nag_glm_poisson (g02gcc)

1. Purpose

nag_glm_poisson (g02gcc) fits a generalized linear model with Poisson errors.

2. Specification

```
#include <nag.h>
#include <nagg02.h>

void nag_glm_poisson(Nag_Link link, Nag_IncludeMean mean, Integer n,
            double x[], Integer tdx, Integer m, Integer sx[],
            Integer ip, double y[], double wt[], double offset[],
            double ex_power, double *dev, double *df, double b[],
            Integer *rank, double se[], double cov[], double v[],
            Integer tdv, double tol, Integer max_iter, Integer print_iter,
            char *outfile, double eps, NagError *fail)
```

3. Description

A generalized linear model with Poisson errors consists of the following elements:

(a) a set of n observations, y_i, from a Poisson distribution:

$$\frac{\mu^y e^{-\mu}}{y!}$$

(b) X, a set of p independent variables for each observation, $x_1, x_2, ..., x_p$.

(c) a linear model:

$$\eta = \sum \beta_j x_j.$$

(d) a link between the linear predictor, η, and the mean of the distribution, μ, $\eta = g(\mu)$. The possible link functions are:

(i) exponent link: $\eta = \mu^a$, for a constant a,

(ii) identity link: $\eta = \mu$,

(iii) log link: $\eta = \log \mu$,

(iv) square root link: $\eta = \sqrt{\mu}$,

(v) reciprocal link: $\eta = \dfrac{1}{\mu}$.

(e) a measure of fit, the deviance:

$$\sum_{i=1}^{n} \mathrm{dev}(y_i, \hat{\mu}_i) = \sum_{i=1}^{n} 2 \left\{ y_i \log \left(\frac{y_i}{\hat{\mu}_i} \right) - (y_i - \hat{\mu}_i) \right\}$$

The linear parameters are estimated by iterative weighted least-squares. An adjusted dependent variable, z, is formed:

$$z = \eta + (y - \mu) \frac{d\eta}{d\mu}$$

and a working weight, w,

$$w = \left(\tau \frac{d\eta}{d\mu} \right)^2, \quad \text{where } \tau = \sqrt{\mu}.$$

At each iteration an approximation to the estimate of β, $\hat{\beta}$ is found by the weighted least-squares regression of z on X with weights w.

nag_glm_poisson finds a QR decomposition of $w^{\frac{1}{2}}X$, i.e.,

$w^{\frac{1}{2}}X = QR$ where R is a p by p triangular matrix and Q is an n by p column orthogonal matrix.

If R is of full rank then $\hat{\beta}$ is the solution to:

$$R\hat{\beta} = Q^T w^{\frac{1}{2}} z$$

If R is not of full rank a solution is obtained by means of a singular value decomposition (SVD) of R.

$$R = Q_* \begin{pmatrix} D & 0 \\ 0 & 0 \end{pmatrix} P^T.$$

where D is a k by k diagonal matrix with non-zero diagonal elements, k being the rank of R and $w^{\frac{1}{2}}X$.

This gives the solution

$$\hat{\beta} = P_1 D^{-1} \begin{pmatrix} Q_* & 0 \\ 0 & I \end{pmatrix} Q^T w^{\frac{1}{2}} z$$

P_1 being the first k columns of P, i.e., $P = (P_1 P_0)$.

The iterations are continued until there is only a small change in the deviance.

The initial values for the algorithm are obtained by taking

$$\hat{\eta} = g(y)$$

The fit of the model can be assessed by examining and testing the deviance, in particular, by comparing the difference in deviance between nested models, i.e., when one model is a sub-model of the other. The difference in deviance between two nested models has, asymptotically, a χ^2 distribution with degress of freedom given by the difference in the degrees of freedom associated with the two deviances.

The parameters estimates, $\hat{\beta}$, are asymptotically Normally distributed with variance-covariance matrix:

$C = R^{-1}R^{-1^T}$ in the full rank case, otherwise
$C = P_1 D^{-2} P_1^T$

The residuals and influence statistics can also be examined.

The estimated linear predictor $\hat{\eta} = X\hat{\beta}$, can be written as $Hw^{\frac{1}{2}}z$ for an n by n matrix H. The ith diagonal elements of H, h_i, give a measure of the influence of the ith values of the independent variables on the fitted regression model. These are known as leverages.

