PESTS
Poisson Estimators for State-Space Time Series
Version 1.0

Patrick T. Brandt and John T. Williams
Department of Political Science
210 Woodburn Hall
Indiana University
Bloomington, IN 47405
E-mail: pbrandt@indiana.edu

August 10, 1999

Abstract

This manual briefly describes the implmentation of the state-space approach to event count and other non-normal data of Brandt, Williams, Fordham (1998), Brandt and Willams (1998a,b). It outlines the GAUSS code and discusses the basic estimation issues.

Contents

1 Introduction

This paper contains a description of the implementation of the state-space approach to event count and non-normal data (Brandt and Williams 1998a, Brandt and Williams 1998b and Brandt, Williams and Fordham 1998). These models can be used to estimate time series models for count data based on an extended Kalman filter. These models provide a generalization of the basic Poisson or event count processes, since they allow one to estimate and test the presence of time series dynamics in event count data. The code contained in this package allows you to estimate both Poisson exponentially weighted moving average models (PEWMA) and the Poisson AR(p) (PAR(p)) models.

Included with this paper should be files of code to implement the models:

`*.g`	Gauss code for each procedure.
`pests.lcg`	Gauss library file to load the PESTS programs
`readme.txt`	A brief reference file.

A description of the basic programs and syntax is included below. It is advised that you have a copy of the original papers, Brandt and Williams (1998a), Brandt and Williams (1998b), and Brandt, Williams and Fordham (1998) for reference. Also, readers should be familiar with state-space models or have access to a good reference such as Harvey (1991).

Detailed descriptions of the statistical estimation performed by these programs are contained in the papers:

Brandt, Patrick, John T. Williams and Benjamin Fordham. 1998. "Modeling Time Series Count Data: A State-Space Approach to Event Counts." presented at the 1998 Society for Political Methodology Summer Meeting, San Diego, California, July 22-26, 1998.

Brandt, Patrick T. and John T. Williams. 1998a. "Dynamic Modeling for Persistent Time Series of Event Counts."

Brandt, Patrick T. and John T. Williams. 1998b. "A Linear Poisson Autoregressive Model: the Poisson AR(p) Model"

IF YOU USE THESE PROGRAMS, PLEASE CITE THESE PAPERS!!

This paper and the code it describes adopt the following conventions:

1. Code, programmed procedures, and functions are written in the `typewriter` font.
2. **The GAUSS code provided is completely self-contained. That means that all you need is a copy of GAUSS and the GAUSS maximum-likelihood module MAXLIK (version 3 or 4) to estimate the models.**

This code is provided with no guarantees and no claims as to its accuracy. It is intended for non-commercial, academic use only. If you find any errors or have questions about the code, please send me an e-mail at pbrandt@indiana.edu, or check my web page, http://php.indiana.edu/~pbrandt for updates.

2 Installation

These files should be installed in the GAUSS path. Normally, this is the directory "`\Gauss\src`". Alternatively, you may install the programs in another directory. As written however, all the programs should be installed in the same directory.

These are the files to be installed:

1. `AICLLF.G` – procedure for computing log-likelihood and AIC values
2. `BETARNG.G` - procedure for Beta distributed random numbers
3. `EWMA.G` - procedure for computing EWMA forecast at time t.
4. `EXAMPLE1.G` - an example of the PEWMA model
5. `LNB.G -` procedure for the log-likelihood of the negative binomial regression model.
6. `LNGM.G` – procedure for computing the log gamma function

7. `LPARP.G` – procedure for the log-likelihood of the PARP model
8. `LPG.G` – procedure for the log-likelihood of the PEWMA model
9. `LPSN.G` – procedure for the log-likelihood of the Poisson model
10. `PARP.G` – procedure to estimate the PARP by maximum likelihood
11. `PARPDGP.G` – procedure for generating PARP simulated data
12. `PARPFILT.G` – procedure for filtering PAR(p) model
13. `PEWMA.G` – procedure to estimate PEWMA by maximum likelihood
14. `PEWMADGP.G` - procedure for generating PEWMA simulated data
15. `PMAFILT.G` – procedure for filtering PEWMA model
16. `POISSON.G` - procedure to estimate a Poisson regression
17. `PSI.G` – procedure to compute psi/derivative of log gamma function
18. `RNDP.G` – an improved Poisson random number generator (Baird)

If you install the programs in another directory, be sure to include that directory in your Gauss path in the gauss.cfg file.

Installation Steps:

1. Copy all the `*.g` and `*.txt` files to the `gauss\src` directory on your machine.
2. Place the library `pests.lcg` file in `gauss\lib`
3. Include the following line in any code that will be using PESTS routines:

```
library pests,maxlik,pgraph;
```

You are now read to run PESTS.

