] = 1 - mTrans[2,1]
#
  ssf = list(mDelta=mDelta, mDelta.other=mDelta.other,
        mPhi=mPhi, mPhi.other=mPhi.other, mOmega=mOmega,
        mTrans=mTrans, mSigma=mSigma)
  CheckSsf(ssf)
}
```

A few comments on the function `GetSsfMSAR2` are as follows:

1. The second parameter `parm[2]` is actually $\log(\mu_2 - \mu_1)$. By employing this transformation, μ_2 is guaranteed to be greater than μ_1, and thus the first regime can be identified as the low volatility regime and the second as the high volatility regime.

2. The fourth and sixth parameters `parm[4]` and `parm[6]` are actually the logarithmic difference between two characteristic roots of their respective AR(2) processes. By employing this transformation the first roots are identified as the smaller roots while the second as the larger ones.

3. Finally it is usually preferred to call the `CheckSsf` function before returning the list with the state space representation, which makes sure the returned list is a valid state space representation.

Now, to fit the Markov switching AR(2) model to `log(ndx.rvol)`, use the following commands:[19]

```
> ndx.start = c(-2, -0.7, -0.7, 0.7, -0.7, 0.7, -2, -2, -3)
> names(ndx.start) = c("mu1", "mu2", "phi11", "phi12",
```

[19] `S+FinMetrics` provides the function `MSAR` for estimating general Markov switching AR(p) processes. The `MSAR` function returns an `"MSAR"` object, and methods for many generic functions, such as `summary`, `plot`, `residuals`, `vcov` and `simulate`, are provided for `"MSAR"` objects. See the online help file for `MSAR` for details.

```
+ "phi21", "phi22", "sigma", "p", "q")
> ndx.msar = SsfFitMS(ndx.start, log(ndx.rvol), GetSsfMSAR2,
+ l.start=11)
Iteration  0 : objective =  0.5575044
Iteration  1 : objective =  0.9047186
Iteration  2 : objective =  0.555338
...
Iteration  98 : objective =  0.5161791
RELATIVE FUNCTION CONVERGENCE
```

Note that the first argument to `SsfFitMS` specifies the starting values, the second argument specifies the data to be used, and the third argument specifies the `S-PLUS` function which takes a vector of model parameters and returns a valid state space representation of a Markov switching model. Since the filtering algorithm is started with diffuse priors on the state vector, the optional argument `l.start` is used to start the log-likelihood function evaluation from the 11th observation, which allows the effects of diffuse priors on the state vector to dissipate before log-likelihood values are computed.

The returned object is a `"SsfFit"` object, and applying the generic `summary` function returns the standard errors of the estimated parameters and associated t-statistics:

```
> class(ndx.msar)
[1] "SsfFit"
> summary(ndx.msar)
Log-likelihood:  -150.724
302 observations
Parameters:
        Value Std. Error  t value
  mu1 -1.8670    0.27600  -6.7640
  mu2 -0.9385    1.08100  -0.8684
phi11 -0.3336    0.23730  -1.4060
phi12  0.4073    0.32060   1.2710
phi21 -0.8366    0.25960  -3.2230
phi22  0.8109    0.22670   3.5760
sigma -1.8310    0.08313 -22.0300
    p -5.3150    1.00900  -5.2670
    q -8.4870    6.00100  -1.4140

Convergence:  RELATIVE FUNCTION CONVERGENCE
```

From the above output, most of the parameters are significant according to the t-statistics. To transform the estimated parameters into the param-

eters for the Markov switching AR(2) model, simply call `GetSsfMSAR2` on the ML estimates[20]:

