DYNAMIC DISTRIBUTION ANALYSIS

User's Guide

by Luciano Gutierrez
Dept. of Agricultural Economics
University of Sassari
Italy

Correspondence :
Luciano Gutierrez
Department of Agricultural Economics
University of Sassari
Via E. De Nicola 1, Sassari 07100
Italy

Tel.: +39.079.229.256
Fax: +39.079.229.356
e-mail: lgutierr@uniss.it

Install

I assume that GAUSS for DOS 3.2 (or higher) for Windows 95, 98, 2000, ME platforms has been installed in the GAUSS directory.

1. Unzip the file panel.zip.
2. Copy "DISTRIB.PRG" in the c:\GAUSS\SRC directory.
3. Copy "DISTRIB.LCG" file in the c:\GAUSS\LIB directory

4.1 At the GAUSS prompt, before using the DISTRIB routines, run the command
>library distrib;

4.2 or include the command library distrib; at the start of the batch file that uses the routines

4.3 or insert the command library distrib; in the GAUSS Startup file usually contained in the GAUS directory. (This may be the best choice. Any time you run GAUSS the panel routines will be on line).

```
/*************************************************************************
*************************************************************************
*************************************************************************
** DYNAMIC DISTRIBUTION ANALYSIS
** written by Luciano Gutierrez, last version Dic. 2003
**
** This  code has  been developed  specifically for use in dynamic distribution
** analysis as part of my research projects.
** While I am fairly confident that it is working well, if people find
** errors/inefficiencies or  can see improvements in the algorithm used,
** I would appreciate hearing from them.
** This  code is  written  and submitted for  public use. There are no
** performance  guarantees.  Please  acknowledge this code and author
** if you find it useful in your own work.
**
** Author  : Luciano Gutierrez
**           Department of Agricultural Economics
**           University of Sassari
**           Via De Nicola 1
**           07100 Sassari, Italy
**           tel. (39)(79) 229.256
**           fax. (39)(79) 229.356
**           e-mail: luciano.gutierrez@tin.it
**
******* Discrete Transition probability matrix:
**
** Proc : Trans    : compute discrete M transition matrix as in Quah (1993)
**
******* Univariate distribution estimation
**
** Proc : uni_ker   : compute univariate density from Gaussian or Epanechnikov's
**                    kernels
**
******* Bivariate distribution estimation
**
** Proc : biv_ker   : compute bivariate density from Gaussian or Epanechnikov's
**                    kernels
**
** Proc : stoch_ker : compute stochastic kernel from Gaussian or Epanechnikov's
**                    kernels
**
** Proc : condit    : conditioning regression
**
** Proc : granger_test : compute Granger's bivariate VAR non causality tests
**
** Proc : med_ker   : compute medians from stochastic kernel density
**
**
******** Kernels
**
** Univariate kernel: &ker_gauss   Gaussian kernel
**                    &ker_epan    Epanechnikov kernel
**
** Bivariate kernel : &ker2_gauss  Gaussian kernel
**                    &ker2_epan1  Epanechnikov kernel
**                    &ker2_epan2  Epanechnikov kernel (squared)
**                    &ker2_epan3  Epanechnikov kernel (cubic)
**
*************************************************************************
*************************************************************************
*************************************************************************
*/
```