3 Description of the procedures

The GAUSS code in these files is intended to estimate two different time series models for count data. The two models are the Poisson exponentially weighted moving average model (PEWMA) and the Poisson AR(p) (PAR(p)). These are structural time series models based on an extended Kalman filter. The code in the files `pewma.g` and `parp.g` contain the main estimation routines. These estimate the PEWMA and PARP respectively. The other files are used by these two procedures to either 1) filter the data prior to estimation, 2) calculate numerical functions, or 3) generate simulated series of PEWMA or PARP data.

3.1 Filtering procedures

Two filtering procedures are provided. The first is `PMAFILT.G`. This is a filter for the PEWMA model. It computes filtered estimates of the state vector parameters for an EWMA process. It generates the values of the prior for each period, t. This procedure is called for each iteration of the maximum-likelihood estimation of the PEWMA model.

The second filter is `PARFILT.G`. This is a filter for the PAR(p) model. It computes filtered estimates of the state vector parameters for an AR(p) process. It generates the values of the prior for each period, t. This procedure is called for each iteration of the maximum-likelihood estimation of the PAR(p) model.

3.2 Estimation procedures

There are two main estimation procedures. The first is `PEWMA.G`. This procedure is used to estimate PEWMA models. The second is `PARP.G`. This procedure is used to estimate PAR(p) models. The procedures are intended to be very easy to use and work like simple regression procedure, despite the complexity of the PEWMA and PAR(p) estimation routines.

Each program has the following basic syntax:

```
(coefficients, covariance matrix, log-likelihood function, AIC) =
              model(y, x, p, starting value method)
```

where the **inputs** are

1. y = $T \times 1$ vector, the dependent variable of counts
2. x = $T \times K$ vector, the regressors
3. p = order of the autoregression, for the PAR(p) model
4. sv = scalar, starting value code. Set this to "P" for Poisson regression starting values and "OLS" for logged OLS. User supplied starting values may also be included by the specification of a conformable vector of starting values.

The **outputs** are

1. coefficients = time series parameters and regression coefficients
2. covariance matrix = estimated covariance matrix
3. log-likelihood function = final estimate of log-likelihood function at the optimum
4. AIC = Akaike's Information Criterion

In addition, each procedure generates a print out by default. It includes,

1. Regression estimates, standard errors and t-statistics.

2. Final likelihood and AIC values.

The procedures themselves contain additional details about their construction and the input of data. The basic syntax is illustrated in the following two examples. Each assumes that y is a vector of counts and x is a vector of regressors that have already been declared or read into memory. This can be done before the call to the estimator in each example.

3.2.1 Example 1: A PEWMA model

Here is a sample PEWMA program.

```
library maxlik,pests,pgraph;              /* loads libraries                    */
                                          /* data loading steps could be here   */
output file = pewma.out;                  /* declare and open an output file    */
output on reset;
{omega,d,cov,llf,aic} = pewma(y,x,``P'');/* estimates a PEWMA regression with  */
                                          /* Poisson regression starting values */
output off;
end;
```

3.2.2 Example 2: A PAR(p) model

Here is a sample PAR(p) program.

```
library maxlik,pests,pgraph;              /* loads libraries                    */
                                          /* data loading steps could be here   */
output file = parp.out;                   /* declare and open an output file    */
output on reset;
{rho,d,cov,llf,aic} = parp(y,x,2,``P''); /* estimates a PAR(2) regression with */
                                          /* Poisson regression starting values */
output off;
end;
```

3.3 Other useful procedures

Included with this code are several additional procedures. These include two "generators" for time series of counts. Also included are a series of numerical procedures used by the likelihood functions and the random number generators.

The two "generators" produce PEWMA and PAR(p) series. Simulated PEWMA series can be generated using the procedure `pewmadgp`. Simulated PAR(p) series can be generated using `parpdgp`. The details of these procedures can be found by looking at the header for each file.

The additional procedures are numerical. These include random number generators for the Beta and Poisson distributions, a procedure to compute an accurate approximation of the logarithm of the gamma function, a procedure to compute an accurate approximation of the

first derivative of the logarithm of the gamma function, and a procedure to compute EWMA's for the PEWMA filter. These procedures are used in the filters and the "generators."

These numerical and random number procedures are the best we have been able to find or construct. We have tried to make them as reliable as possible, since some of them are defined only over limited domains. This is especially true for the two procedures based on the log-gamma function. The log-gamma procedure `lngm.g` is based on an approximation found in Press et .al (1986). It seems subject to fewer underflow and math co-processor errors than the `lnfact` procedure in GAUSS.

The Beta random number generator is the one written by Mooney (1997).

The Poisson random number generator is different from that shipped with GAUSS. The procedure is from a collection of random number generators by Dr. David Baird. It uses a much faster acceptance-rejection method than that used in the GAUSS procedure. The increased speed is based the fact that the mode of the Poisson distribution can be easily found from the mean. Since Poisson random numbers are characterized by this value, it uses the mode to find the starting point when searching for a random number for a given probability.