```
> ndx.ssf = GetSsfMSAR2(ndx.msar$parameters)
> cbind(ndx.ssf$mDelta, ndx.ssf$mDelta.other)
         [,1]      [,2]
[1,] -1.86719 -1.475965
[2,]  0.00000  0.000000
[3,]  0.00000  0.000000
> ndx.ssf$mPhi
          [,1]       [,2]
[1,] 0.3606984 0.08693354
[2,] 1.0000000 0.00000000
[3,] 1.0000000 0.00000000
> ndx.ssf$mPhi.other
          [,1]      [,2]
[1,] 0.2130623 0.2406814
[2,] 1.0000000 0.0000000
[3,] 1.0000000 0.0000000
> ndx.ssf$mOmega
          [,1] [,2] [,3]
[1,] 0.1601773    0    0
[2,] 0.0000000    0    0
[3,] 0.0000000    0    0
> ndx.ssf$mTrans
             [,1]         [,2]
[1,] 0.9951049274 0.004895073
[2,] 0.0002061726 0.999793827
```

Note that the intercept terms in both regimes and the AR coefficients in the high volatility regime are similar to those estimated by the SETAR model `ndx.setar.r` in Section 18.3. However, the AR coefficients in the low volatility regime are somewhat different from those estimated by `ndx.setar.r`. In addition, both the transition probabilities P_{11} and P_{22} are estimated to be very close to one, which suggests that once y_t is in a certain regime, it tends to stay in that regime.

Filtered and Smoothed Estimates of Regime Probabilities

Once the unknown parameters of Markov switching models are estimated, it is usually of interest to obtain the *filtered* estimates of the latent discrete state or regime probability $\Pr(S_t = j|\mathcal{Y}_t)$. However, this quantity is already computed by Kim's filtering algorithm and thus is a side product of the log-likelihood function evaluation. In addition, it is also of interest to obtain the

[20] Standard errors for these parameters may be obtained using the delta method.

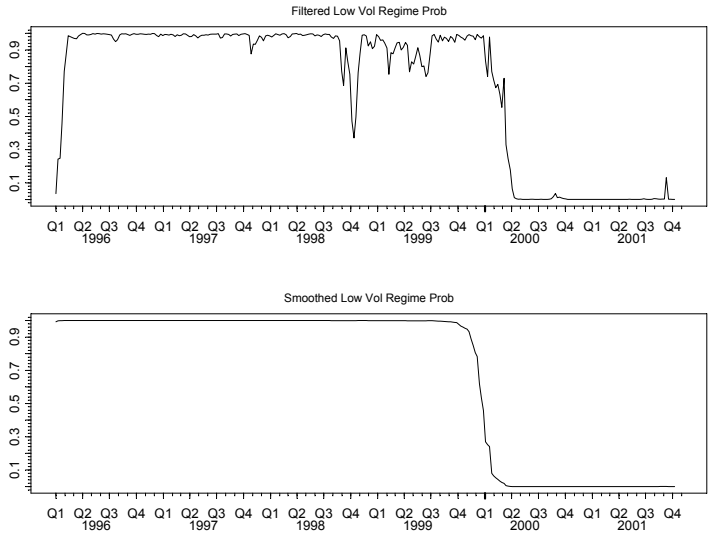

FIGURE 18.11. Filtered and smoothe regime probabilities of NASDAQ realized volatility.

smoothed estimates of the latent discrete state probability $\Pr(S_t = j|\mathcal{Y}_n)$, which is useful for retrospective analysis. To obtain the smoothed estimates $\Pr(S_t = j|\mathcal{Y}_n)$, note that at time n:

$$\begin{aligned}
\Pr(S_n = j, S_{n-1} = i|\mathcal{Y}_n) &= \Pr(S_n = j|I_n)\Pr(S_{n-1} = i|S_n = j, \mathcal{Y}_n) \\
&\approx \Pr(S_n = j|\mathcal{Y}_n)\Pr(S_{n-1} = i|S_n = j, \mathcal{Y}_{n-1}) \\
&= \frac{\Pr(S_n = j|\mathcal{Y}_n)\Pr(S_{n-1} = i, S_n = j|\mathcal{Y}_{n-1})}{\Pr(S_n = j|\mathcal{Y}_{n-1})} \\
&= \frac{\Pr(S_n = j|\mathcal{Y}_n)\Pr(S_{n-1} = i|\mathcal{Y}_{n-1})\Pr(S_n = j|S_{n-1} = i)}{\Pr(S_n = j|\mathcal{Y}_{n-1})}
\end{aligned}$$

and thus the smoothed estimate $\Pr(S_{n-1} = j|\mathcal{Y}_n)$ can be obtained as:

$$\Pr(S_{n-1} = j|\mathcal{Y}_n) = \sum_{i=1}^{k} \Pr(S_n = j, S_{n-1} = i|\mathcal{Y}_n)$$

This procedure can be repeated iteratively backwards from time $n-1$ to time 1 to obtain the smoothed estimates of regime probabilities.

In `S+FinMetrics` the filtered and smoothed regime probabilities can be obtained using the `SsfLoglikeMS` function with the optional argument `save.regm` set to `TRUE`. For example, the following commands plot the

filtered and smoothed estimates of regime probabilities based on the fit `ndx.msar`:

