NAG Library Routine Document

G05PVF

Note: before using this routine, please read the Users' Note for your implementation to check the interpretation of ***bold italicised*** terms and other implementation-dependent details.

1 Purpose

G05PVF generates training and validation datasets suitable for use in cross-validation or jack-knifing.

2 Specification

```
SUBROUTINE G05PVF (K, FOLD, N, M, SORDX, X, LDX, USEY, Y, USEW, W, NT,      &
                   STATE, IFAIL)
INTEGER            K, FOLD, N, M, SORDX, LDX, USEY, USEW, NT, STATE(*),     &
                   IFAIL
REAL (KIND=nag_wp) X(LDX,*), Y(*), W(*)
```

3 Description

Let X_o denote a matrix of n observations on m variables and y_o and w_o each denote a vector of length n. For example, X_o might represent a matrix of independent variables, y_o the dependent variable and w_o the associated weights in a weighted regression.

G05PVF generates a series of training datasets, denoted by the matrix, vector, vector triplet (X_t, y_t, w_t) of n_t observations, and validation datasets, denoted (X_v, y_v, w_v) with n_v observations. These training and validation datasets are generated as follows.

Each of the original n observations is randomly assigned to one of K equally sized groups or folds. For the kth sample the validation dataset consists of those observations in group k and the training dataset consists of all those observations not in group k. Therefore at most K samples can be generated.

If n is not divisible by K then the observations are assigned to groups as evenly as possible, therefore any group will be at most one observation larger or smaller than any other group.

When using $K = n$ the resulting datasets are suitable for leave-one-out cross-validation, or the training dataset on its own for jack-knifing. When using $K \neq n$ the resulting datasets are suitable for K-fold cross-validation. Datasets suitable for reversed cross-validation can be obtained by switching the training and validation datasets, i.e., use the kth group as the training dataset and the rest of the data as the validation dataset.

One of the initialization routines G05KFF (for a repeatable sequence if computed sequentially) or G05KGF (for a non-repeatable sequence) must be called prior to the first call to G05PVF.

4 References

None.

5 Arguments

1: K – INTEGER *Input*

On entry: K, the number of folds.

Constraint: $2 \leq \text{K} \leq \text{N}$.

2: FOLD – INTEGER *Input*

On entry: the number of the fold to return as the validation dataset.

On the first call to G05PVF FOLD should be set to 1 and then incremented by one at each subsequent call until all K sets of training and validation datasets have been produced. See Section 9 for more details on how a different calling sequence can be used.

Constraint: $1 \leq \text{FOLD} \leq \text{K}$.

3: N – INTEGER *Input*

On entry: n, the number of observations.

Constraint: $\text{N} \geq 1$.

4: M – INTEGER *Input*

On entry: m, the number of variables.

Constraint: $\text{M} \geq 1$.

5: SORDX – INTEGER *Input*

On entry: determines how variables are stored in X.

Constraint: $\text{SORDX} = 1$ or 2.

6: X(LDX, $*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the second dimension of the array X must be at least M if $\text{SORDX} = 1$ and at least N if $\text{SORDX} = 2$.

The way the data is stored in X is defined by SORDX.

If $\text{SORDX} = 1$, $\text{X}(i, j)$ contains the ith observation for the jth variable, for $i = 1, 2, \ldots, \text{N}$ and $j = 1, 2, \ldots, \text{M}$.

If $\text{SORDX} = 2$, $\text{X}(j, i)$ contains the ith observation for the jth variable, for $i = 1, 2, \ldots, \text{N}$ and $j = 1, 2, \ldots, \text{M}$.

On entry: if $\text{FOLD} = 1$, X must hold X_o, the values of X for the original dataset, otherwise, X must not be changed since the last call to G05PVF.

On exit: values of X for the training and validation datasets, with X_t held in observations 1 to NT and X_v in observations $\text{NT} + 1$ to N.

7: LDX – INTEGER *Input*

On entry: the first dimension of the array X as declared in the (sub)program from which G05PVF is called.

