Chapter 5

Threshold VAR/Cointegration

The extension of threshold models to multivariate settings is largely a matter of replacing univariate likelihoods with multivariate ones. Threshold error correction models are probably the most useful of the three basic techniques examined here, since the equilibrium condition is an obvious candidate for a threshold variable.

5.1 Threshold Error Correction

The first paper to address the question of *threshold cointegration* is Balke and Fomby (1997), though it's probably more accurate to describe this as threshold error correction. Their paper[1] analyzed the spread between the Federal Funds rate and the discount rate; thus, the cointegrating vector is considered to be known in advance as $(1, -1)$. The case where the cointegrating vector isn't known and must be estimated is much more complicated, and will be addressed in Section 5.3.

The point raised by Balke and Fomby is that the Fed would be unlikely to allow arbitrary spreads between those two rates. The Fed controls the discount rate, but exercises only indirect control over the funds rate. They would likely make a policy reaction to reduce the gap if the spread got either too large or too small, but would be less likely to intervene in a band closer to a normal spread. The expected behavior would be that the error correction term would be small (effectively zero) in the center band, but non-zero in the high and low bands.

Most of the analysis is a TAR model on the the spread itself. The authors do a Tsay threshold test (Section 4.2.1) with several different delays, and in both the direct and reversed direction. Because the **@TSAYTEST** procedure allows arbitrary threshold series (not just lags of the dependent variable), the reversed test is easily done with:

```
set thresh = -spread{d}
@tsaytest(thresh=thresh) spread
# constant spread{1 2}
```

All the test statistics are significant, but the two with $d = 1$ are by far the strongest, so the remaining analysis uses the lagged spread as the threshold variable.

[1] The empirical application is only in the working paper version, not the journal article.

Under the presumed behavior, the spread should show stationary behavior in the two tails and unit root behavior in the middle portion. To analyze this, the authors do an arranged Dickey-Fuller test—a standard D-F regression, but with the data points added in threshold order. Computing this requires the use of the **RLS** instruction, saving both the history of the coefficients and of the standard errors of coefficients to compute the sequence of D-F t-tests:

```
set dspread = spread-spread{1}
set thresh  = spread{1}
rls(order=thresh,cohistory=coh,sehistory=seh) dspread
# spread{1} constant dspread{1}
*
set tstats = coh(1)(t)/seh(1)(t)
```

This is done with the recursions in each direction. The clearer of the two is with the threshold in increasing order (Figure 5.1). Once you have more than a handful of data points in the left tail, the D-F tests very clearly reject unit root behavior, a conclusion which reverses rather sharply beginning around a threshold of 1, though where exactly this starts to turn isn't easy to read off the graph since the data in use at that point are overwhelmingly in the left tail—the observed values of the spread are mainly in the range of -.5 to .5.