3.4 A note on the PEWMA model output

The first coefficient reported in the PEWMA procedure is for the hyperparameter ω. The PEWMA nests the Poisson regression and is exactly the Poisson model when $\omega = 1$. So the t-test reported, based on the null that omega is zero is incorrect. We have included a new t-test at the bottom of the print out for the PEWMA code to generate a test for $\omega < 1$.

4 Using GAUSS' MAXLIK

This section is mainly provided for people who are only occasional users of GAUSS, or have never used the MAXLIK module. That said, even experienced users may learn from what is in the next few sections.

4.1 Optimization Algorithms

The PEWMA and PAR(p) estimation routines do NOT use analytical derivatives. In part, this is because the optimization of the likelihood takes a very long time if analytical derivative routines are used. While preferable on theoretical grounds, they make the present optimization routines very slow. The reason for the loss of speed is that the data must be filtered twice for each observation: once to compute the value of the objective function and a second time to compute the optimal step direction using an analytical derivative. The numerical derivative routines supplied with GAUSS appear to do a good job in most cases, based on Monte Carlo evidence (Brandt, Williams and Fordham 1998).

That said, not every optimization algorithm works equally well for the PEWMA or PAR(p). Based on many runs (about 10,000 estimated models), it appears that robust quasi-Newton method such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) or the gradient method suggested by Berndt, Hall, Hall, and Hausman (BHHH) perform well. For speed,

we recommend the BFGS method. However, in some cases the numerical Hessian used by this method will be ill-conditioned. In this case, the BHHH method is rather robust since it does not use numerical second derivatives.

If changing the algorithm does not improve the performance of the optimization, then different starting values may be needed. The PEWMA and PAR(p) routines allow for three different sets of starting values, so try them all: Poisson regression, logged-OLS, and a Gaussian EWMA (for the PEWMA). If that does not work, try replacing the starting value of the dynamic parameter in the likelihood call for the PEWMA or PAR(p) optimization.

If the basic estimation procedures, `pewma()` and `parp()`bog down or cannot estimate the likelihood, then you may need to program the optimization directly using alternative starting values for the parameters. This can be done using the `lpg.g` and `lparp.g` functions. These compute the PEWMA and PAR(p) likelihood functions respectively. The syntax for doing this is

```
 {b,f,g,cov,ret} = maxlik(y~x,0,&lpg,sv);
{b,f,g,cov,ret} = maxlik(y~x,0,&lparp,sv);
```

where the left hand side of the statement are the standard Maxlik outputs and the likelihood functions are included in the right hand side maxlik statement.

4.2 Adding options to the procedures

All MAXLIK options are available with this code. This includes setting optimization algorithms, step methods and routines for computing covariance matrices (see GAUSS documentation),

Probably the most useful tool not included in this code is adding variable names to the PEWMA and PAR(p) output. This can be done as follows. Suppose you have a matrix of regressors X that is $T \times K$. Then, you can define variable names for the PEWMA and PAR(p) as follows:

For the PEWMA include the variable definition line:

```
__altnam = {''Omega'', ''X1'', ''X2'',..., ''XK''};
```

For the PAR(p) include the variable definition line:

```
__altnam = {''X1'', ''X2'',..., ''XK'',''Rho1'', ''Rho2'',..., ''RhoP''};
```

These variable names should be declared before the procedure is run.

5 An Example and Model Interpretation

This section briefly outlines an example of how the PEWMA model can be estimated. The example we have included in the PESTS package is the data and models presented in Brandt and Williams (1998a). In this paper, we conducted an intervention analysis to determine the effect of the 1953 Supreme Court term on the number of cases the Court heard on civil rights and economic regulation issues, using data from Pacelle (1991, 1995).

The example is contained in the file `example1.g`. This file reads in the Supreme Court Case agenda series and estimates a series of models. The results of these models are then compared by computing the predicted series and the prediction errors. In this example, we show how the PEWMA models are constructed for the various different intervention periods. We hypothesize that the effects of the 1953 term (i.e. the ascendance of Earl Warren to Chief Justice; Brown v. Board, etc.) may have a delayed effect on changes in the Court's agenda. To assess this, we specify one to ten period leads for the effect of the 1953 term. We then choose the model with the highest log-likelihood / smallest AIC value. These models are estimated in the program and the results displayed in a set of tables.