```
> ndx.f = SsfLoglikeMS(log(ndx.rvol), ndx.ssf, save.rgm=T)
> par(mfrow=c(2,1))
> plot(timeSeries(ndx.f$regimes[,1], pos=positions(ndx.rvol)),
+ reference.grid=F, main="Filtered Low Vol Regime Prob")
> plot(timeSeries(ndx.f$regimes[,3], pos=positions(ndx.rvol)),
+ reference.grid=F, main="Smoothed Low Vol Regime Prob")
```

and the plot is shown in Figure 18.11. The smoothed regime probabilities suggest that there is actually an abrupt switch around the first quarter of 2000.

18.6 An Extended Example: Markov Switching Coincident Index

The United States Department of Commerce periodically publishes the *Index of Coincident Economic Indicators* (CEI) based on four macroeconomic coincident variables, which provides a composite measure of the general state of the economy. The method used for the construction of the coincident index is *ad doc*, and the coincident index is subject to revisions after it is published. To provide a systematic probabilistic model for building an alternative coincident index, Stock and Watson (1991) have developed a dynamic factor model using a state space representation that models the coincident index as a common factor driving the four macroeconomic coincident variables: industrial production (IP), total personal income less transfer payments (PI), total manufacturing and trade sales (Sales) and employees on nonagricultural payrolls (Payroll). Stock and Watson (1991) show that their probabilistic coincident index matches very well with the Index of CEI compiled by the Department of Commerce.

Stock and Watson's dynamic factor model has been extended by Kim and Yoo (1995), Chauvet (1998) and Kim and Nelson (1998) to allow for Markov switching dynamics in the common factor which represents the coincident index. In addition to matching very well with the Index of CEI compiled by the Department of Commerce, the Markov switching coincident index is also shown to capture the economic expansions and recessions in the U.S. economy as classified by the National Bureau of Economic Research (NBER). Chauvet and Potter (2000) have developed coincident indicators for the U.S. stock market using the same methodology.

This section is provided to show how the Markov switching coincident index model can be represented as a Markov switching state space model, and estimated using the functions in `S+FinMetrics`.

18.6.1 State Space Representation of Markov Switching Coincident Index Model

Since the levels of most macroeconomic variables are usually found to be non-stationary (for example, see Nelson and Plosser, 1982), it is reasonable to assume that the coincident index representing the state of the economy is also non-stationary. Thus in this example the growth rates of the four macroeconomic variables $\Delta \mathbf{y}_t$ are modeled, and they are assumed to be driven by a common factor ΔC_t interpreted as the change in the coincident index:

$$\Delta \mathbf{y}_t = \boldsymbol{\beta} + \boldsymbol{\lambda}_1 \Delta C_t + \boldsymbol{\lambda}_2 \Delta C_{t-1} + \mathbf{e}_t \tag{18.43}$$

where $\Delta \mathbf{y}_t$, $\boldsymbol{\beta}$, $\boldsymbol{\lambda}_1$, $\boldsymbol{\lambda}_2$ and $\mathbf{e}_t$ are 4×1 vectors with

$$\boldsymbol{\lambda}_1 = \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \lambda_{41} \end{bmatrix}, \ \boldsymbol{\lambda}_2 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \lambda_{42} \end{bmatrix}$$

So the four macroeconomic coincident variables are driven by the common factor ΔC_t and idiosyncratic components $\mathbf{e}_t$. Note that only the current value of ΔC_t affects the first three variables (IP, PI and Sales) in $\Delta \mathbf{y}_t$, while both ΔC_t and ΔC_{t-1} affect the last variable (employees on nonagricultral payroll) because the employment data tend to lag other coincident variables.