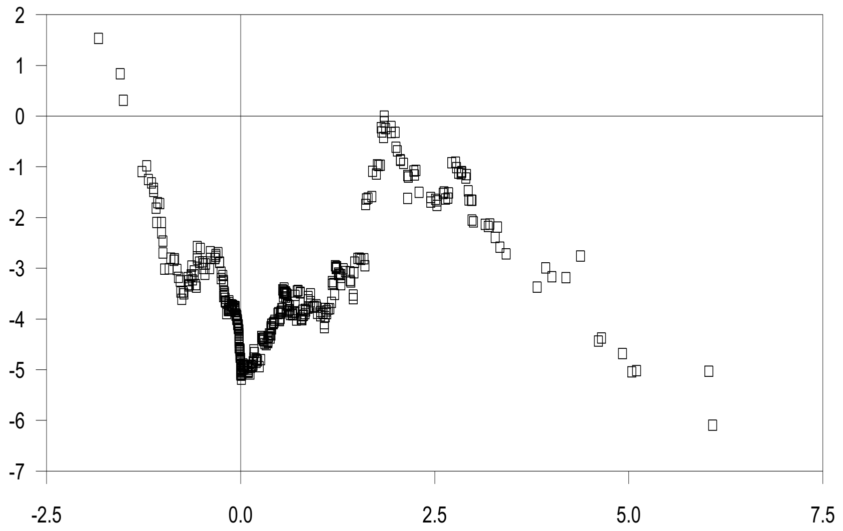

Figure 5.1: Recursive D-F T-Statistics

Based upon this graph, the authors did a (very) coarse grid search for the two threshold values in a two-lag TAR on the spread using all combinations of a lower threshold in $\{-.2, -.1, 0, .1, .2\}$ and an upper threshold in $\{1.6, 1.7, 1.8, 1.9, 2.0\}$, getting the minimum sum of squares at -.2 and 1.6. A more complete grid search can be done fairly easily and quickly at modern computer speeds. As is typical of (hard) threshold models, the sum of squares function is flat for thresholds between observed values, so the search needs only to look at those observed values as potential break points. A more comprehensive search finds the optimal breaks as -.45 and 1.45. If we look at the cross section of the log likelihood surface with the upper threshold fixed at the optimizing 1.45 (figure 5.2), you can see that the function is not just discontinuous, but more generally quite ill-behaved. This is fairly typical since each new threshold value generally shifts only one data point between two of the partitions—if that data point happens to be very influential in one or both of the subsamples, it could cause a temporary blip up or down to the overall sum of squares.

One thing to note is that the overall range is actually quite small—the top to bottom range is only about 5 and almost 1/3 of the values (across a broad range) have log likelihoods high enough that they would be acceptable at the .05 level in a likelihood ratio test for a specific value for the left threshold given the right threshold.[2] Thus, while the data strongly support a two threshold model, they don't so strongly identify where they are. The analysis using the techniques from the paper are shown in Example **5.1**.

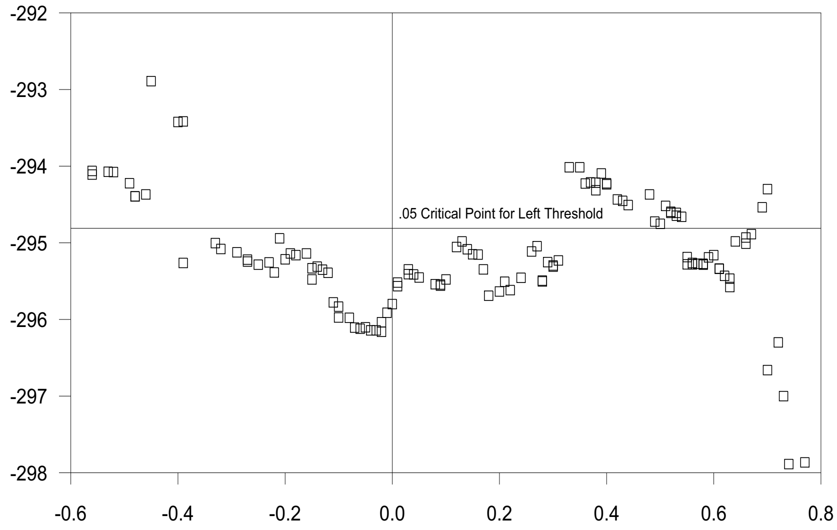

Figure 5.2: Log Likelihood as function of left threshold given right threshold

An alternative to doing the TAR model on the spread is to choose breaks using the multivariate likelihood for the actual VECM model. This is also done easily using the instruction **SWEEP**. The following does the base multivariate regression of the two differenced variables on 1, one lag of the differenced variables and the lagged spread:

```
sweep
# dff ddr
# constant dff{1} ddr{1} spread{1}
compute loglbest=%logl
```

[2] And since the right threshold of 1.45 isn't even necessarily the constrained maximizer for any given value of the left threshold, even more left threshold values are likely to be acceptable if we did the required number crunching to do a formal likelihood ratio test for each.

%LOGL is the log likelihood of the systems regression. With groupings, it does separate systems estimates, aggregating the outer product of residuals to get an overall likelihood:

```
sweep(group=(thresh>=tvalues(lindex))+(thresh>tvalues(uindex)))
# dff ddr
# constant dff{1} ddr{1} spread{1}
```

The estimation using this gives the (much) tighter range of .63 to 1.22 as the center subsample.