The basic program has been written to run under MAXLIK 3.0. If you use MAXLIK 4.0, you will need to modify the globals in the file accordingly. The example file (with comments) is an follows:

```
/* EXAMPLE 1: Event Count Time Series Analysis of Pacelle's Supreme Court
** Agenda Data
**
** This file is an example of how to estimate PEWMA
** count models using the PESTS code.
**
** April 20, 1998
**
*/

/* Load the required libraries and initialize the workspace */
new;
library maxlik, pgraph, pests;
format 8,6;

/* Read the data from the ASCII data file */
load dta[61,3] = sctdta.txt;
term = dta[.,1]; @ Court Term Variable @
equal = dta[.,2]; @ Number of equality cases @
reg = dta[.,3]; @ Number of regulation cases @

/* Create Brown v. Board leads interventions */
brown = zeros(20,1)|1|zeros(40,1);
brown1 = zeros(21,1)|1|zeros(39,1);
brown2 = zeros(22,1)|1|zeros(38,1);
```

```
brown3 = zeros(23,1)|1|zeros(37,1);
brown4 = zeros(24,1)|1|zeros(36,1);
brown5 = zeros(25,1)|1|zeros(35,1);
brown6 = zeros(26,1)|1|zeros(34,1);
brown7 = zeros(27,1)|1|zeros(33,1);
brown8 = zeros(28,1)|1|zeros(32,1);
brown9 = zeros(29,1)|1|zeros(31,1);
brown10 = zeros(30,1)|1|zeros(30,1);
brown11 = zeros(31,1)|1|zeros(29,1);

/* Turn on output file */
output file = court.out;
output on reset;

/* Fit models for Equality series:
** 1) Fit Models to determine optimal lag for intervention
** 2) Generate the predictions for best model
**
** Note that the previous models values are used as starting values
** in the next model to speed things up.
*/

/* PEWMA model for Equality series */
_mlalgr = 6; @ Sets ML optimization algorithm @
 @ to BHHH in MAXLIK 3.0 @

_mlparnm = {''Omega'',''BvBp''}; @ Sets parameter names @
{o0,d0,cov0,llf0,aic0} = pewma(equal,brown,''P'');
_mlparnm = {''Omega'',''BvBp-1''};
{o1,d1,cov1,llf1,aic1} = pewma(equal,brown1,o0|d0);
_mlparnm = {''Omega'',''BvBp-2''};
{o2,d2,cov2,llf2,aic2} = pewma(equal,brown2,o1|d1);
_mlparnm = {''Omega'',''BvBp-3''};
{o3,d3,cov3,llf3,aic3} = pewma(equal,brown3,o2|d2);
_mlparnm = {''Omega'',''BvBp-4''};
{o4,d4,cov4,llf4,aic4} = pewma(equal,brown4,o3|d3);
_mlparnm = {''Omega'',''BvBp-5''};
{o5,d5,cov5,llf5,aic5} = pewma(equal,brown5,o4|d4);
_mlparnm = {''Omega'',''BvBp-6''};
{o6,d6,cov6,llf6,aic6} = pewma(equal,brown6,o5|d5);
_mlparnm = {''Omega'',''BvBp-7''};
{o7,d7,cov7,llf7,aic7} = pewma(equal,brown7,o6|d6);
_mlparnm = {''Omega'',''BvBp-8''};
{o8,d8,cov8,llf8,aic8} = pewma(equal,brown8,o7|d7);
_mlparnm = {''Omega'',''BvBp-9''};
```

```
{o9,d9,cov9,llf9,aic9} = pewma(equal,brown9,o8|d8);
_mlparnm = {''Omega'',''BvBp-10''};
{o10,d10,cov10,llf10,aic10} = pewma(equal,brown10,o9|d9);
_mlparnm = {''Omega'',''BvBp-11''};
{o11,d11,cov11,llf11,aic11} = pewma(equal,brown11,o10|d10);
print;
print ''-----------------------------------------------------------------'';
print ''PEWMA estimates for equality series with interventions'';
print ''-----------------------------------------------------------------'';
print ''Intervention period Omega I(t) Log-lik AIC'';
print '' 1953 '';;o0~d0~llf0~aic0;
print '' 1954 '';;o1~d1~llf1~aic1;
print '' 1955 '';;o2~d2~llf2~aic2;
print '' 1956 '';;o3~d3~llf3~aic3;
print '' 1957 '';;o4~d4~llf4~aic4;
print '' 1958 '';;o5~d5~llf5~aic5;
print '' 1959 '';;o6~d6~llf6~aic6;
print '' 1960 '';;o7~d7~llf7~aic7;
print '' 1961 '';;o8~d8~llf8~aic8;
print '' 1962 '';;o9~d9~llf9~aic9;
print '' 1963 '';;o10~d10~llf10~aic10;
print '' 1964 '';;o11~d11~llf11~aic11;
print ''-----------------------------------------------------------------'';
print;
print;

/* Now generate the fitted series for the best model */
{a,aa,aaa,b,bb,bbb,mu,r} = pewma_filter(equal,brown10,o10,d10);
eqpma = trimr(aa,1,0)./trimr(bb,1,0);

/*-----------------------------------------------------------------*/
/* Fit models for Regulation series:
** 1) Fit Models to determine optimal lag for intervention
** 2) Generate the predictions for best model
*/

print ''Econ reg, PEWMA and temp. intervention'';
print;
_mlparnm = {''Omega'',''BvBp''};
{o0,d0,cov0,llf0,aic0} = pewma(reg,brown,''P'');
_mlparnm = {''Omega'',''BvBp-1''};
{o1,d1,cov1,llf1,aic1} = pewma(reg,brown1,o0|d0);
_mlparnm = {''Omega'',''BvBp-2''};
{o2,d2,cov2,llf2,aic2} = pewma(reg,brown2,o1|d1);
_mlparnm = {''Omega'',''BvBp-3''};
```

```
{o3,d3,cov3,llf3,aic3} = pewma(reg,brown3,o2|d2);
_mlparnm = {''Omega'',''BvBp-4''};
{o4,d4,cov4,llf4,aic4} = pewma(reg,brown4,o3|d3);
_mlparnm = {''Omega'',''BvBp-5''};
{o5,d5,cov5,llf5,aic5} = pewma(reg,brown5,o4|d4);
print;
print ''-----------------------------------------------------------------'';
print ''PEWMA estimates for economic regulation series with interventions'';
print ''-----------------------------------------------------------------'';
print ''Intervention period Omega I(t) Log-lik AIC'';
print '' 1953 '';;o0~d0~llf0~aic0;
print '' 1954 '';;o1~d1~llf1~aic1;
print '' 1955 '';;o2~d2~llf2~aic2;
print '' 1956 '';;o3~d3~llf3~aic3;
print '' 1957 '';;o4~d4~llf4~aic4;
print '' 1958 '';;o5~d5~llf5~aic5;
print ''-----------------------------------------------------------------'';
print;
print;

/* Now generate the fitted series for the best model */
{a,aa,aaa,b,bb,bbb,mu,r} = pewma_filter(reg,brown1,o1,d1);
regpma = trimr(aa,1,0)./trimr(bb,1,0);

/* Turn off output file */

output off;

/*-----------------------------------------------------------------*/
```