The idiosyncratic components are assumed to be independent of each other, and are assumed to follow simple AR(1) models:

$$\mathbf{e}_t = \boldsymbol{\Psi} \mathbf{e}_{t-1} + \boldsymbol{\epsilon}_t, \ \boldsymbol{\epsilon}_t \sim N(0, \boldsymbol{\sigma}^2) \tag{18.44}$$

where $\boldsymbol{\Psi}$ is a diagonal matrix with $(\psi_1, \psi_2, \psi_3, \psi_4)$ on the diagonal, and $\boldsymbol{\sigma}^2$ is a diagonal matrix with $(\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2)$ on the diagonal. The common factor ΔC_t is assumed to follow a Markov switching AR(2) process:

$$\Delta C_t = \delta_{S_t} + \phi_1 \Delta C_{t-1} + \phi_2 \Delta C_{t-2} + u_t, \ u_t \sim N(0, \sigma_C^2) \tag{18.45}$$

where the unobserved discrete state variable S_t follows a two-regime Markov chain, and only the intercept term δ_{S_t} is Markov switching. When the economy is in a recession ($S_t = 1$), the coincident index C_t grows at a slower rate δ_1; and when the economy is in an expansion ($S_t = 2$), the coincident index C_t grows at a faster rate δ_2.

Note that in the above model the intercept term $\boldsymbol{\beta}$ and δ_{S_t} are not separately identified, and the variance term σ_C^2 cannot be separated from the coefficients $\boldsymbol{\lambda}_1$ and $\boldsymbol{\lambda}_2$. To make the model identifiable, the original data $\Delta \mathbf{y}_t$ are standardized to remove its mean and make it scale free so $\boldsymbol{\beta}$ can be set to zero. In addition, the error variance σ_C^2 for ΔC_t can be normalized to one. Using $\boldsymbol{\alpha}_t = (\Delta C_t, \Delta C_{t-1}, \mathbf{e}_t, C_{t-1})$ as the continuous state vector,

the Markov switching coincident index model in (18.43)-(18.45) can now be written in a state space form with the following representation:

$$\boldsymbol{\delta}_t = \begin{bmatrix} \delta_{S_{t+1}} \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \boldsymbol{\Phi}_{S_t} = \begin{bmatrix} \phi_1 & \phi_2 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \psi_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \psi_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \psi_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \psi_4 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ \gamma_1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \gamma_2 & 0 & 0 & 1 & 0 & 0 & 0 \\ \gamma_3 & 0 & 0 & 0 & 1 & 0 & 0 \\ \gamma_{41} & \gamma_{42} & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

and

$$\boldsymbol{\Omega} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_1^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_2^2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_3^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_4^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Note that $C_t = \Delta C_t + C_{t-1}$ is also included as one of the state variables, but does not enter the measurement equation for the observables $\Delta \mathbf{y}_t$. By including C_t as one of the state variables, filtered estimates of C_t can be readily obtained from Kim's filtering algorithm.

By collecting the unknown model parameters in the vector $\boldsymbol{\Theta} = (\delta_1, \delta_2, \phi_1, \phi_2, \psi_1, \psi_2, \psi_3, \psi_4, \lambda_1, \lambda_2, \lambda_3, \lambda_{41}, \lambda_{42}, \sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2, P_{12}, P_{21})$, the following function takes such a vector and returns the state space representation of the model in `S-PLUS`:

```
GetSsfCoinIndex = function(parm) {
  parm = as.vector(parm)
  mDelta = mDelta.other = rep(0, 11)
  mDelta[1] = parm[1]
  mDelta.other[1] = parm[1] + exp(parm[2])
#
  AR.C1 = 2/(1+exp(-parm[3])) - 1
  AR.C2 = 2/(1+exp(-(parm[3]+exp(parm[4])))) - 1
#
  AR.e1 = 2/(1+exp(-parm[5])) - 1
```

```
  AR.e2 = 2/(1+exp(-parm[6])) - 1
  AR.e3 = 2/(1+exp(-parm[7])) - 1
  AR.e4 = 2/(1+exp(-parm[8])) - 1
#
  mPhi = matrix(0, 11, 7)
  mPhi[1,1:2] = c(AR.C1+AR.C2, -AR.C1*AR.C2)
  mPhi[2,1] = 1
  mPhi[3,3] = AR.e1
  mPhi[4,4] = AR.e2
  mPhi[5,5] = AR.e3
  mPhi[6,6] = AR.e4
  mPhi[7,1] = mPhi[7,7] = 1
#
  mPhi[8:10,1] = parm[9:11]
  mPhi[11,1:2] = parm[12:13]
  mPhi[8,3] = mPhi[9,4] = mPhi[10,5] = mPhi[11,6] = 1
#
  mOmega = matrix(0, 11, 11)
  mOmega[1,1] = 1
  mOmega[3,3] = exp(parm[14])
  mOmega[4,4] = exp(parm[15])
  mOmega[5,5] = exp(parm[16])
  mOmega[6,6] = exp(parm[17])
#
  mTrans = matrix(0, 2, 2)
  mTrans[1,2] = 1/(1+exp(-parm[18]))
  mTrans[1,1] = 1-mTrans[1,2]
  mTrans[2,1] = 1/(1+exp(-parm[19]))
  mTrans[2,2] = 1-mTrans[2,1]
#
  mSigma = matrix(0, 8, 7)
  mSigma[1:7, 1:7] = diag(1e+6, 7)
  ans = list(mDelta=mDelta, mDelta.other=mDelta.other,
             mSigma=mSigma, mOmega=mOmega,
             mPhi=mPhi, mTrans=mTrans)
  CheckSsf(ans)
}
```

A few comments on the function `GetSsfCoinIndex` are in order:

1. The second parameter `parm[2]` is actually $\log(\delta_2 - \delta_1)$. By employing this transformation, δ_2 is guaranteed to be greater than δ_1, and thus the first regime can be identified as the recessionary regime and the second as the expansionary regime.

2. Like in the Markov switching AR(2) model for the NASDAQ realized volatility, instead of directly estimating the AR(2) coefficients for ΔC_t, the two real characteristic roots are estimated and the first root is constrained to be the smaller one. By constraining the real characteristic roots to lie within the unit circle, the estimated AR(2) process is guaranteed to be stationary and aperiodic.

3. The AR(1) coefficients for the idiosyncratic components are transformed to guarantee that they lie within $(-1, 1)$, and the corresponding AR processes are stationary.

4. The logarithmic variances $\log \sigma_i^2$ $(i = 1, 2, 3, 4)$ are estimated because they are unbounded.

5. Like in the Markov switching AR(2) model for the NASDAQ realized volatility, the transition probabilities P_{12} and P_{21} are transformed to guarantee that they lie within $(0, 1)$.

6. Finally, diffuse priors on the state vector $\boldsymbol{\alpha}_t$ are employed by setting the top 7×7 block of `mSigma` to a diagonal matrix with large values on the diagonal and zero in the last row.

18.6.2 Approximate MLE of Markov Switching Coincident Index

To fit the above Markov switching model to the four coincident variables, the data are first standardized for model identification and better numerical convergence:

```
> DOC.dat = getReturns(DOC.ts[,1:4], percentage=T)
> DOC.dat@data = t(t(DOC.dat@data) - colMeans(DOC.dat@data))
> DOC.dat@data = t(t(DOC.dat@data) / colStdevs(DOC.dat@data))
```

then the `SsfFitMS` function can be used to fit the model with the following starting values:

```
> DOC.start = c(-1.5, 0.6, 0.3, 0.1, .1, .1, .1, .1, 0.3,
+      0.3, 0.3, 0.3, 0.1, -.5, -.5, -.5, -.5, -1.5, -3)
+ names(DOC.start) = c("mu1", "mu2", "phi1", "phi2", "psi1",
+      "psi2", "psi3", "psi4", "L1", "L2", "L3", "L41",
+      "L42", "s1", "s2", "s3", "s4", "p", "q")
> DOC.fit = SsfFitMS(DOC.start, DOC.dat, GetSsfCoinIndex,
+      l.start=13, trace=T)
> summary(DOC.fit)
Log-likelihood:  -1998.11
432 observations
Parameters:
```

```
       Value Std. Error t value
 mu1 -1.5650    0.30180  -5.187
 mu2  0.6053    0.16900   3.582
phi1 -0.8171    0.20610  -3.965
phi2  0.7124    0.17010   4.187
psi1  0.3711    0.14940   2.484
psi2 -0.6070    0.10590  -5.731
psi3 -0.5169    0.10930  -4.729
psi4 -0.7584    0.18340  -4.135
  L1  0.5059    0.03832  13.200
  L2  0.2977    0.03193   9.322
  L3  0.3480    0.03406  10.220
 L41  0.4443    0.04013  11.070
 L42  0.1966    0.03504   5.610
  s1 -1.1590    0.12180  -9.517
  s2 -0.2758    0.07225  -3.817
  s3 -0.4155    0.07624  -5.449
  s4 -1.3940    0.15220  -9.156
   p -1.9560    0.52340  -3.738
   q -3.7600    0.43460  -8.652

Convergence:  RELATIVE FUNCTION CONVERGENCE
```

Note the optional argument `l.start` to `SsfFitMS` is used to start log-likelihood evaluation from the 13th observation. From the summary output, it can be seen that all the estimated model parameters are significantly different from zero.

To transform the parameters into the original model form, simply call the `GetSsfCoinIndex` function with the estimated parameters:

```
> DOC.ssf = GetSsfCoinIndex(DOC.fit$parameters)
> c(DOC.ssf$mDelta[1], DOC.ssf$mDelta.other[1])
[1] -1.565361  0.266435
> print(DOC.ssf$mPhi, digits=3)
       [,1]  [,2]  [,3]   [,4]   [,5]   [,6] [,7]
 [1,] 0.158 0.211 0.000  0.000  0.000  0.000    0
 [2,] 1.000 0.000 0.000  0.000  0.000  0.000    0
 [3,] 0.000 0.000 0.183  0.000  0.000  0.000    0
 [4,] 0.000 0.000 0.000 -0.295  0.000  0.000    0
 [5,] 0.000 0.000 0.000  0.000 -0.253  0.000    0
 [6,] 0.000 0.000 0.000  0.000  0.000 -0.362    0
 [7,] 1.000 0.000 0.000  0.000  0.000  0.000    1
 [8,] 0.506 0.000 1.000  0.000  0.000  0.000    0
 [9,] 0.298 0.000 0.000  1.000  0.000  0.000    0
[10,] 0.348 0.000 0.000  0.000  1.000  0.000    0
[11,] 0.444 0.197 0.000  0.000  0.000  1.000    0
```

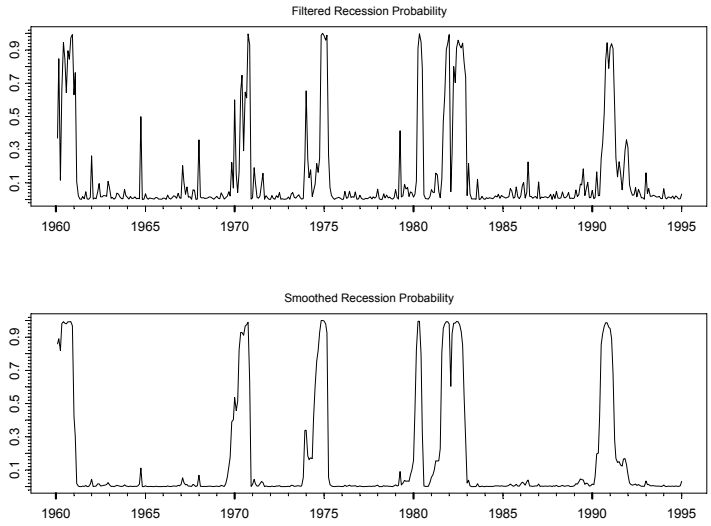

FIGURE 18.12. Filtered and smoothed recession probabilities of Markov switching coincident index.

The growth rate of ΔC_t in a recession is estimated to be -1.57, and the growth rate in an expansion is estimated to be 0.27. Although the growth rates of the four macroeconomic variables are positively correlated with the Markov switching coincident index, only the idiosyncractic component of industrial production has a positive AR(1) coefficient and all other idiosyncratic components have a negative AR(1) coefficient.

To obtain the filtered and smoothed regime probabilities, simply call the `SsfLoglikeMS` function with the estimated state space representation and set the optional argument `save.rgm` to `TRUE`:

```
> DOC.f = SsfLoglikeMS(DOC.dat, DOC.ssf, save.rgm=T,
+ l.start=13)
> DOC.dates = positions(DOC.dat)[-(1:12)]
> filt.p = timeSeries(DOC.f$regimes[,1], pos=DOC.dates)
> smoo.p = timeSeries(DOC.f$regimes[,3], pos=DOC.dates)
> par(mfrow=c(2,1))
> plot(filt.p, reference.grid=F,
+ main="Filtered Recession Probability")
> plot(smoo.p, reference.grid=F,
+ main="Smoothed Recession Probability")
```

and Figure 18.12 shows the filtered and smoothed probabilities for the recession regime.

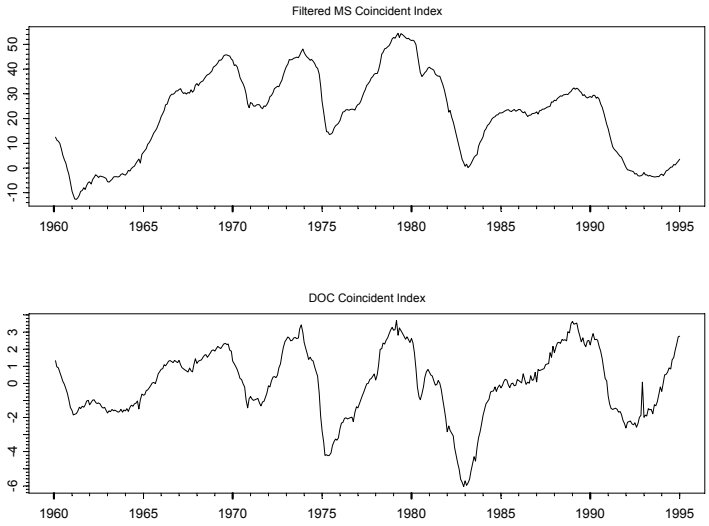

FIGURE 18.13. Filtered Markov switching coincident index and DOC coincident index.

To visualize the estimated Markov switching coincident index, note that the object `DOC.f` also has a `states` component:

```
> names(DOC.f)
[1] "loglike" "err"     "regimes" "states"
```

which contains the filtered estimates of the states $\boldsymbol{\alpha}_{t|t}^{(j)}$ for $j = 1, 2$. Since there are seven state variables in the model, the first seven columns correspond to $\boldsymbol{\alpha}_{t|t}^{(1)}$ and the next seven columns correspond to $\boldsymbol{\alpha}_{t|t}^{(2)}$. The following commands plot the weighted average of filtered estimates of C_t and compare it with the coincident index compiled by the U.S. Department of Commerce:

```
> DOC.index = lm(DOC.ts@data[,5]~I(1:433))$residuals[-(1:13)]
> filt.ci = rowSums(DOC.f$state[,c(7,14)]*DOC.f$regime[,1:2])
> filt.ci = timeSeries(filt.ci, pos=DOC.dates)
> plot(filt.ci, reference.grid=F,
+ main="Filtered MS Coincident Index")
> doc.ci = timeSeries(DOC.index, pos=DOC.dates)
> plot(doc.ci, reference.grid=F,
+ main="DOC Coincident Index")
```

and the plot is shown in Figure 18.13. Note that since the Markov switching coincident index is estimated with demeaned data, a time trend is also

removed from the coincident index `DOC.ts[,5]` compiled by the U.S. Department of Commerce. In general both series share the same pattern, though the Markov switching coincident index seems to be smoother.

18.7 References

[1] Andersen, T., T. Bollerslev, F.X. Diebold and H. Ebens (2001). "The Distribution of Realized Stock Return Volatility", *Journal of Financial Economics*, 61, 43-76.

[2] Aguilar, O. and M. West (2000). "Bayesian Dynamic Factor Models and Portfolio Allocation", *Journal of Business and Economic Statistics*, 18 (3), 338-357.

[3] Barnett, W.A., R.A. Gallant, M.J. Hinich, J.A. Jungeilges, D.T. Kaplan and M.J. Jensen (1997). "A Single-blind Controlled Competition Among Tests for Nonlinearity and Chaos", *Journal of Econometrics*, 82, 157-192.

[4] Brock, W.A., W.D. Dechert and J.A. Scheinkman (1987). "A Test for Independence Based on the Correlation Dimension", mimeo.

[5] Brock, W.A., W.D. Dechert, J.A. Scheinkman and B. LeBaron (1996). "A Test for Independence Based on the Correlation Dimension", *Econometric Reviews*, 15, 197-235.

[6] Brock, W.A., D.A. Hsieh and B. LeBaron (1991). *Nonlinear Dynamics, Chaos, and Instability: Statistical Theory and Economic Evidence*. MIT Press.

[7] Brock, W.A. and S. Potter (1993). "Nonlinear Time Series and Macroeconomics", in G.S. Maddala, C. R. Rao and H. D. Vinod (eds.), *Handbook of Statistics, Vol. II*. North Holland, Amsterdam.

[8] Caporale, G. M., C. Ntantamis, T. Pantelidis and N. Pittis (2004). "The BDS Test As a Test for the Adequacy of a GARCH(1,1) Specification: A Monte Carlo Study", Working Paper, Brunel University.

[9] Chauvet, M. (1998). "An Econometric Characterization of Business Cycle Dynamics with Factor Structure and Regime Switching", *International Economic Review*, 39 (4), 969-996.

[10] Chauvet, M. and S. Potter (2000). "Coincident and Leading Indicators of the Stock Market", *Journal of Empirical Finance*, 7, 87-111.

[11] Davidson, R. and J. G. MacKinnon (1985). "Heteroskedasticity-Robust Tests in Regressions Directions", *Annales de l'INSEE*, 59/60, 183-218.

[12] de Lima, P.J.F. (1996). "Nuisance Parameter Free Properties of Correlation Integral Based Statistics", *Econometric Reviews*, 15, 237-259.

[13] Fernandes, M. and P.-Y. Preumont (2004). "The Finite-Sample Size of the BDS Test for GARCH Standardized Residuals", mimeo.

[14] Franses, P.H. and D. van Dijk (2000). *Non-Linear Time Series Models in Empirical Finance*. Cambridge University Press.