In order to do any type of forecasting calculation with this type of model, we need to define identities to connect the five series: the two rates, the spread and the changes. And we need to define formulas for the two rates of change which switch with the simulated values of the threshold variable.

With the lower and upper threshold values in the variables of the same name, the following formula will evaluate to 1, 2 or 3 depending upon where the value of spread{1} falls:

```
dec frml[int] switch
frml switch = 1+fix((spread{1}>=lower)+(spread{1}>upper))
```

Note that you must use spread{1} in this, not a fixed series generated from spread that we can use during estimation—when we do simulations spread is no longer just data.

The most flexible way to handle the switching functions for the changes is to define a RECT[FRML] with rows for equations and columns for partitions. Using separate formulas for each case (rather than using the same equation changing the coefficients) makes it easier to use different forms. In this case, we *are* using the same functional form for all three branches, but that might not be the best strategy.

```
system(model=basevecm)
variables dff ddr
lags 1
det constant spread{1}
end(system)
*
dec rect[frml] tvecfrml(2,3)
do i=1,3
   estimate(smpl=(switch(t)==i))
   frml(equation=%modeleqn(basevecm,1)) tvecfrml(1,i)
   frml(equation=%modeleqn(basevecm,2)) tvecfrml(2,i)
end do i
```

The identities are straightforward:

```
frml(identity) ffid fedfunds = fedfunds{1}+dff
frml(identity) drid mdiscrt  = mdiscrt{1}+ddr
frml(identity) spid spread   = fedfunds-mdiscrt
```

With the work already done above, the switching formulas are now quite simple:

```
frml dffeq dff = tvecfrml(1,switch(t))
frml ddreq ddr = tvecfrml(2,switch(t))
```

The working model is then constructed with:

```
group tvecm dffeq ddreq ffid drid spid
```

An eventual forecast function can be generated using the **FORECAST** instruction. As before, this is intended to show one possible path of the system, not to give a minimum mean-square error forecast.

```
forecast(model=tvecm,from=1981:1,steps=40,results=eff)
graph(footer=$
   "Eventual Forecast Function for SPREAD, starting at 1981:1")
# eff(5)
graph(footer=$
   "Eventual Forecast Function for Change in DR, starting at 1981:1")
# eff(2)
```

The full code for the joint analysis of the model is in example **5.2**.

5.2 Threshold VAR

This is handled in a similar fashion to the threshold error correction model and is simpler. An example is in the replication file for Tsay (1998), which is also given here as Example **5.3**. Unfortunately, his U.S. interest rates example seems quite poorly done. To estimate the one break model, he uses a grid over values of the threshold running from -.30 to .05.[3] In this case, using a grid based upon empirical values would have required almost exactly the same amount of calculation. Given the values that the average spread series takes, .05 is far too large to be considered—it's larger than the 95%-ile of the series, and with a seven lag VAR, there are almost no degrees of freedom in an estimate partitioned there. A more reasonable upper limit would have been -.05, but even with that, the best break is at -.05.[4] The following does the one break model as described in the article:

[3] His threshold series is a three-month moving average of the spread between the two interest rates.

[4] We discovered (accidentally) that the results in the paper can be reproduced almost exactly by using an incorrect calculation of the likelihood. The error adds a term to the likelihood which is maximized when the number of data points is roughly the same in each partition.

```
@gridseries(from=-.30,to=.05,n=300,pts=ngrid) rgrid
set aic 1 ngrid = 0.0
*
do i=1,ngrid
   compute rtest=rgrid(i)
   sweep(group=thresh<rtest,var=hetero)
   # g3year g3month
   # constant g3year{1 to p} g3month{1 to p}
   compute aic(i)=-2.0*%logl+2.0*%nregsystem
end do i
scatter(footer=$
  "AIC Values for Interest Rates Regression vs Break Point")
# rgrid aic
```

Note that this uses the VAR=HETERO option on **SWEEP**. That allows the covariance matrix to be different between partitions and computes the likelihood function on that basis. This can also be done with **SWEEP** with just a single target variable.