Routines

```
/*
** Procedure Trans
**
** Compute transition matrix M as in Quah (1993)
**
**  {M_mat,F_mat,Marg,F_con,P} = trans(x,grid);
**
**  Input:      x   = matrix (N,T) N=cross-sections, T=observations
**            grid1 = grid  values
**             iter = number of iteration to compute long-run transition
**                    matrix
**
** Output:    M_mat = matrix of obs (stat_j,state_k)
**            F_mat = Joint probability discrete density
**            Marg  = Marginal probability discrete density
**            F_con = Conditional probability discrete density
**            P     = ergodic density
**
**
** Ex. x(101,2) = Income of country i relative to average income.
**                i=1,...,N where N=101 countries
**                t=1,...,T where T=2 ==> x_1 first column average 1960-1980
**                                        x_2 second column average 1981-1998
**
**      grid1 = {0, 0.25, 0.5, 1, 2, inf} average income intervals, Quah(1993)
**
** output:
**
**      M_mat  = number of obs. in each cell     : Rows, x_2; Cols : x_1
**      F_mat  = Joint probability density       : p(x_2,x_1)
**      Marg   = Marginal probability density    : p(x_1)
**      F_con  = Conditional probability density : p(x_2|x_1)=p(x_2,x_1)/p(x_1)
**      F_erg  = ergodic distribution
**      years  = half_life (years) mobility index (if 9999999 second max
**                 eigenvalue = 1;
*/
```

```
/*
** Procedure Uni_ker
**
** Compute univariate density estimation
**
**  {f,h} = uni_ker(x,min,max,h,&ker_func);
**
**  Input:  x          = matrix (N,T) N=cross-sections, T=observations
**          min        = left  initial grid value if 0 min=minc(x)
**          max        = right final grid value if 0 max=maxc(x);
**          h          = window with if 0 computed following Silverman (1986)
**          &ker_func  = kernel method &ker_gauss Gaussian kernel
**                                    &ker_epan Epanechnikov kernel
**
** Output:    F = estimated density
**            h = window width
**
*/
```

```
/*
** Procedure Stoch_ker
**
** Compute stochastic kernel density estimation
**
** {f_cond,f_joint,f_erg}=stoch_ker(x1,x2,min,max,M,auto_width,iter,&ker_func);
**
**  Input:  x1         = matrix (N,T) N=cross-sections, T=observations
**          x2         = matrix (N,T) N=cross-sections, T0observations
**          min        = vector (2,1) left initial grid values for x1 and x2.
**                       if (-1|-1) min = {minc(x1) | minc(x2)};
**          max        = vector (2,1) right final grid values for x1 and x2;
**                       if (-1|-1) max = {maxc(x1) | maxc(x2)};
**          M          = number of intervals in the grid - 1.
**          auto_width = multiply windows with automatically choosen by the
**                       constant autowidth.
**          iter       = number of iterations when computing ergodic density
**          &ker_func  = kernel method &ker2_gauss Gaussian kernel
**                                     &ker2_epan1         Epanechnikov kernel
**                                     &ker2_epan2 squared Epanechnikov kernel
**                                     &ker2_epan3 cubic   Epanechnikov kernel
**
**
** Output: f_cond  = stochastic kernel
**         f_joint = joint density
**         f_erg   = ergodic f_y density
**
*/
```

```
/*
** Procedure Biv_ker
**
** Compute bivariate density estimation
**
**  {f_joint,step1,step2} = biv_ker(x1,x2,min,max,M,auto_width,&ker_func);
**
**  Input:  x1         = matrix (N,T) N=cross-sections, T=observations
**          x2         = matrix (N,T) N=cross-sections, T0observations
**          min        = vector (2,1) left initial grid values for x1 and x2.
**                       if (-1|-1)  min = {minc(x1) | minc(x2)};
**          max        = vector (2,1) right final grid values for x1 and x2;
**                       if (-1|-1) max = {maxc(x1) | maxc(x2)};
**          M          = grid width.
**          auto_width = multiply the windows width automatically choosen by the
**                       constant autowidth.
**          &ker_func  = kernel method &ker2_gauss Gaussian kernel
**                                     &ker2_epan1         Epanechnikov kernel
**                                     &ker2_epan2 squared Epanechnikov kernel
**                                     &ker2_epan3 cubic   Epanechnikov kernel
**
** Output: f_joint = joint density distribution
**         step1   = grid used X1 variable
**         step2   = grid used X2 variable
**
*/
```

```
/*
** Procedure Condit
**
** Compute residuals from conditioning regression as in Quah (1996)
**
**  {y,res,b_coe,OLS_std,W_std,R2}=condit(y,x,lead,lag);
**
**  Input:  y          = dependent var.   (T,N) N=cross-sections, T=observations
**          x          = conditioning var (T,N) N=cross-sections, T=observations
**          lead       = number of leads
**          lag        = number of lags
**
**
** Output:           y  = new dependent variable (T-lead-lag,N)
**                 res  = residualds (T-lead-lag,N)
**               b_coe  = beta coefficients
**             OLS_std = OLS standard errors
**               W_std = White's heteroskedasticity corrected standard errors
**                  R2  = R_square coefficient
**
*/
```

```
/*
** Procedure Granger_test
**
** Compute Granger causality test : bivariate VAR
**
**  {S_y_x,p_y_x,S_x_y}=granger_test(y,x,p);
**
**  Input:  y          = dependent var.   (T,N) N=cross-sections, T=observations
**          x          = conditioning var (T,N) N=cross-sections, T=observations
**          p          = number of lags in the bivariate VAR
**
** Output:       S_y_x = F test on the null that x does not cause y
**               p_y_x = p value
**               S_x_y = F test on the null that y does not cause x
**               p_x_y = p value
**
*/
```

```
/*
** Procedure med_ker
**
** Compute medians from stochastic kernel density
**
**                   {med} = med_ker(st_ker,step);
**
**  Input:  st_ker     = stochastic kernel density matrix (M,M)
**          grid       = grid interval, (M,1) values
**
** Output:          med = median values (M,1)
**
*/
```