The output file (`court.out`) contains estimates of the all the model specifications, as well as a summary table of the log-likelihoods and Akaike Information Criterion values for the models. The summary tables are as follows:

```
PEWMA estimates for equality series with interventions

Intervention
period   Omega      I(t)      Log-lik   AIC
1953    0.568445 -0.231062  -168.070 338.140
1954    0.569538 -0.234397  -168.066 338.133
1955    0.569771 -0.303042  -168.002 338.003
1956    0.564397  0.302657  -167.947 337.894
1957    0.566138  0.0570067 -168.148 338.296
1958    0.572914 -0.204467  -168.072 338.144
1959    0.586174 -0.555759  -167.577 337.155
1960    0.594278 -1.05327   -166.158 334.316
1961    0.567887 -0.113186  -168.108 338.216
1962    0.558844 -0.281366  -167.825 337.649
1963    0.628433  0.902446  -162.141 326.282
1964    0.575436  0.160108  -168.013 338.025

PEWMA estimates for economic regulation series with interventions

Intervention
period   Omega      I(t)      Log-lik   AIC
1953    0.580625 -0.0330256 -205.928 413.855
1954    0.607772 -0.716524  -201.547 405.094
1955    0.577512  0.0596929 -205.894 413.788
1956    0.579907  0.0954543 -205.812 413.624
1957    0.581630  0.0895828 -205.823 413.645
1958    0.584763  0.175692  -205.465 412.930
```

In our earlier analyses, we used this output to select our model specification for prediction. Using the AIC criteria, we chose the model with a 10 period lead for the equality series, and a one period lead for the economic regulation series. It therefore appears that the change in the Supreme Court case agenda for civil rights was delayed, while the change in the regulation case agenda was rather proximate to the 1953 Supreme Court term.

6 Technical Descriptions and Syntax of the Procedures

This section describes the procedures contained in the zip archive. This section, as well as the code for each file serves as the main technical documentation for each of the procedures. The procedures are presented alphabetically. For each procedure, the following are noted:

- **Purpose:** what the procedure does
- **Format:** the syntax of the procedure
- **Inputs:** the valid input arguments for the procedure
- **Output:** the output from the procedure
- **Remarks:** any other information or technical details of the procedure. This includes any dependencies on other procedures in this library of programs.

6.1 AICLLF: Computing log-likelihood and AIC values

Purpose: Computes the log-likelihood and AIC values for any given likelihood function and set of data.

Format: `{aic,llf} = aicllf(data,param,&fct)`

Inputs:
`data` = TxK matrix of data for log-likelihood function
`param` = Kx1 matrix of parameters
`&fct` = likelihood function to be evaluated

Output:
AIC = Akaike's Information Criterion value
LLF = Log-likelihood function evaluated the parameters in param.