The double break model can now (with faster computers) be done in a reasonable amount of time using the empirical grid. As in the paper, this uses a somewhat coarser grid, though it's a slightly different one than is described there.

```
@gridseries(from=-.30,to=.05,n=80,pts=ngrid) rgrid
*
compute bestaic=%na
do i=1,ngrid-19
   do j=i+20,ngrid
      sweep(group=(thresh<rgrid(i))+(thresh<rgrid(j)),var=homo)
      # g3year g3month
      # constant g3year{1 to p} g3month{1 to p}
      compute thisaic=-2.0*%logl+2.0*%nregsystem
      if .not.%valid(bestaic).or.thisaic<bestaic
         compute bestaic=thisaic,bestlower=rgrid(i),bestupper=rgrid(j)
   end do j
end do i
*
disp "Best double break is at" bestlower "and" $
    bestupper "with AIC" bestaic
```

5.3 Threshold Cointegration

The first paper to tackle the question of threshold cointegration with an unknown cointegrating vector was Hansen and Seo (2002). The empirical example in this unfortunately suffered from a number of major errors, not the least of which was an incorrect data set.[5]

This is much more complicated than the case of the known cointegration vector—whether it's even worth the trouble is unclear since it's unlikely that it will be possible to do much in the way of interesting inference on the cointegrating vector. With a known cointegrating vector, we can compute it as a series, graph it, analyze it. We know its values, so we can do the calculations for break-points across a fixed set of points. If it's unknown, then different values of β give rise to different series with different behavior. We've already seen that the log likelihood with a *known* cointegrating vector produces a highly variable function. This is compounded quite a bit when you search both across β and the threshold. Given β, the possible thresholds can be computed as before, so an exhaustive search is possible. That isn't possible for β itself, which must be analyzed on a grid. With a grid which is too coarse, it's very easy to miss the global maximum, and no matter how fine the grid, you can never really be sure that you've even found the global maximum since the likelihood function is so ill-behaved.

The procedures for implementing the Hansen-Seo techniques are all in the replication program. The estimation itself is done with the procedure **@HansenSeo**. This uses the first lag of the cointegrating relation as the threshold variable.

```
@HansenSeo( options )  start  end   y1  y2
```

Its options are:

- LAGS=# *of lags in the textscvar*
- BETA =*input value for cointegrating coefficient* (y1-beta*y2 stationary) [not used]
- GAMMA=*threshold value for partioning sample* [not used]
- BSIZE=*size of beta grid search* [300]
- GSIZE=*size of gamma grid search* [300]
- PI=*minimum fraction sample in a partition* [.05]

The two key parameters are β, the coefficient in the cointegrating relation, and γ, the breakpoint on the threshold variable. You can input either or search for either.

The procedure **@HSLMTest** tests for threshold cointegration *given an input β*. This does fixed regressor bootstrapping for computing an approximate p-value.

[5] The data from roughly the first 50 observations were accidentally read twice and appended to the proper data set.