The fitted values are given by $\hat{\mu} = g^{-1}(\hat{\eta})$.

nag_glm_poisson also computes the deviance residuals, r:

$$r_i = \text{sign}(y_i - \hat{\mu}_i)\sqrt{\text{dev}(y_i, \hat{\mu}_i)}.$$

An option allows prior weights to be used with the model.

In many linear regression models the first term is taken as a mean term or an intercept, i.e., $x_{i,1} = 1$, for $i = 1, 2, ..., n$. This is provided as an option.

Often only some of the possible independent variables are included in a model; the facility to select variables to be included in the model is provided.

If part of the linear predictor can be represented by a variable with a known coefficient then this can be included in the model by using an offset, o:

$$\eta = o + \sum \beta_j x_j.$$

If the model is not of full rank the solution given will be only one of the possible solutions. Other estimates be may be obtained by applying constraints to the parameters. These solutions can be obtained by using nag_glm_tran_model (g02gkc) after using nag_glm_poisson. Only certain linear combinations of the parameters will have unique estimates, these are known as estimable functions, these can be estimated and tested using nag_glm_est_func (g02gnc).

Details of the SVD, are made available, in the form of the matrix P^*:

$$P^* = \begin{pmatrix} D^{-1}P_1^T \\ P_0^T \end{pmatrix}.$$

The generalized linear model with Poisson errors can be used to model contingency table data, see Cook and Weisberg(1982) and McCullagh and Nelder(1983).

4. Parameters

link

Input: indicates which link function is to be used.
If **link** = **Nag_Expo**, then an exponent link is used.
If **link** = **Nag_Iden**, then an identity link is used.
If **link** = **Nag_Log**, then a log link is used.
If **link** = **Nag_Sqrt**, then a square root link is used.
If **link** = **Nag_Reci**, then a reciprocal link is used.
Constraint: **link** = **Nag_Expo**, **Nag_Iden**, **Nag_Log**, **Nag_Sqrt** or **Nag_Reci**.

mean

Input: indicates if a mean term is to be included.
If **mean** = **Nag_MeanInclude**, a mean term, (intercept), will be included in the model.
If **mean** = **Nag_MeanZero**, the model will pass through the origin, zero point.
Constraint: **mean** = **Nag_MeanInclude** or **Nag_MeanZero**.

n

Input: the number of observations, n.
Constraint: **n** ≥ 2.

x[n][tdx]

Input: **x**$[i-1][j-1]$ must contain the ith observation for the jth independent variable, for $i = 1, 2, \ldots, n$; $j = 1, 2, \ldots,$**m**.

tdx

Input: the second dimension of the array **x** as declared in the function from which nag_glm_poisson is called.
Constraint: **tdx** $\geq$ **m**.

m

Input: the total number of independent variables.
Constraint: **m** ≥ 1.

sx[m]

Input: indicates which independent variables are to be included in the model.
If **sx**$[j-1] > 0$, then the variable contained in the jth column of **x** is included in the regression model.
Constraints: **sx**$[j-1] \geq 0$, for $j = 1, 2, \ldots,$**m**.
if **mean** = **Nag_MeanInclude**, then exactly **ip** -1 values of **sx** must be > 0.
if **mean** = **Nag_MeanZero**, then exactly **ip** values of **sx** must be > 0.

ip

Input: the number p of independent variables in the model, including the mean or intercept if present.
Constraint: **ip** > 0.

y[n]

Input: observations on the dependent variable, y_i, for $i = 1, \ldots, n$.
Constraint: **y**$[i-1] \geq 0$, for $i = 1, 2, \ldots, n$.

wt[n]

Input: if weighted estimates are required then **wt** must contain the weights to be used with the model, ω_i. Otherwise **wt** must be supplied as the null pointer, (double *)0.
If $\mathbf{wt}[i-1] = 0.0$, then the ith observation is not included in the model, in which case the effective number of observations is the number of observations with positive weights.
If **wt** = null pointer, then the effective number of observations is n.
Constraint: **wt** = null pointer or $\mathbf{wt}[i-1] \geq 0.0$, for $i = 1, 2, \ldots, n$.

offset[n]

Input: if an offset is required then **offset** must contain the values of the offset o. Otherwise **offset** must be supplied as the null pointer, (double *)0.

