PANEL COINTEGRATION

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 "PANEL.PRG" and "PANEL.DEC" and "ANURAG.SRC" file in the c:\GAUSS\SRC directory.
3. Copy "PANEL.LCG" file in the c:\GAUSS\LIB directory

4.1 At the GAUSS prompt, before using PANEL routines, run the command "library panel;",

4.2 or include the command "library panel;" at the start of the batch file that uses the Panel Cointegration routines,

4.3 or insert the command "library panel;" in the GAUSS "startup" file, usually included in the GAUSS directory. (This is the best choice. Any time you run GAUSS the panel routines will be on line).

```
/*
*******************************************************************************
** PANEL UNIT ROOT TESTS and COINTEGRATION ANALYSIS
** written by Luciano Gutierrez, last version Dec. 2002
**
** This  code has  been developed  specifically for  use in panel unit root
** tests and panel cointegration 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: lgutierr@uniss.it
**
******* Unit Root Test Procedures :
**
** Proc :  Augmented Dickey-Fuller    test    adf_var
**         Ng and Perron (1996)       test    M_tests
**         Ng and Perron (2001)       test    M_gls
**         Kwiatkowski et al.(1992)   test    kpss
**         Leybourne and McCabe(1994) test    l_m
**         Elliott et al. (1996)      test    df_gls
**
******* AR(1) Autoregressive Parameter Estimation Procedure
**
** Proc :  Andrew (1993) LS and Median estimates   m_andrew proc
**                                                 m_search proc
**
******* Panel Unit Root Test Procedures :
**
** Proc :  Im,Pesaran and Shin (1997) test    IPS97
**         Levin and Lin (1993)       test    LL93
**         Bai and Ng (2001)(2003)    test    BN_2003
**         Moon and Perron (2003)     test    Moon_Perron
**         Phillips and Sul (2003)    test    G_ols_test
**         Phillips and Sul (2003)    test    G_emu_test
**         Choi (2002)                test    choi_test
**
******* Panel Estimation Procedures :
**
** Proc :  OLS estimation                     ols_panel
**         OLS corrected estimation           ols_panel_adj
**         FIML estimation                    fm_panel
**         Dynamic OLS estimation             dols_panel
**
******* Panel Cointegration Test Procedures :
**
** Proc : Pedroni(1995) homogeneous    test   coint_ped95
**        Pedroni(1999) heterogeneous  test   coint_ped99
**        Kao (1999)    homogeneous    test   coint_kao
**        Larsson et al (2001)  heter. test   coint_lll
**
******* Common Factors
**
** Proc : Bai and Ng (2001)(2003) procedures: BN_k    finds the number of common
**                                                   factors
**                                         BN_fact estimates common factors and
**                                                   residuals
**        Hodrick and Prescott (1997)     HP_filt
*******************************************************************************
*/
```