Example 5.1 Threshold Error Correction Model

This largely reproduces the results in Balke and Fomby (1997). The data file is a reconstruction, so the results differ slightly.

```
cal(m) 1955:1
open data irates.xls
data(format=xls,org=columns) 1955:01 1990:12 fedfunds mdiscrt
*
set spread = fedfunds-mdiscrt
@dfunit(lags=12) fedfunds
@dfunit(lags=12) mdiscrt
@dfunit(lags=12) spread
*
* Pick lags
*
@arautolags(crit=bic) spread
*
* Tsay threshold tests with direct ordering
*
do d=1,4
   set thresh = spread{d}
   @tsaytest(thresh=thresh,$
      title="Tsay Test-Direct Order, delay="+d) spread
   # constant spread{1 2}
end do d
*
* And with reversed ordering
*
do d=1,4
   set thresh = -spread{d}
   @tsaytest(thresh=thresh,$
     title="Tsay Test-Reverse Order, delay="+d) spread
   # constant spread{1 2}
end do d
*
* Arranged D-F t-statistics
*
set dspread = spread-spread{1}
set thresh  = spread{1}
rls(order=thresh,cohistory=coh,sehistory=seh) dspread
# spread{1} constant dspread{1}
*
set tstats = coh(1)(t)/seh(1)(t)
scatter(footer="Figure 1. Recursive D-F T-Statistics\\"+$
  "Arranged Autoregression from Low to High")
# thresh tstats
*
* Same thing in reversed order
*
rls(order=-thresh,cohistory=coh,sehistory=seh) dspread
# spread{1} constant dspread{1}
*
```

```
set tstats = coh(1)(t)/seh(1)(t)
scatter(footer="Figure 2. Recursive D-F T-Statistics\\"+$
   "Arranged Autoregression from High to Low")
# thresh tstats
*
* The authors do a very coarse grid. This does a full grid over the
* possible threshold values requiring at least 15% of observed values to
* value in each group. (This isn't necessarily 15% of the data points
* because of possible duplication).
*
set thresh = spread{1}
linreg(noprint) spread
# constant spread{1 2}
compute loglbest=%logl
@UniqueValues(values=tvalues) thresh %regstart() %regend()
*
compute n=%rows(tvalues)
compute pi=.15
*
* These are for doing a graph later, and aren't necessary for the
* analysis.
*
compute x=tvalues
compute y=tvalues
compute f=%fill(n,n,%na)
*
compute spacing=fix(pi*n)
*
* These are the bottom and top of the permitted index values for the
* lower index, and the top of the permitted values for the upper index.
*
compute lstart=spacing,lend=n+1-2*spacing
compute uend  =n+1-spacing
do lindex=lstart,lend
   do uindex=lindex+spacing,uend
      sweep(group=(thresh>=tvalues(lindex))+(thresh>tvalues(uindex)))
      # spread
      # constant spread{1 2}
      if %logl>loglbest
         compute lindexbest=lindex,uindexbest=uindex,loglbest=%logl
      compute f(lindex,uindex)=%logl
   end do uindex
end do lindex
*
disp "Best Break Values" tvalues(lindexbest) "and" tvalues(uindexbest)
*
* This graphs the log likelihood function across values of the left
* threshold, given the maximizing value for the right threshold. It also
* includes a line at the .05 critical value for a likelihood ratio test
* for a specific value of the left threshold.
*
set testf 1 n = f(t,uindexbest)
set testx 1 n = tvalues(t)
compute yvalue=loglbest-.5*%invchisqr(.05,1)
```

```
spgraph
scatter(vgrid=yvalue,footer=$
   "Log likelihood as function of left threshold given right threshold")
# testx testf
scatter(vgrid=yvalue)
# testx testf
grtext(y=yvalue,x=0.0,direction=45) ".05 Critical Point for Left Threshold"
spgraph(done)
*
* This will be 0, 1 or 2, depending upon the value of thresh
*
set group = (thresh>=tvalues(lindexbest))+(thresh>tvalues(uindexbest))
*
dofor i = 0 1 2
   disp "**** Group " i "****"
   linreg(smpl=(group==i)) spread
   # constant spread{1 2}
   summarize(noprint) %beta(2)+%beta(3)-1.0
   disp "DF T-Stat" %cdstat
end do i
*
* Threshold error correction models
*
set dff = fedfunds-fedfunds{1}
set ddr = mdiscrt -mdiscrt{1}
dofor i = 0 1 2
   disp "**** Group " i "****"
   linreg(smpl=(group==i)) dff
   # constant dff{1} ddr{1} spread{1}
   linreg(smpl=(group==i)) ddr
   # constant dff{1} ddr{1} spread{1}
end do i
```