[15] Granger, C.W.J. and T. Teräsvirta (1993). *Modelling Nonlinear Economic Relationships*. Oxford University Press.

[16] Hamilton, J.D. 1989. "A New Approach to the Economic Analysis of Nonstationary Time Series Subject to Changes in Regime", *Econometrica*, 57, 357-384.

[17] Hamilton, J.D. (1994). *Time Series Analysis*. Princeton University Press.

[18] Hansen, B.E. (1996). "Inference When a Nuisance Parameter is Not Identified Under the Null Hypothesis", *Econometrica*, 64, 413-430.

[19] Hansen, B.E. (1997). "Inference in TAR Models", *Studies in Nonlinear Dynamics and Econometrics*, 2, 1-14.

[20] Hansen, B.E. (1999). "Testing for Linearity", *Journal of Economic Surveys*, 13 (5), 551-576.

[21] Kim, C.J. (1994). "Dynamic Linear Models with Markov-Switching", *Journal of Econometrics*, 60, 1-22.

[22] Kim, C.-J. and C.R. Nelson (1998). "Business Cycle Turning Points, a New Coincident Index, and Tests of Duration Dependence Based on a Dynamic Factor Model with Regime-Switching", *Review of Economics and Statistics*, 80, 188-201.

[23] Kim, C.-J. and C.R. Nelson (1999). *State-Space Models with Regime-Switching: Classical and Gibbs-Sampling Approaches with Applications*. MIT Press.

[24] Kim, M.-J. and J.-S. Yoo (1995). "New Index of Coincident Indicators: A Multivariate Markov Switching Factor Model Approach", *Journal of Monetary Economics*, 36, 607-630.

[25] Kim, S., N. Shephard and S. Chib (1998). "Stochastic Volatility: Likelihood Inference and Comparison with ARCH Models", *Review of Economic Studies*, 65, 361-393.

[26] LeBaron, B. (1997). "A Fast Algorithm for the BDS Statistic", *Studies in Nonlinear Dynamics and Econometrics*, 2, 53-59.

[27] Lundbergh, S. and T. Teräsvirta (1998). "Modelling Economic High-Frequency Time Series with STAR-GARCH Models", Working Paper Series in Economics and Finance No. 291, Stockholm School of Economics.

[28] Lundbergh, S. and T. Teräsvirta (2002). "Forecasting with Smooth Transition Autoregressive Models", in M. P. Clements and D. F. Hendry (eds.), *A Companion to Economic Forecasting*. Blackwell Publishers.

[29] Luukkonen, R, P. Saikkonen and T. Teräsvirta (1988). "Testing Linearity Against Smooth Transition Autoregressive Models", *Biometrika* 75, 491–499.

[30] Nelson, C.R. and C.I. Plosser (1982). "Trends and Random Walks in Macroeconomic Time Series: Some Evidence and Implications", *Journal of Monetary Economics*, 10, 139-162.

[31] Teräsvirta, T. (1994). "Specification, Estimation, and Evaluation of Smooth Transition Autoregressive Models", *Journal of the American Statistical Association*, 89, 208–218.

[32] Stock, J.H. and M.W. Watson (1991). "A Probability Model of the Coincident Economic Indicators", in K. Lahiri and G.H. Moore (eds.), *Leading Economic Indicators: New Approaches and Forecasting Records*. Cambridge University Press.

[33] Tong, H. (1978). "On a Threshold Model", in C.H. Chen (ed.), *Pattern Recognition and Signal Processing*. Amsterdam: Sijhoff & Noordhoff.

[34] Tong, H. (1990). *Non-Linear Time Series: A Dynamical System Approach*. Oxford University Press.

[35] Tsay, R.S. (1989). "Testing and Modeling Threshold Autoregressive Processes", *Journal of the American Statistical Association*, 84 (405), 231-240.

[36] van Dijk, D., T. Teräsvirta and P.H. Franses (2002). "Smooth Transition Autoregressive Models – A Survey of Recent Developments", *Econometric Reviews*, 21 (1), 1-47.