Unit Root Tests

```
/*
** adf_var
**
** Purpose: Compute the ADF test
**
** Format:  {beta,t_test,l} = adf_var(x,co,l);
**
** Input:   x       =  vector (Tx1) times serie
**          co      =  equal to -1 if no constant and trend
**                     equal to  0 a constant is included
**                     equal to  1 a constant and trend is included
**          l       =  l ge  0 : fixed lag l
**                  =  l eq -1 :  Schwert (1989) lag
**                  =  l eq -2 :  Ng and Perron (1995) procedure
**
** Output:  beta    =  coefficient estimate
**          t_test  =  t statistic value beta estimate
**          l       =  lag
**
*/
```

```
/*
** M_test
**
** Purpose: Compute Perron and Ng (1996) M unit root tests
**
**
** Format  :  {MZ_a,MSB,MZ_t} =  m_test(x,co,l_lag);
**
** Input:     x       = matrix(T,N) input data set
**            co      = equal  0 for a constant term
**                      equal  1 for a linear trend
**            l_lag   = adf lag (if -1 lag defined by Ng and Perron proc.)
** Output:    MZ_a    = vector (N,1) MZ_a test
**            MSB     = vector (N,1) MSB  test
**            MZ_t    = vector (N,1) MZ_t test
**
*/
```

```
/*
** M_GLS
**
** Purpose: Compute Ng and Perron (2001) MP_GLS unit root test
**
** Asymptotic Critical Values
** p=0 c_bar=-7;
** Percentile    MZ_a_GLS  MZ_t_GLS   MSB_GLS   PT_GLS,MPT_GLS
** 0.01           -13.8     -2.58      .174          1.78
** 0.05            -8.1     -1.98      .233          3.17
** 0.10            -5.7     -1.62      .275          4.45
**
** p=1 c_bar=-13.5;
** Percentile    MZ_a_GLS  MZ_t_GLS   MSB_GLS   PT_GLS,MPT_GLS
** 0.01           -23.8     -3.42      .143          4.03
** 0.05           -17.3     -2.91      .168          5.48
** 0.10           -14.2     -2.62      .185          6.67
**
** Format  :  {PT_GLS,MPT_GLS,MZ_a_GLS,MSB_GLS,MZ_t_GLS} = m_gls(x,co,l_lag);
**
** Input:     x        = matrix(T,N) input data set
**            co       = equal  0 for a constant term
**                       equal  1 for a linear trend
**            l_lag    = adf lag (if -1 lag defined by Ng and Perron proc.)
** Output:    PT_GLS   = vector (N,1) PT_gls   test
**            MPT_GLS  = vector (N,1) MPT_GLS  test
**            MZ_a_GLS = vector (N,1) MZ_a_GLS test
**            MSB_GLS  = vector (N,1) MSB_GLS  test
**            MZ_t_GLS = vector (N,1) MZ_t_GLS test
**
*/
```

```
/*
** KPSS
**
** Purpose: Compute Kwiatkowski, Phillips, Schmidt, Shin (1992) test.
** Test                      Critical values
**                            0.10   0.05   0.01
** Intercept only           0.347  0.463  0.739
** Linear trend             0.119  0.146  0.216
**
** Format  :  lm = kpss(x,co,k_lag);
**
** Input:     x       = matrix(T,N) input data set
**            co      = equal to -1 for no deterministic part
**                      equal to  0 for a constant term
**                      equal to  1 for a linear trend
**            k_lag   = number of lag spectral windows
** Output:    lm      = KPSS test
**
*/
```

```
/*
** L_M
**
** Purpose: Compute Leybourne and McCabe (1994) test.
**          This program needs Gauss Optimization Package
**
** Format  :  s_beta = l_m(x,p);
**
** Input:     x       = matrix(T,N) input data set
**            p       = number of lags in the AR process
**            co      = -1 no constant in the ARIMA process
**            co      =  0 constant in the ARIMA process
** Output:    s_beta  =  Leybourne and McCabe test
**
*/
```

```
/*
** DF_GLS
**
** Purpose: Compute Elliot, Rothemberg, Stock (1996) test
**
** Sample size             Critical values (linear trend model)
**                           0.10   0.05   0.01
**   50                     -2.89  -3.19  -3.77
**  100                     -2.74  -3.03  -3.58
**  200                     -2.64  -2.93  -3.46
**  inf                     -2.57  -2.89  -3.48
**
** Format  :  {beta_df,t_test,l_lag} = df_gls(x,co,l_lag);
**
** Input:     x       = matrix(T,N) input data set
**            co      = equal to  0 for a constant term
**                      equal to  1 for a linear trend
**            l_lag   = adf lag (if -1 lag defined by Ng and Perron proc.)
**
** Output:    beta_df = beta coefficient
**            t_test  = vector (N,1) t_test
**            l_lag   = lag used
**
*/
```

```
/*
** Vratio
**
** Purpose: Compute Cochrane(1988) variance ratio statistic and Lo MacKinlay
**          (1988) test
**
** Format:  {v,ML,z} = vratio(y,s);
**
** Input:   y      =  Tx1 vector
**          s      =  number of variance statistics
**
** Output:  v      =  sx1 variance ratio statistics
**          z      =  Lo and MacKinlay (1988) variance ratio test
**
*/
```