Remarks: This procedure is used to compute the log-likelihood and AIC values for the maximum likelihood estimation routines. This is a general procedure that can be used with any MLE estimator.

6.2 BETARNG: Beta random number generator

Purpose: Pseudo-random number generator for beta distributed random variables

Format: `x = betarng(a,b)`

Inputs:
a = shape parameter for beta distribution
b = scale parameter for beta distribution
n = number of beta random variates to generate

Output:
x = Nx1 vector of beta(a,b) distributed random variables

Remarks: This procedure is used to generate the stochastic process for pseudo PEWMA data in PEWMADGP.

6.3 CUSUM: Computes the CUSUM for a series

Purpose: Computes and graphs the CUSUM for a series. It assumes that the series is already a vector in memory

Format: s = CUSUM(resids,gr,a)

Inputs:
resids = TxK vector of residuals
gr = graphing code; 0 = no graph, 1 = graph

a = significance level for graph; 0.05 or 0.1

Output:
s = TxK vector of standardized CUSUM values
A graph, if gr=1

Remarks: The critical values for the CUSUM should be viewed as a rough estimate of the confidence interval, not as an exact interval. This code is programmed based on the description of the CUSUM in Harvey (1981).

6.4 CUSUMSQ: Computes the CUSUM of squares for a series

Purpose: Computes the CUSUM of squares for a series. It assumes that the series is already a vector in memory

Format: s = CUSUMSQ(resids)

Inputs:resids = TxK vector of residuals

Output: s = TxK vector of standardized CUSUMSQ values

Remarks: The confidence intervals for the CUSUMSQ test depend on the degrees of freedom and have a non-standard distribution. See Harvey (1981: Table C, pp.336-367) for a table of critical values.

6.5 EWMA: EWMA forecast function

Purpose: Compute an EWMA of the data.

Format: z = EWMA(x,omega)

Inputs:
x = Nx1 vector
omega = scalar, the weight (0<w<=1)

Output: z = scalar, exponentially weighted moving average of x

Remarks: This procedure computes the exact EWMA weight. It does not approximate the forecast function, so it is a bit slow. The procedure computes

$$z_t = \frac{\sum_{j=0}^{t-1} \omega^j x_{t-j}}{\sum_{j=0}^{t-1} \omega^j}$$

In small samples, it is exact, which makes it preferable to the large sample numerical approximation:

$$z_t = \omega z_{t-1|t-2} + (1-\omega)\, x_t$$

6.6 EWMAEST: EWMA estimator

Purpose: Computes the optimal EWMA by minimizing the sum of squares.

Format: w = ewmaest(x)

Input: X = Tx1 vector to be smoothed

Output: w = optimal weighting

Remarks: This procedure minimizes the sum of forecast residuals for the various values of w. It uses a simple grid search to find the optimal weight. The grid is from 0.01 to 1. If w = 1, then the optimal weight is the sample mean. The value of w is exact, since the EWMA's are computed without any approximations. As a consequence, this procedure will be very slow for large samples (i.e. T>100). The smoothed vector can be found by using EWMAV(x,w).

6.7 EWMAV: Smoother for Exponential Weighted Moving Averages

Purpose: Generating a vector exponentially weighted moving average

Format: z = EWMAV(x,omega);

Input :
x = Nx1 vector
omega = scalar, the weight (0<w<=1)

Output: z = Nx1 vector of exponentially weighted moving average

Remarks: EWMA's can be estimated using EWMAEST. If omega==1, then the procedure will return the sample mean. See also: EWMA, EWMAEST

6.8 HILLU: Estimate an ARIMA(0,0,1) model using a grid search

Purpose: Estimates an ARIMA(0,0,1) model via the Hildreth-Lu procedure

Format: {b,rho,cov} = hillu(y,x)

Inputs:
Y = Tx1 vector dependent variable
X = TxK matrix of regressors

Output:
b = estimated coefficients for mean
rho = estimated AR(1) coefficient
cov = covariance matrix computed from b and rho. The last row/column of cov are the covariances for rho.

Remarks: This code implements the Hildreth-Lu estimator for a regression model with AR(1) errors. The description of this procedure can be found in Davidson and MacKinnon (1993).

6.9 LNB: Log-likelihood function for a negative binomial regression model

Purpose: Computes the log-likelihood function for a negative binomial regression

Format: lnb(b,d)

Inputs:
b = vector of parameters. First element is the dispersion parameter
d = TxK data matrix, with the y variable in the first column.

Output: Log-likelihood vector for negative binomial regression model.

Remarks: Based on King (1989).

6.10 LNGM: log gamma function

Purpose: Computes the natural logarithm of the gamma function

Format: y = lngm(x)

Inputs: Nx1 vector, x>0

Output: y, Nx1 vector of log gamma values

Remarks: Computes log gamma function based on the algorithm in Press et al, 1986.