Constraints:

if $\text{SORDX} = 2$, $\text{LDX} \geq \text{M}$;
otherwise $\text{LDX} \geq \text{N}$.

8: USEY – INTEGER *Input*

On entry: if $\text{USEY} = 1$, the original dataset includes y_o and y_o will be processed alongside X_o.

Constraint: $\text{USEY} = 0$ or 1.

9: Y($*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the dimension of the array Y must be at least N if $\text{USEY} = 1$.

If $\text{USEY} = 0$, Y is not referenced on entry and will not be modified on exit.

On entry: if $\text{FOLD} = 1$, Y must hold y_o, the values of y for the original dataset, otherwise Y must not be changed since the last call to G05PVF.

On exit: values of y for the training and validation datasets, with y_t held in elements 1 to NT and y_v in elements $\mathrm{NT}+1$ to N.

10: USEW – INTEGER *Input*

On entry: if $\mathrm{USEW}=1$, the original dataset includes w_o and w_o will be processed alongside X_o.

Constraint: $\mathrm{USEW}=0$ or 1.

11: W($*$) – REAL (KIND=nag_wp) array *Input/Output*

Note: the dimension of the array W must be at least N if $\mathrm{USEW}=1$.

If $\mathrm{USEW}=0$, W is not referenced on entry and will not be modified on exit.

On entry: if $\mathrm{FOLD}=1$, W must hold w_o, the values of w for the original dataset, otherwise W must not be changed since the last call to G05PVF.

On exit: values of w for the training and validation datasets, with w_t held in elements 1 to NT and w_v in elements $\mathrm{NT}+1$ to N.

12: NT – INTEGER *Output*

On exit: n_t, the number of observations in the training dataset.

13: STATE($*$) – INTEGER array *Communication Array*

Note: the actual argument supplied **must** be the array STATE supplied to the initialization routines G05KFF or G05KGF.

On entry: contains information on the selected base generator and its current state.

On exit: contains updated information on the state of the generator.

14: IFAIL – INTEGER *Input/Output*

On entry: IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this argument you should refer to Section 3.4 in How to Use the NAG Library and its Documentation for details.

For environments where it might be inappropriate to halt program execution when an error is detected, the value -1 or 1 is recommended. If the output of error messages is undesirable, then the value 1 is recommended. Otherwise, because for this routine the values of the output arguments may be useful even if $\mathrm{IFAIL}\neq 0$ on exit, the recommended value is -1. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

On exit: $\mathrm{IFAIL}=0$ unless the routine detects an error or a warning has been flagged (see Section 6).

6 Error Indicators and Warnings

If on entry $\mathrm{IFAIL}=0$ or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Note: G05PVF may return useful information for one or more of the following detected errors or warnings.

Errors or warnings detected by the routine:

$\mathrm{IFAIL}=11$

On entry, $\mathrm{K}=\langle\mathit{value}\rangle$ and $\mathrm{N}=\langle\mathit{value}\rangle$.
Constraint: $2\leq\mathrm{K}\leq\mathrm{N}$.

IFAIL $= 21$

On entry, FOLD $= \langle value \rangle$ and K $= \langle value \rangle$.
Constraint: $1 \leq$ FOLD $\leq$ K.

IFAIL $= 31$

On entry, N $= \langle value \rangle$.
Constraint: N ≥ 1.

IFAIL $= 41$

On entry, M $= \langle value \rangle$.
Constraint: M ≥ 1.

IFAIL $= 51$

On entry, SORDX $= \langle value \rangle$.
Constraint: SORDX $= 1$ or 2.

IFAIL $= 61$

More than 50% of the data did not move when the data was shuffled. $\langle value \rangle$ of the $\langle value \rangle$ observations stayed put.

IFAIL $= 71$

On entry, LDX $= \langle value \rangle$ and N $= \langle value \rangle$.
Constraint: if SORDX $= 1$, LDX $\geq$ N.

IFAIL $= 72$

On entry, LDX $= \langle value \rangle$ and M $= \langle value \rangle$.
Constraint: if SORDX $= 2$, LDX $\geq$ M.