AR(1) Autoregressive Parameter Estimation

```
/*
** M_Andrew
**
** Purpose: Compute Andrew (1993) median_estimate AR(1) parameter
**
** Format  :  {a_LS,a_U,res_a_U} = m_andrew(y,co,q_p);
**
** Input:     y       = vector(T,1) input data set
**            co      = -1 no  constant
**                       0 for a constant term
**                       1 for a linear trend
**            q_p     = quantile at probability p (ex. 0.5 to estimate median)
**
** Output:    a_LS    = Least square estimate
**            a_U     = Median estimate when q_p is used
**            res_a_U = vector((T-1),1) residuals
**
** i.  Table I-II-III in Andrew(1993) can be reproduced by using m_search proc
** ii. Some probelm in the procedure when a_LS is near -1
** iii.The procedure utilizes Imhof(1961) quadratic function.
**     The GAUSS routine is supplied by Banerjee at GAUSS applications
*/
```

```
/*
** M_Searc1
**
** Purpose: Reproduce Andrew's (1993) tables I-II-III
**
** Format  :    q(a_hat) = m_searc1(T,co,a,q_p);
**
** Input:     T       = number of observation
**            co      = -1 no  constant
**                       0 for a constant term
**                       1 for a linear trend
**            a       = autoregressive parameter
**            q_p     = quantile at probability p (ex. 0.5 to estimate median)
**
** Output:    q(a_hat)= quantile estimate when q_p is used
**
*/

/*
** M_Searc2
**
** Purpose : Compute Andrew (1993) median estimator
**
** Format  :    {a_U} = m_searc2(T,co,a,q_p);
**
** Input:     T       = number of observation
**            co      = -1 no  constant
**                       0 for a constant term
**                       1 for a linear trend
**            a       = autoregressive parameter
**            q_p     = quantile at probability p (ex. 0.5 to estimate median)
**
** Output:    a_U     = Autoregressive estimate when q_p is used
**
*/
```

Panel Unit Root Tests

```
/*
** Pool_KPSS
**
** Purpose: Compute Pooled KPSS (2001) tests as presented in Bai and Ng (2001)
**
** Format  :  {chi_2,nr}lm = pool_kpss(x);
**
** Input:     x       = matrix(T,N) input data set
**
** Output:    chi_2   = -2sum(log(pval)) statistic
**            nr      = (pval-2N)/sqrt(4N) statistic
**
*/
```

```
/*
** Pool_ADF
**
** Purpose: Compute Pooled ADF tests
**
** Format  :  {beta,t_test,chi_2,nr} = pool_adf(y,co,l);
**
** Input:     y       =   TxN dataset
**            co      =   -1 no drift parameter
**                    =    0 constant in the regression
**                    =    1 linear trend
**            l       =    l ge 0  fixed lag l
**                    =   -1  lag = 4*ceil(minc(N | T)./T)^(1/4)
**                    =   -2  Ng and Perron procedure
** Output:    beta    =  Nx1 vector of autoregressive coefficients
**            t_test  =  Nx1 vector of DF t_tests
**            chi_2   = -2*sumc(ln(pval)) statistic
**            nr      = (chi_2-2N)/sqrt(4N) statistic
**
*/
```

```
/*
** Pool_DFGLS
**
** Purpose: Compute Pooled DF_GLS tests
**
** Format  :  {beta,t_test,chi_2,nr} = pool_dfgls(y,co,l);
**
** Input:     y       = TxN dataset
**                    =  0 constant in the regression
**                    =  1 linear trend
**            l       =   l ge 0 fixed lag
**                    =  -1  lag = 4*ceil(minc(N | T)./T)^(1/4);
**                    =  -2  Ng and Perron procedure
** Output:    beta    =  Nx1 vector of autoregressive coeffcients
**            t_test  =  Nx1 vector of DF t_tests
**            chi_2   = -2*sumc(ln(pval)) statistic
**            nr      = (chi_2-2N)/sqrt(4N) statistic
**
*/
```

```
/*
** IPS97
**
** Purpose : provide t-statistic as in Im et al. (1997)
**           equation 5.3 pg. 10
**
** Format  :  t = IPS97(y_t,co,l,mu,sig);
**
** Input:     y       = matrix(T,N) input data set
**            co      = equal to -1 for no deterministic part
**                      equal to  0 for a constant term
**                      equal to  1 for a constant + trend
**            l       = lag order ADF process
**            mu      = mean adjustment, see Im et al. (1997) Table 2
**            sig     = standard error adj, see Im et al. (1997) Table 2
**                      (note that Im et al. provide variance instead std).
**
** Output:  -  t-statistic see equation 5.3 pg. 10 in Im et al. (1997)
**
*/
```

```
/*
** LL93
**
** Purpose : provide t_statistic as in Levin and Lin (1993)
**           equation 16 pg.13
**
** Format  :  t = LL93(y_t,co,l,mu,sig);
**
** Input:     y       = matrix(T,N) input data set
**            co      = equal to -1 if no constant and trend
**                      equal to  0 a constant is included
**                      equal to  1 a constant and trend is included
**            l       = -1 the program chooses the best lag for each y_it
**                      = > or = 0 to fix the same lag for each y_it
**            mu      = mean adjustment, see Levin and Lin (1993) Table 2
**                      pg. 33
**            sig     = standard error adj, see Levin and Lin (1993) Table 2
**                      pg. 33
**
** Output:  -  t-statistc, see equation 16 pg. 13 in Levin and Lin (1993).
**
*/
```