ex_power

Input: if **link** = **Nag_Expo** then **ex_power** must contain the power a of the exponential.
If **link** $\neq$ **Nag_Expo**, **ex_power** is not referenced.
Constraint: If **link** = **Nag_Expo**, **ex_power** $\neq 0.0$.

dev

Output: the deviance for the fitted model.

df

Output: the degrees of freedom associated with the deviance for the fitted model.

b[ip]

Output: the estimates of the parameters of the generalized linear model, $\hat{\beta}$.
If **mean** = **Nag_MeanInclude**, then $\mathbf{b}[0]$ will contain the estimate of the mean parameter and $\mathbf{b}[i]$ will contain the coefficient of the variable contained in column j of **x**, where $\mathbf{sx}[j-1]$ is the ith positive value in the array **sx**.
If **mean** = **Nag_MeanZero**, then $\mathbf{b}[i-1]$ will contain the coefficient of the variable contained in column j of **x**, where $\mathbf{sx}[j-1]$ is the ith positive value in the array **sx**.

rank

Output: the rank of the independent variables.
If the model is of full rank, then **rank** = **ip**.
If the model is not of full rank, then **rank** is an estimate of the rank of the independent variables. **rank** is calculated as the number of singular values greater than **eps** $\times$ (largest singular value). It is possible for the SVD to be carried out but **rank** to be returned as **ip**.

se[ip]

Output: the standard errors of the linear parameters.
$\mathbf{se}[i-1]$ contains the standard error of the parameter estimate in $\mathbf{b}[i-1]$, for $i = 1, 2, \ldots,$ **ip**.

cov[ip∗(ip+1)/2]

Output: the $\mathbf{ip} \times (\mathbf{ip}+1)/2$ elements of **cov** contain the upper triangular part of the variance-covariance matrix of the **ip** parameter estimates given in **b**. They are stored packed by column, i.e., the covariance between the parameter estimate given in $\mathbf{b}[i]$ and the parameter estimate given in $\mathbf{b}[j]$, $j \geq i$, is stored in $\mathbf{cov}[j(j+1)/2 + i]$, for $i = 0, 1, \ldots, \mathbf{ip}-1$ and $j = i, i+1, \ldots, \mathbf{ip}-1$.

v[n][tdv]

Output: auxiliary information on the fitted model.
$\mathbf{v}[i-1][0]$, contains the linear predictor value, η_i, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][1]$, contains the fitted value, $\hat{\mu}_i$, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][2]$, contains the variance standardization, τ_i, for $i = 1, 2, ..., n$.
$\mathbf{v}[i-1][3]$, contains the working weight, w_i, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][4]$, contains the deviance residual, r_i, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][5]$, contains the leverage, h_i, for $i = 1, 2, \ldots, n$.
$\mathbf{v}[i-1][j-1]$, for $j = 7, \ldots,$**ip**+6, contains the results of the QR decomposition or the singular value decomposition.

If the model is not of full rank, i.e., **rank** < **ip**, then the first **ip** rows of columns 7 to **ip**+6 contain the P^* matrix.

tdv

Input: the second dimension of the array **v** as declared in the function from which nag_glm_poisson is called.
Constraint: **tdv** $\geq$ **ip**+6.

tol

Input: indicates the accuracy required for the fit of the model.
The iterative weighted least-squares procedure is deemed to have converged if the absolute change in deviance between interactions is less than **tol** $\times$ (1.0+Current Deviance). This is approximately an absolute precision if the deviance is small and a relative precision if the deviance is large.
If $0.0 \leq$ **tol** $<$ ***machine precision***, then the function will use $10\times$ ***machine precision***.
Constraint: **tol** ≥ 0.0.

max_iter

Input: the maximum number of iterations for the iterative weighted least-squares.
If **max_iter** $= 0$, then a default value of 10 is used.
Constraint: **max_iter** ≥ 0.

print_iter

Input: indicates if the printing of information on the iterations is required and the rate at which printing is produced. The following values are available.
If **print_iter** ≤ 0, then there is no printing.
If **print_iter** > 0, then the following items are printed every **print_iter** iterations:

(i) the deviance,
(ii) the current estimates, and
(iii) if the weighted least-squares equations are singular then this is indicated.

outfile

Input: a null terminated character string giving the name of the file to which results should be printed. If **outfile = NULL** or an empty string then the `stdout` stream is used. Note that the file will be opened in the append mode.

eps

Input: the value of **eps** is used to decide if the independent variables are of full rank and, if not, what the rank of the independent variables is. The smaller the value of **eps** the stricter the criterion for selecting the singular value decomposition.
If $0.0 \leq$ **eps** $<$ ***machine precision***, then the function will use ***machine precision*** instead.
Constraint: **eps** ≥ 0.0.

fail

The NAG error parameter, see the Essential Introduction to the NAG C Library.