6.11 LPARP: log-likelihood of the PAR(p) model

Purpose: Computes the log-likelihood of the PAR(p) model

Format: lparp(p,yin)

Inputs: p = (P+K)x1 vector of regression and AR coefficients

Output: PAR(p) log-likelihood values

Remarks:

6.12 LPG: log-likelihood of the PEWMA model

Purpose: Computes the log-likelihood of the PEWMA model

Format: lpg(p,yin)

Inputs: (1+K)x1 = vector of coefficients. The first element is ω
yin = data matrix

Output: PEWMA log-likelihood values

Remarks:

6.13 LPSN: log-likelihood of the Poisson model

Purpose: Computes the log-likelihood value for the Poisson regression model

Format: lpsn(p,yin)

Inputs: p = Kx1 vector of parameters
yin = Tx(K+1) data matrix

Output: Poisson log-likelihood values

Remarks:

6.14 PARP: estimate a PAR(p) by maximum likelihood

Purpose: Purpose: Estimates a time series model for count data. The model is based on an extended Kalman filter where the state variable follows an AR(p).

Format: {rho,d,cov,llf,aic} = parp(y,x,p,sv)

Inputs:
y = Tx1 vector of counts
x = TxK matrix of covariates
p = order of autoregression
sv = starting value code or starting values

If the user provides the starting values, they must be in a ((K+P)x1) vector
in the form,
`b1|b2|...|bk|r1|r2|...|rp`,
where
`bj` is the j'th regression coefficient
`rk` is the k'th AR coefficient

The starting value codes are
sv = "P"; Lagged Poisson regression

sv = "OLS"; OLS on the natural logarithm of y
sv = vector; a vector of starting values

Output:
rho = Px1 vector of AR coefficients
d = Kx1 matrix of regression parameters
cov = (K+P)x(K+P) covariance matrix
llf = final log-likelihood value
aic = Akaike Information Criterion value

Remarks:

6.15 PARPDGP: generating PAR(p) simulated data

Purpose: Generates Poisson distributed random variables that have a gamma distributed AR(p) prior. Data follows a state-space model with:

$$\begin{aligned} y_t &\sim Poisson\left(\mu_t\right) \\ and\ \mu_t &= \sum_{i=1}^{p} \rho_i y_{t-1} + \left(1 - \sum_{i=1}^{p} \rho_i m\right) \\ and\ m &= \exp\left(X_t \delta\right) \end{aligned}$$

Format: {y} = parpdgp(n,x,r,d);

Inputs:
n = length of sample path
x = vector of covariates (nxk–with a constant)
r = Px1 weighting parameter for AR(p)
d = kx1 vector of parameters for link function

Output: y = PAR(p) distributed vector

Remarks:

6.16 PARPFILT: procedure for filtering PAR(p) model

Purpose: Computes filtered estimates for PAR(p) model.

Format: {m,s} = parpfilt(y,x,r,d)

Inputs:
y = Tx1 vector of counts
x = TxK matrix of covariates
p = AR(p) coefficients
d = Kx1 matrix of parameter for covariates

Output:

m = vector of conditional mean (t|t-1) in period t
s = vector of conditional variance (t|t-1) in period t

Remarks:

6.17 PEWMA: estimate a PEWMA by maximum likelihood

Purpose: Estimates a local level time series model for count data. The model is based on an extended Kalman filter where the state variable follows an EWMA.

Format: {omega,d,cov,llf,aic} = pewma(y,x,sv)

Inputs:
y = Tx1 vector of counts
x = TxK matrix of covariates (no constant!)
sv = starting value code or starting values

If the user provides the starting values, they must be in a ((K+1)x1) vector in the form, `omega|b2|...|bk`,

where
`omega` is EWMA parameter
`bj` is the j'th regression coefficient

Starting value codes:
sv = "P"; Poisson regression (default); initial value of w = 0.5;
sv = "OLS"; OLS on the natural logarithm of y; inital value of w = 0.5;
sv = "EWMA"; Gaussian EWMA; inital value of w estimated by EWMA;
sv = vector; a vector of starting values.

Output:
omega = scalar estimate of hyperparameter for EWMA
d = Kx1 matrix of regression parameters
cov = (K+1)x(K+1) covariance matrix
llf = final log-likelihood value
aic = Akaike Information Criterion value

Remarks:

6.18 PEWMADGP: generating PEWMA simulated data

Purpose: Generate data according to a specified PEWMA model.

Format: {y,mu} = pewmadgp(n,a0,b0,x,w,d)

Inputs:
n = length of sample path
a0 = prior for a

b0 = prior for b
x = vector of covariates (nxk–without a constant)
w = weighting parameter for EWMA of mu
d = kx1 vector of parameters for link function

Output:
y = PEWMA distributed vector
mu = conditional mean used to generate each count

Remarks:

6.19 PEWMA_FILTER: Filter for the PEWMA model

Purpose: Filters event count data according to a PEWMA model.