Example 5.2 Threshold Error Correction Model: Forecasting

This is based upon Balke and Fomby (1997). However, this estimates the threshold using the bivariate likelihood, and computes eventual forecast functions and GIRFs.

```
cal(m) 1955:1
open data irates.xls
data(format=xls,org=columns) 1955:01 1990:12 fedfunds mdiscrt
*
set spread = fedfunds-mdiscrt
set thresh = spread{1}
linreg spread
# constant spread{1 2}
@UniqueValues(values=tvalues) thresh %regstart() %regend()
*
compute n=%rows(tvalues)
compute pi=.15
*
```

```
compute spacing=fix(pi*n)
*
* These are the bottom and top of the permitted index values for the
* lower index, and the top of the permitted values for the upper index.
*
compute lstart=spacing,lend=n+1-2*spacing
compute uend  =n+1-spacing
*
set dff = fedfunds-fedfunds{1}
set ddr = mdiscrt -mdiscrt{1}
*
sweep
# dff ddr
# constant dff{1} ddr{1} spread{1}
compute loglbest=%logl
*
do lindex=lstart,lend
   do uindex=lindex+spacing,uend
      sweep(group=(thresh>=tvalues(lindex))+(thresh>tvalues(uindex)))
      # dff ddr
      # constant dff{1} ddr{1} spread{1}
      if %logl>loglbest
         compute lindexbest=lindex,uindexbest=uindex,loglbest=%logl
   end do uindex
end do lindex
disp "Best Break Values" tvalues(lindexbest) "and" tvalues(uindexbest)
*
compute lower=tvalues(lindexbest),upper=tvalues(uindexbest)
dec frml[int] switch
frml switch = 1+fix((spread{1}>=lower)+(spread{1}>upper))
*
* Estimate the model at the best breaks to get the covariance matrix.
*
sweep(group=switch(t))
# dff ddr
# constant dff{1} ddr{1} spread{1}
compute tvecmsigma=%sigma
*
set dff = fedfunds-fedfunds{1}
set ddr = mdiscrt -mdiscrt{1}
*
system(model=basevecm)
variables dff ddr
lags 1
det constant spread{1}
end(system)
*
dec rect[frml] tvecfrml(2,3)
do i=1,3
   estimate(smpl=(switch(t)==i))
   frml(equation=%modeleqn(basevecm,1)) tvecfrml(1,i)
   frml(equation=%modeleqn(basevecm,2)) tvecfrml(2,i)
end do i
*
```

```
frml(identity) ffid fedfunds = fedfunds{1}+dff
frml(identity) drid mdiscrt  = mdiscrt{1}+ddr
frml(identity) spid spread   = fedfunds-mdiscrt
*
frml dffeq dff = tvecfrml(1,switch(t))
frml ddreq ddr = tvecfrml(2,switch(t))
*
group tvecm dffeq ddreq ffid drid spid
*
* Eventual forecast function, starting with 1981:1 data (largest value
* of spread).
*
forecast(model=tvecm,from=1981:1,steps=40,results=eff)
graph(footer=$
   "Eventual Forecast Function for SPREAD, starting at 1981:1")
# eff(5)
graph(footer=$
   "Eventual Forecast Function for Change in DR, starting at 1981:1")
# eff(2)
*
* GIRF starting in 1969:3 for a one s.d. shock to DR correlated with FF
* using the estimated covariance matrix. (1969:3 has values for both
* rates which are close to the average for the full period).
*
compute ndraws=5000
compute baseentry=1969:3
compute nsteps   =40
*
dec vect[series] fshocks(2) girf(5)
dec series[vect] bishocks
dec vect ishocks
*
smpl baseentry baseentry+(nsteps-1)
do i=1,5
   set girf(i) = 0.0
end do i
*
compute fsigma=%psdfactor(tvecmsigma,||2,1||)
*
do draw=1,ndraws
   gset bishocks = %ranmvnormal(fsigma)
   set fshocks(1) = bishocks(t)(1)
   set fshocks(2) = bishocks(t)(2)
   forecast(paths,model=tvecm,results=basesims)
   # fshocks
   compute ishock=fsigma(2,2)
   compute ishocks=inv(fsigma)*bishocks(baseentry)
   compute ishocks(2)=ishock/fsigma(2,2)
   compute bishocks(baseentry)=fsigma*ishocks
   compute fshocks(1)(baseentry)=bishocks(baseentry)(1)
   compute fshocks(2)(baseentry)=bishocks(baseentry)(2)
   forecast(paths,model=tvecm,results=sims)
   # fshocks
   do i=1,5
```

```
      set girf(i) = girf(i)+(sims(i)-basesims(i))
   end do i
end do draw
*
do i=1,5
   set girf(i) = girf(i)/ndraws
end do i
*
graph(footer=$
  "GIRF for Discount Rate to One S.D. Shock in Discount Rate")
# girf(4)
graph(footer=$
  "GIRF for FedFunds Rate to One S.D. Shock in Discount Rate")
# girf(3)
graph(footer=$
  "GIRF for Spread to One S.D. Shock in Discount Rate")
# girf(5)
*
smpl
```