For this function the values of output parameters may be useful even if **fail.code** $\neq$ **NE_NOERROR** on exit. Users are therefore advised to supply the **fail** parameter and set **fail.print** = `TRUE`.

5. Error Indications and Warnings

NE_BAD_PARAM

On entry, parameter **link** had an illegal value.
On entry, parameter **mean** had an illegal value.

NE_INT_ARG_LT

On entry, **n** must not be less than 2: **n** = ⟨*value*⟩.
On entry, **m** must not be less than 1: **m** = ⟨*value*⟩.
On entry, **ip** must not be less than 1: **ip** = ⟨*value*⟩.
On entry, **sx**[⟨*value*⟩] must not be less than 0: **sx**[⟨*value*⟩] = ⟨*value*⟩.
On entry, **max_iter** must not be less than 0: **max_iter** = ⟨*value*⟩.

NE_2_INT_ARG_LT

On entry, **tdx** = ⟨*value*⟩ while **m** = ⟨*value*⟩. These parameters must satisfy **tdx** $\geq$ **m**.
On entry, **tdv** = ⟨*value*⟩ while **ip** = ⟨*value*⟩. These parameters must satisfy **tdv** $\geq$ **ip**+6.

NE_REAL_ARG_LT

On entry, **tol** must not be less than 0.0: **tol** = ⟨*value*⟩.
On entry, **eps** must not be less than 0.0: **eps** = ⟨*value*⟩.
On entry, **wt**[⟨*value*⟩] must not be less than 0.0: **wt**[⟨*value*⟩] = ⟨*value*⟩.
On entry, **y**[⟨*value*⟩] must not be less than 0.0: **y**[⟨*value*⟩] = ⟨*value*⟩.

NE_REAL_ENUM_ARG_CONS

On entry **ex_power** = 0.0, **link** = **Nag_Expo**. These parameters must satisfy **link** == **Nag_Expo** && **ex_power** $\neq$ 0.0.

NE_ALLOC_FAIL

Memory allocation failed.

NE_IP_INCOMP_SX

Parameter **ip** is incompatible with **mean** and **sx**.

NE_IP_GT_OBSERV

Parameter **ip** is greater than the effective number of observations.

NE_SVD_NOT_CONV

The singular value decomposition has failed to converge.

NE_VALUE_AT_BOUNDARY_C

A fitted value is at a boundary, i.e., $\hat{\mu} = 0.0$. This may occur if there are y values of 0.0 and the model is too complex for the data. The model should be reformulated with, perhaps, some observations dropped.

NE_LSQ_ITER_NOT_CONV

The iterative weighted least-squares has failed to converge in **max_iter** = ⟨*value*⟩ iterations. The value of **max_iter** could be increased but it may be advantageous to examine the convergence using the **print_iter** option. This may indicate that the convergence is slow because the solution is at a boundary in which case it may be better to reformulate the model.

NE_RANK_CHANGED

The rank of the model has changed during the weighted least-squares iterations. The estimate for β returned may be reasonable, but the user should check how the deviance has changed during iterations.

NE_ZERO_DOF_ERROR

The degrees of freedom for error are 0. A saturated model has been fitted.

NE_NOT_APPEND_FILE

Cannot open file ⟨*string*⟩ for appending.

NE_NOT_CLOSE_FILE

Cannot close file ⟨*string*⟩.

6. Further Comments

6.1. Accuracy

The accuracy is determined by **tol** as described in Section 4. As the adjusted deviance is a function of $\log\mu$ the accuracy of the $\hat{\beta}$'s will be a function of **tol**. **tol** should therefore be set to a smaller value than the accuracy required for $\hat{\beta}$.