IFAIL $= 81$

Constraint: USEY $= 0$ or 1.

IFAIL $= 101$

Constraint: USEW $= 0$ or 1.

IFAIL $= 131$

On entry, STATE vector has been corrupted or not initialized.

IFAIL $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

See Section 3.9 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -399$

Your licence key may have expired or may not have been installed correctly.

See Section 3.8 in How to Use the NAG Library and its Documentation for further information.

IFAIL $= -999$

Dynamic memory allocation failed.

See Section 3.7 in How to Use the NAG Library and its Documentation for further information.

7 Accuracy

Not applicable.

8 Parallelism and Performance

G05PVF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

G05PVF makes calls to BLAS and/or LAPACK routines, which may be threaded within the vendor library used by this implementation. Consult the documentation for the vendor library for further information.

Please consult the X06 Chapter Introduction for information on how to control and interrogate the OpenMP environment used within this routine. Please also consult the Users' Note for your implementation for any additional implementation-specific information.

9 Further Comments

G05PVF will be computationality more efficient if each observation in X is contiguous, that is SORDX $= 2$.

Because of the way G05PVF stores the data you should usually generate the K training and validation datasets in order, i.e., set FOLD $= 1$ on the first call and increment it by one at each subsequent call. However, there are times when a different calling sequence would be beneficial, for example, when performing different cross-validation analyses on different threads. This is possible, as long as the following is borne in mind:

> G05PVF must be called with FOLD $= 1$ first.
>
> Other than the first set, you can obtain the training and validation dataset in any order, but for a given X you can only obtain each once.

For example, if you have three threads, you would call G05PVF once with FOLD $= 1$. You would then copy the X returned onto each thread and generate the remaing K $- 1$ sets of data by splitting them between the threads. For example, the first thread runs with FOLD $= 2, \ldots, L_1$, the second with FOLD $= L_1 + 1, \ldots, L_2$ and the third with FOLD $= L_2 + 1, \ldots,$ K.

10 Example

This example uses G05PVF to facilitate K-fold cross-validation.

A set of simulated data is split into 5 training and validation datasets. G02GBF is used to fit a logistic regression model to each training dataset and then G02GPF is used to predict the response for the observations in the validation dataset.

The counts of true and false positives and negatives along with the sensitivity and specificity is then reported.