Example 5.3 Threshold VAR

This is based upon Tsay (1998). Data are similar, but not identical to those used in the paper.

```
open data usrates.xls
calendar(m) 1959
data(format=xls,org=columns) 1959:1 1993:2 fcm3 ftb3
set g3year  = log(fcm3/fcm3{1})
set g3month = log(ftb3/ftb3{1})
set spread  = log(ftb3)-log(fcm3)
set sspread = (spread+spread{1}+spread{2})/3
compute sspread(1959:1)=spread(1959:1)
compute sspread(1959:2)=(spread(1959:1)+spread(1959:2))/2
*
spgraph(vfields=3,$
  footer="Figure 3. Time Plots of Growth Series of U.S. Monthly Interest Rates")
graph(vlabel="3-month")
# g3month
graph(vlabel="3-year")
# g3year
graph(vlabel="Spread")
# sspread
spgraph(done)
*
@VARLagSelect(lags=12,crit=aic)
# g3year g3month
*
compute p =7
compute k =2
*
do d=1,7
```

```
   dofor m0 = 50 100
      set thresh = sspread{d}
      *
      rls(noprint,order=thresh,condition=m0) g3year / rr3year
      # constant g3year{1 to p} g3month{1 to p}
      rls(noprint,order=thresh,condition=m0) g3month / rr3month
      # constant g3year{1 to p} g3month{1 to p}
      *
      * We need to exclude the conditioning observations, so we generate
      * the series of ranks of the threshold variable over the
      * estimation range.
      *
      order(ranks=rr) thresh %regstart() %regend()
      *
      linreg(noprint,smpl=rr>m0) rr3year / wr3year
      # constant g3year{1 to p} g3month{1 to p}
      linreg(noprint,smpl=rr>m0) rr3month / wr3month
      # constant g3year{1 to p} g3month{1 to p}
      *
      ratio(mcorr=%nreg,degrees=k*%nreg,noprint)
      # rr3year rr3month
      # wr3year wr3month
      disp "D=" d "m0=" m0 @16 "C(d)=" *.## %cdstat @28 "P-value" #.##### %signif
   end dofor m0
end do d
*
* Evaluate the AIC across a grid of threshold settings
*
set thresh = sspread{1}
@gridseries(from=-.30,to=.05,n=300,pts=ngrid) rgrid
set aic 1 ngrid = 0.0
*
compute bestaic=%na
*
do i=1,ngrid
   compute rtest=rgrid(i)
   sweep(group=thresh<rtest,var=homo)
   # g3year g3month
   # constant g3year{1 to p} g3month{1 to p}
   compute aic(i)=-2.0*%logl+2.0*%nregsystem
   if i==1.or.aic(i)<bestaic
      compute bestaic=aic(i),bestbreak=rtest
end do i
*
scatter(footer=$
  "AIC Values for Interest Rates Regression vs Break Point")
# rgrid aic
disp "Best Break is" bestbreak "with AIC" bestaic
*
set thresh = sspread{1}
*
* This is a slightly different set of grids than used in the paper.
*
@gridseries(from=-.30,to=.05,n=80,pts=ngrid) rgrid
```

```
*
compute bestaic=%na
do i=1,ngrid-19
   do j=i+20,ngrid
      sweep(group=(thresh<rgrid(i))+(thresh<rgrid(j)),var=homo)
      # g3year g3month
      # constant g3year{1 to p} g3month{1 to p}
      compute thisaic=-2.0*%logl+2.0*%nregsystem
      if .not.%valid(bestaic).or.thisaic<bestaic
         compute bestaic=thisaic,bestlower=rgrid(i),bestupper=rgrid(j)
   end do j
end do i
*
disp "Best double break is at" bestlower "and" $
    bestupper "with AIC" bestaic
```