Format: {a,aa,aaa,b,bb,bbb,mu,r} = pewma_filter(y,x,w,d)

Inputs:
y = Tx1 vector of count series to be filtered
x = TxK matrix of covariates
w = EWMA parameter
d = Kx1 vector of covariate parameters

Output:
a = vector of a(t-1) parameters
aa = vector of a(t|t-1) parameters
aaa = vector of a(t) parameters
b = vector of b(t-1) parameters
bb = vector of b(t|t-1) parameters
bbb = vector of b(t) parameters
mu = level component of model
r = growth rate parameter

Remarks: This procedure can be used to compute forecasts from the PEWMA model. Since the output parameters are the parameters for the forecast functions, predictions and prediction errors can be computed based on the formulas in Brandt and Williams (1998a).

6.20 POISSON: Poisson regression estimator

Purpose: Estimates a Poisson regression model

Format: {d,cov,llf,aic} = poisson(y,x,sv)

Inputs:
y = Tx1 vector of counts
x = TxK matrix of covariates

sv = scalar, starting value code

Starting value codes:

sv = 0; OLS on the natural logarithm of y;
sv = 1; Zeros;

Output:
d = Kx1 matrix of regression parameters
cov = (K)x(K) covariance matrix
llf = final log-likelihood value
aic = Akaike Information Criterion value
Remarks:

6.21 PSI: compute Ψ, the first derivative of log gamma function

Purpose: Computes the first derivative of the log-gamma function.

Format: y=psi(x)

Inputs: x = vector of positive numbers

Output: y = vector of first derivative of the log-gamma of x

Remarks: Based on an approximation in Abramowitz and Stegun (1972)

6.22 RNDP: Poisson random number generator (Baird)

Purpose: Returns matrix of pseudo random numbers from Poisson distributions with means given in input matrix.

Format: y = rndp(r,c,m)

Inputs:
R - scalar - number of rows in returned matrix
C - scalar - number of columns in returned matrix
M - ExE matrix of means for Poisson distribution (Conformable with RxC matrix)

Output:
Y - RxC matrix containing samples from a poisson distribution with the mean given in the corresponding element of the input matrix.

Remarks: The maximum size of input matrix for which this procedure is guaranteed to work for is maxvec()./6 elements (1620 for Gauss 2). Larger input matrices can be used provided the mean values are not all in the range break1 < mean < break2. The seed is taken from the system clock at startup but may be set using the RNDSEED command (see GAUSS command reference).

Example: y = rndp(10,2,2~4) ; this gives a 10x2 matrix with column : 1 being 10 samples from a poisson dist. with mean 2; 2 being 10 samples from a poisson dist. with mean 4

This code is by Dr. David Baird. It is significantly faster than other random Poisson generators. The modal and normal approximations used in the code are similar to those described by Press et al. (1986).

7 Known bugs and problems

There is minimal error checking in these procedures. In part, this is because we have not found every possible bug! However, the procedures do check to be sure that the vectors of counts are really counts (i.e. they are greater than zero). Also, the procedures check whether the first observation is non-zero. The distribution of the counts is undefined when the first count is zero for the PEWMA and PAR(p). If your first few counts are zero, you will have to truncate the data in order to estimate the model(s).

The other major issue is the optimization routines.. These procedures are implemented for unconstrained optimization routine. A case can be made that since the admissible values of the hyperparameter, ω for the PEWMA are bounded, that a constrained procedure should be use. In practice, however, we have not found this to be necessary. When the true model is the Poisson regression and not the PEWMA or PAR(p), estimating a time series count model yields dynamic parameters (either the PEWMA parameter ω, or the PAR(p) coefficients) that are insignificant

8 References

- Brandt, Patrick, John T. Williams and Benjamin Fordham. 1998. "Modeling Time Series Count Data: A State-Space Approach to Event Counts." presented at the 1998 Society for Political Methodology Summer Meeting, San Diego, California, July 22-26, 1998.

- Brandt, Patrick T. and John T. Williams. 1998a. "Dynamic Modeling for Persistent Time Series of Event Counts."

- Brandt, Patrick T. and John T. Williams. 1998b. "A Linear Poisson Autoregressive Model: the Poisson AR(p) Model"

- Harvey, A.C. 1991. Forecasting, structural time series models and the Kalman filter. Cambridge: Cambridge University Press.

- Mooney, Christopher Z. 1997 Monte Carlo Simulation. Beverly Hills: Sage.

- Press, et.al. 1986. Numerical Recipies in Fortran: The Art of Scientific Computing. Cambridge: Cambridge University Press.