Panel CointegrationTests

```
/*
** coint_ped95
**
** Purpose :  compute panel cointegration Pedroni(1995) homogenous tests
**
** Format  : {PC_1,PC_2} = coint_ped95(res,y,x)
**
** Input:     res     = vector(T*N) residuals from coint-regression
**            y       = matrix(T,N)    dependent variable
**            x       = matrix(T,N*K)  indipendent K variables
**
** Output:  PC_1 test
**          PC_2 test
*/
```

```
/*
** coint_ped99
**
** Purpose :  compute panel cointegration Pedroni(1999) heterogeneous tests
**
** Format  :  {t_1,t_2,t_3,t_4,t_5,t_6,t_7}=coint_ped99(res,y,x,c,n_lag)
**
** Input:     res     = vector(T*N) residuals from coint-regression
**            y       = matrix(T,N)    dependent variable
**            x       = matrix(T,N*K)  indipendent K variables
**            c       = -1 no deterministic component
**                       0 individual constant in the cointegrated regression
**                       1 individual constant and trend in the coint. regress.
**            n_lag   = number of lags in the ADF regression
**
**
** Output:    adjusted Panel v-statistic
**            adjusted Panel rho-statistic
**            adjusted Panel t-statistic (non-parametric)
**            adjusted Panel t-statistic (parametric)
**            adjusted Group rho-statistic
**            adjusted Group t-statistic (non-parametric)
**            adjusted Group t-statistic (parametric)
**
*/
```

```
/*
** coint_kao
**
** Purpose :  compute cointegration Kao(1999) homogeneous tests
**
** Format  :  {t_1,t_2,t_3,t_4,t_5}=coint_kao(res,y,x,c,n_lag)
**
** Input:     res     = vector(T*N) residuals from coint-regression
**            y       = matrix(T,N)    dependent variable
**            x       = matrix(T,N*K)  indipendent K variables
**            c       = -1 no deterministic component
**                       0 individual constant in the cointegrated regression
**                       1 individual constant and trend in the coint. regress.
**            n_lag   = number of lags in the ADF regression
**
** Output:    Panel DF_rho statistic
**            Panel DF_t   statistic
**            Panel DF_star_rho statistic
**            Panel DF_star_t statistic
**            Panel ADF statistic for lag = n_lag
**
*/
```

```
/*
** coint_lll
**
** Purpose : provide LR cointegration tests, Larsson et al. (2001)
**           equation 12 pg. 112.
**
** Format  :  Ltrace = coint_lll(y,x,co,lagmax);
**
** Input:     y       = matrix(T,N) dependent variable
**            x       = matrix(T,N*K) independent variables
**            co      = equal to -1 for no deterministic part
**                      equal to  0 for a constant term
**                      equal to  1 for a constant + trend
**            lagmax  = maximum lag in the VAR process
**
** Output:  -  LR-statistic
**
*/
```