6.2. References

Cook R D and Weisberg S (1982) *Residuals and Influence in Regression.* Chapman and Hall.
McCullagh P and Nelder J A (1983) *Generalized Linear Models* Chapman and Hall.
Plackett R L (1974) *The Analysis of Categorical Data.* Griffin.

7. See Also

nag_glm_normal (g02gac)
nag_glm_binomial (g02gbc)

nag_glm_gamma (g02gdc)
nag_glm_tran_model (g02gkc)
nag_glm_est_func (g02gnc)

8. Example

A 3 by 5 contingency table given by Plackett (1974) is analysed by fitting terms for rows and columns. The table is:

141	67	114	79	39
131	66	143	72	35
36	14	38	28	16

8.1. Program Text

```
/* nag_glm_poisson(g02gcc) Example Program.
 *
 * Copyright 1996 Numerical Algorithms Group.
 *
 * Mark 4, 1996.
 *
 */

#include <nag.h>
#include <stdio.h>
#include <nag_stdlib.h>
#include <ctype.h>
#include <nagg02.h>

#ifdef NAG_PROTO
static void set_enum(char linkc, Nag_Link *link, char meanc,
                     Nag_IncludeMean *mean);
#else
static void set_enum();
#endif

#define NMAX 15
#define MMAX  9
#define TDX MMAX
#define TDV MMAX+6

main()
{
  char  linkc, meanc, weightc;
  Nag_Link  link;
  Nag_IncludeMean  mean;
  Integer i, j, m, n, ip;
  double  ex_power, scale;
  Integer sx[MMAX];
  double  b[MMAX], v[NMAX][TDV], wt[NMAX], x[NMAX][MMAX],
  y[NMAX];
  double  *wtptr, *offsetptr=(double *)0;
  Integer max_iter;
  Integer print_iter;
  double eps;
  double tol;
  double df;
  double dev;
  Integer rank;
  double se[MMAX], cov[MMAX*(MMAX+1)/2];
  static  NagError fail;

  Vprintf("g02gcc Example Program Results\n");
  /*  Skip heading in data file */
  Vscanf("%*[^\n]");
  Vscanf(" %c %c %c %ld %ld %ld", &linkc, &meanc, &weightc, &n,
         &m, &print_iter);
```

```
/* Check and set control parameters */
set_enum(linkc, &link, meanc, &mean);
if (n<=NMAX && m<MMAX)
  {
    if (toupper(weightc)=='W')
      {
        wtptr = wt;
        for (i=0; i<n; i++)
          {
            for (j=0; j<m; j++)
              Vscanf("%lf", &x[i][j]);
            Vscanf("%lf%lf", &y[i], &wt[i]);
          }
      }
    else
      {
        wtptr = (double *)0;
        for (i=0; i<n; i++)
          {
            for (j=0; j<m; j++)
              Vscanf("%lf", &x[i][j]);
            Vscanf("%lf", &y[i]);
          }
      }
    for (j=0; j<m; j++)
      Vscanf("%ld", &sx[j]);

    /* Calculate ip */
    ip = 0;
    for (j=0; j<m; j++)
      if (sx[j]>0) ip += 1;
    if (mean == Nag_MeanInclude)
      ip += 1;
    if (link == Nag_Expo)
      Vscanf("%lf", &ex_power);
    else
      ex_power = 0.0;

    /* Set other control parameters */
    max_iter = 10;
    tol = 5e-5;
    eps = 1e-6;

    g02gcc(link, mean, n, (double *)x, (Integer)TDX, m,sx, ip, y,
           wtptr, offsetptr, ex_power, &dev, &df, b, &rank, se, cov,
           (double *)v, (Integer)TDV, tol, max_iter, print_iter,
           "", eps, &fail);

    if (fail.code == NE_NOERROR || fail.code == NE_LSQ_ITER_NOT_CONV ||
        fail.code == NE_RANK_CHANGED || fail.code == NE_ZERO_DOF_ERROR)
      {
        Vprintf("\nDeviance = %12.4e\n", dev);
        Vprintf("Degrees of freedom = %3.1f\n\n", df);
        Vprintf("       Estimate     Standard error\n\n");
        for (i=0; i<ip; i++)
          Vprintf("%14.4f%14.4f\n", b[i], se[i]);
        Vprintf("\n");
        Vprintf("    y   fitted value  Residual   Leverage\n\n");
        for (i = 0; i < n; ++i)
          {
            Vprintf("%7.1f%10.2f%12.4f%10.3f\n", y[i], v[i][1], v[i][4],
                    v[i][5]);
          }
      }
    else
      {
        Vprintf("%s\n",fail.message);
        exit(EXIT_FAILURE);
      }
  }
```

```
  else
    {
      Vfprintf(stderr, "One or both of m and n are out of range:\
 m = %-3ld while  n = %-3ld\n", m, n);
      exit(EXIT_FAILURE);
    }
  exit(EXIT_SUCCESS);
}

#ifdef NAG_PROTO
static void set_enum(char linkc, Nag_Link *link, char meanc,
                     Nag_IncludeMean *mean)
#else
     static void set_enum(linkc, link, meanc, mean)
     char linkc;
     Nag_Link *link;
     char meanc;
     Nag_IncludeMean *mean;
#endif
{
  if (toupper(linkc) == 'E' || toupper(linkc) == 'I' || toupper(linkc) == 'L'
      || toupper(linkc) == 'S' || toupper(linkc) == 'R')
    {
      switch (toupper(linkc))
        {
        case ('E'):
          *link = Nag_Expo;
          break;
        case ('I'):
          *link = Nag_Iden;
          break;
        case ('L'):
          *link = Nag_Log;
          break;
        case ('S'):
          *link = Nag_Sqrt;
          break;
        case ('R'):
          *link = Nag_Reci;
          break;
        default:
          ;
        }
    }
  else
    {
      Vfprintf(stderr, "The parameter link has an invalid value: link = %c\n",
               linkc);
      exit(EXIT_FAILURE);
    }
  if (toupper(meanc)=='M')
    *mean = Nag_MeanInclude;
  else if (toupper(meanc)=='Z')
    *mean = Nag_MeanZero;
  else
    {
      Vfprintf(stderr, "The parameter mean has an invalid value: mean = %c\n",
               meanc);
      exit(EXIT_FAILURE);
    }
  return;
}
```