Panel Cointegration Estimation Methods

```
/*
** ols_panel
**
** Purpose :  compute ols conventional slope panel estimates and its
**            t_test statistics, p_values and R_square.
**
** Format  : {beta,t_test,res} = ols_panel(y,x,c)
**
** Input:     y       = matrix(T,N)    dependent variable
**            x       = matrix(T,N*K)  indipendent K variables
**            c       = -1 no constant in the model
**                    =  0 constant in the model
**                    =  1 constant plus trend in the model
** (i)  N = cross-sections, T=time dimension
**
** Output:    beta    =  beta_coefficients
**            t_test  =  t_test statistics
**            res     =  vector of residuals T*N
*/
```

```
/*
** ols_panel_adj
**
** Purpose :  compute ols adjusted slope panel estimates and its
**            t_test statistics, p_values and R_square.
**
** Format  :  {beta,t_test,res} = ols_panel_adj(y,x,c)
**
**
** Input:     y       = matrix(T,N)    dependent variable
**            x       = matrix(T,N*K)  indipendent K variables
**            c       = -1 no constant in the model
**                    =  0 constant in the model
**                    =  1 constant plus trend in the model
** (i)  N = cross-sections, T=time dimension
**
** Output:    beta    =  beta_coefficients
**            t_test  =  t_test statistics
**            res     =  vector of residuals T*N
**
*/
```

```
/*
** FM_panel
**
** Purpose :  compute FIML slope panel estimates and its
**            t_test statistics, p_values and R_square.
**
** Format  :  {beta,t_test,res} = fm_panel(y,x,c)
**
** Input:     y       = matrix(T,N)    dependent variable
**            x       = matrix(T,N*K)  indipendent K variables
**            c       = -1 no constant in the model
**                    =  0 constant in the model
**                    =  1 constant plus trend in the model
** (i)  N = cross-sections, T=time dimension
**
** Output:    beta    =  beta_coefficients
**            t_test  =  t_test statistics
**            res     =  vector of residuals T*N
*/
```

```
/*
** dols_panel
**
** Purpose :  compute dynamic OLS slope panel estimates and its
**            t_test statistics, p_values and R_square.
**
** Format  :  {beta,t_stat,res} = dols_panel(y,x,c,n_lead,n_lag)
**
** Input:     y       = matrix(T,N)    dependent variable
**            x       = matrix(T,N*K)  indipendent K variables
**            c       = -1 no constant in the model
**                    =  0 constant in the model
**                    =  1 common trend in the model
**            n_lead  = max lead variable
**            n_lag   = max lag variable
**
** (i)  N = cross-sections, T=time dimension
**
** Output:    beta    =  beta_coefficients
**            t_stat  =  t_test statistics
**            res     =  vector of residuals T*N
*/
```

Panel Unit Root Tests for Cross-Sectionally Correlated Panels

```
/*
** BN_2003
**
** Purpose : Calculate tests as in Bai and Ng (2003)
**
** Format: {t_e,t_f,P_val,Adf_pool,GLS_pool,Y_c,MQ}=bn_2003(x,co,nfac,l);
**
** Input:     x       = matrix(T,N) input data set
**                      equal to  0 for a constant term
**                      equal to  1 for a linear trend
**            nfac    = number of factors
**            l       =    l ge 0  fixed lag l
**                    =   -1  lag = 4*ceil(minc(N | T)./T)^(1/4)
**                    =   -2  Ng and Perron procedure
**
** Output:    t_e     = vector(N,1) ADF test idiosyncratic components
**            t_e1    = vector(N,1) DFGLS test idiosyncratic components
**            t_f     = scalar ADF test common factor (when -9999 not computed)
**            P_ADF   = Chi2(N) ADF   panel unit root test
**            P_GLS   = Chi2(N) DFGLS panel unit root test
**            ADF_pool= N(0,1) ADF panel unit root test
**            GLS_pool= N(0,1) DFGLS panel unit root test
**            Y_c     = vector common factor(s).
**            MQ      = MQ_c_c and MQ_c_f tests (when -9999 not computed)
*/

/*
** BN_k
**
** Purpose : Calculate the number of factors as in Bai and NG (2000)
**
** Format  :  {r_PC1,r_PC2,r_PC3,r_IC1,r_IC2,r_IC3,r_AIC1,r_AIC2,r_AIC3,r_BIC1,
**             r_BIC2,r_BIC3) = BN_k(x,co_mx_fac);
**
** Input:     x       = matrix(T,N) input data set
**            max_fac = maximum number of factors
**
** Output:    Number of factors for PC statistics,  IC statistics,
**                                AIC statistics, BIC statistics
**
*/

/*
** BN_fact
**

** Purpose : Calculate factor and residual matrices as in Bai and Ng  (2001)
**
** Format  : {f_hat,e_hat,df_hat,z_hat,lambda,eigval}=bn_fact(x,co,nfac);
**
** Input:     x       = matrix(T,N) input data set
**                      equal to  0 for a constant term
**                      equal to  1 for a linear trend
**            nfac    = number of factors
**
** Output:    F_hat   = matrix(T-1,nfac) of cumulated factors
**            e_hat   = matrix(T-1,N) of cumulated residuals
**            df_hat  = matrix(T-1,nfac) of differenced factors
**            z_hat   = matrix(T-1,N) of differenced residuals
**            lambda  = vector(N,1) of factor loading
**            eigval  = eigenvalues.
*/
```

```
/*
** Moon_Perron
**
** Purpose : Compute Panel unit root tests as in Moon and Perron (2003)
**
** Format  :  {rho_ols,rho_star_pool,t_star_a,t_star_b} = Moon_Perron(x,nfac);
**
** Input:     x        = matrix(T,N) input data set
**            nfac     = number of factors
**
** Output:    rho_pool = rho ols value
**        rho_star_pool= corrected rho value
**            t_star_a = t_star_a test
**            t_star_b = t_star_b test
*/
```

```
/*
** G_ols_test
**
** Purpose : Compute Panel unit root tests as in Phillips and Sul (2002)
**
** Format  :  {G_ols_plus,P,Z} = G_ols_test(y,co,l,m_eps,sig_eps);
**
** Input:     y          = matrix(T,N) input data set
**            co         = equal to -1 for no deterministic part
**                         equal to  0 for a constant term
**                         equal to  1 for a linear trend
**            l          =    l ge 0  fixed lag l
**                       =   -1  lag = 4*ceil(minc(N | T)./T)^(1/4)
**                       =   -2  Ng and Perron procedure
**            mu_eps     = mean value epsilon
**            sig_eps    = std error epsilon
**
** Output:    G_ols_plus = G_ols_plus_plus test, N(0,1) distributed
**                     P = P test, N(0,1) distributed
** Output:             Z = Z test, N(0,1) distributed
**
*/

/*
** G_emu_test
**
** Purpose : Compute Panel unit root tests as in Phillips and Sul (2002)
**
** Format  :  {G_emu_plus,P,Z} = G_emu_test(y,co,m_eps,sig_eps);
**
** Input:     y          = matrix(T,N) input data set
**            co         = equal to -1 for no deterministic part
**                         equal to  0 for a constant term
**                         equal to  1 for a linear trend
**            mu_eps     = mean value epsilon
**            sig_eps    = std error epsilon
**
** Output:    G_emu_plus = G_emu_plus_plus test, N(0,1) distributed
**                     P = P test, N(0,1) distributed
** Output:             Z = Z test, N(0,1) distributed
**
*/
```

```
/*
** Choi_test
**
** Purpose : Compute Panel unit root tests as in Choi (2002)
**
** Format  :  {P,Z,L} = choi_test(y,co,l);
**
** Input:     y          = matrix(T,N) input data set
**            co         = equal 0 for a constant term
**                         equal 1 for a linear trend
**            l          = l ge 0  fixed lag l
**                       = -1  lag = 4*ceil(minc(N | T)./T)^(1/4)
**                       = -2  Ng and Perron (1995) procedure
**
** Output:    P          = P chi square test
**            Z          = inverse normal test
**            L          = logit test
*/
```