8.2. Program Data

```
g02gcc Example Program Data
l   m   n 15 8 0
1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 141.
1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0  67.
1.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 114.
1.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0  79.
1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0  39.
0.0 1.0 0.0 1.0 0.0 0.0 0.0 0.0 131.
0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0  66.
0.0 1.0 0.0 0.0 0.0 1.0 0.0 0.0 143.
0.0 1.0 0.0 0.0 0.0 0.0 1.0 0.0  72.
0.0 1.0 0.0 0.0 0.0 0.0 0.0 1.0  35.
0.0 0.0 1.0 1.0 0.0 0.0 0.0 0.0  36.
0.0 0.0 1.0 0.0 1.0 0.0 0.0 0.0  14.
0.0 0.0 1.0 0.0 0.0 1.0 0.0 0.0  38.
0.0 0.0 1.0 0.0 0.0 0.0 1.0 0.0  28.
0.0 0.0 1.0 0.0 0.0 0.0 0.0 1.0  16.
1   1   1   1   1   1   1   1
```

8.3. Program Results

```
g02gcc Example Program Results

Deviance =   9.0379e+00
Degrees of freedom = 8.0

       Estimate     Standard error

        2.5977        0.0258
        1.2619        0.0438
        1.2777        0.0436
        0.0580        0.0668
        1.0307        0.0551
        0.2910        0.0732
        0.9876        0.0559
        0.4880        0.0675
       -0.1996        0.0904

     y   fitted value  Residual   Leverage

   141.0    132.99      0.6875     0.604
    67.0     63.47      0.4386     0.514
   114.0    127.38     -1.2072     0.596
    79.0     77.29      0.1936     0.532
    39.0     38.86      0.0222     0.482
   131.0    135.11     -0.3553     0.608
    66.0     64.48      0.1881     0.520
   143.0    129.41      1.1749     0.601
    72.0     78.52     -0.7465     0.537
    35.0     39.48     -0.7271     0.488
    36.0     39.90     -0.6276     0.393
    14.0     19.04     -1.2131     0.255
    38.0     38.21     -0.0346     0.382
    28.0     23.19      0.9675     0.282
    16.0     11.66      1.2028     0.206
```