Some Routines

```
/*
** HP_filt
**
** Purpose: Compute Hodrick and Prescott's (1997) filter
**
** Format:  {shp} = hp_filt(x,w);
**
** Input:   x       =  vector (Tx1) times serie
**          w       =  weights (100 or 6.25 annual series, 1600 quarterly
**
** Output:  shp     =  Hodrick and Prescott's trend
**
*/
```

```
/*
** p_val
**
** Purpose: Compute p_value ADF and DF_GLS tests following the
**          MacKinnon (1994) procedure
**
** Format:  p_value = p_val(T,t_test,cs);
**
** Input:   T       =  number of observation in the time serie.
**          t_test  =  value of t_test
**          cs      =  1 ADF test no constant and trend in the process
**                  =  2 ADF test only constant in the process
**                  =  3 ADF test constant and trend in the process
**                  =  4 DF_GLS test only constant in the process
**                  =  5 DF_GLS test constant and trend in the process
**
** Output:  p_value =  p_value of test
**
*/
```

```
/*
** d_trend
**
** Purpose : detrend series
**
**
** Format  : {d_y_t} = d_trend(y_t,p);
**
** Input:     y_t    = vector (T,1)
**              p    = polinomial order;
**
** Output: -  d_y_t  = detrended serie (T,1)
**
*/
```

```
/*
** l_dtrend
**
** Purpose : locally detrend series
**
**
** Format  : {y_t,ssr} = l_dtrend(x_t,c);
**
** Input:       x_t  = vector (T,1)
**              c    = -1 no deterministic component
**                   =  0 only constant
**                   =  1 linear trend
**
** Output: -    y_t  = locally detrended series (T,1)
**              ssr  = sum squared residuals
**
*/
```