10.1 Program Text

```
    Program g05pvfe

!     G05PVF Example Program Text

!     Mark 26 Release. NAG Copyright 2016.

!     .. Use Statements ..
      Use nag_library, Only: g02gbf, g02gpf, g05kff, g05pvf, nag_wp
!     .. Implicit None Statement ..
      Implicit None
!     .. Parameters ..
      Integer, Parameter              :: lseed = 1, nin = 5, nout = 6
!     .. Local Scalars ..
      Real (Kind=nag_wp)              :: a, dev, eps, s, tol
```

```
      Integer                            :: fn, fold, fp, genid, i, idf, ifail,  &
                                            ip, iprint, irank, k, ldv, ldx,      &
                                            lstate, lwk, m, maxit, max_nv, n,    &
                                            nn, np, nt, nv, obs_val, pred_val,   &
                                            sordx, subid, tn, tp, uset, usey
      Logical                            :: vfobs
      Character (1)                      :: errfn, link, mean, offset, weight
!     .. Local Arrays ..
      Real (Kind=nag_wp), Allocatable    :: b(:), cov(:), eta(:), pred(:),       &
                                            se(:), seeta(:), sepred(:), t(:),    &
                                            v(:,:), wk(:), x(:,:), y(:)
      Real (Kind=nag_wp)                 :: off(1), wt(1)
      Integer, Allocatable               :: isx(:), state(:)
      Integer                            :: seed(lseed)
!     .. Intrinsic Procedures ..
      Intrinsic                          :: ceiling, count, int, real
!     .. Executable Statements ..
      Write (nout,*) 'G05PVF Example Program Results'
      Write (nout,*)

!     Skip heading in data file
      Read (nin,*)

!     Set variables required by the regression (G02GBF) ...

!     Read in the type of link function, whether a mean is required
!     and the problem size
      Read (nin,*) link, mean, n, m

!     Set storage order for G05PVF (pick the one required by G02GBF and
!     G02GPF)
      sordx = 1

      ldx = n
      Allocate (x(ldx,m),y(n),t(n),isx(m))

!     This example is not using an offset or weights
      offset = 'N'
      weight = 'U'

!     Read in data
      Read (nin,*)(x(i,1:m),y(i),t(i),i=1,n)

!     Read in variable inclusion flags
      Read (nin,*) isx(1:m)

!     Read in control parameters for the regression
      Read (nin,*) iprint, eps, tol, maxit

!     Calculate IP
      ip = count(isx(1:m)>0)
      If (mean=='M' .Or. mean=='m') Then
        ip = ip + 1
      End If
!     ... End of setting variables required by the regression

!     Set variables required by data sampling routine (G05PVF) ...

!     Read in the base generator information and seed
      Read (nin,*) genid, subid, seed(1:lseed)

!     Will always have a Y and T variable
      usey = 1
      uset = 1

!     Query the required size of the STATE array
      lstate = 0
      Allocate (state(lstate))
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)
```

```
!     Reallocate STATE
      Deallocate (state)
      Allocate (state(lstate))

!     Initialize the generator to a repeatable sequence
      ifail = 0
      Call g05kff(genid,subid,seed,lseed,state,lstate,ifail)

!     Read in the number of folds
      Read (nin,*) k

!     ... End of setting variables required by data sampling routine

!     Set variables required by prediction routine (G02GPF) ...

!     Regression is performed using G02GBF so error structure is binomial
      errfn = 'B'

!     This example does not use the predicted standard errors, so
!     it doesn't matter what VFOBS is set to
      vfobs = .False.
!     ... End of setting variables required by prediction routine

!     This is the maximum size for a validation dataset
      max_nv = ceiling(real(n,kind=nag_wp)/real(k,kind=nag_wp))

!     Allocate arrays
      ldv = n
      lwk = (ip*ip+3*ip+22)/2
      Allocate (b(ip),se(ip),cov(ip*(ip+1)/2),v(ldv,ip+7),wk(lwk))
      Allocate (eta(max_nv),seeta(max_nv),pred(max_nv),sepred(max_nv))

!     Initialize counts
      tp = 0
      tn = 0
      fp = 0
      fn = 0

!     Loop over each fold
      Do fold = 1, k
!       Split the data into training and validation datasets
        ifail = -1
        Call g05pvf(k,fold,n,m,sordx,x,ldx,usey,y,uset,t,nt,state,ifail)
        If (ifail/=0 .And. ifail/=61) Then
          Go To 100
        End If

!       Calculate the size of the validation dataset
        nv = n - nt

!       Call routine to fit generalized linear model, with Binomial errors
!       to training data
        ifail = -1
        Call g02gbf(link,mean,offset,weight,nt,x,ldx,m,isx,ip,y,t,wt,dev,idf,  &
          b,irank,se,cov,v,ldv,tol,maxit,iprint,eps,wk,ifail)
        If (ifail/=0) Then
          If (ifail<6) Then
            Go To 100
          End If
        End If

!       Predict the response for the observations in the validation dataset
        ifail = 0
        Call g02gpf(errfn,link,mean,offset,weight,nv,x(nt+1,1),ldx,m,isx,ip,   &
          t(nt+1),off,wt,s,a,b,cov,vfobs,eta,seeta,pred,sepred,ifail)

!       Count the true/false positives/negatives
        Do i = 1, nv
          obs_val = int(y(nt+i))

          If (pred(i)>=0.5_nag_wp) Then
```

```
            pred_val = 1
          Else
            pred_val = 0
          End If

          Select Case (obs_val)
          Case (0)
!           Negative
            Select Case (pred_val)
            Case (0)
!             True negative
              tn = tn + 1
            Case (1)
!             False positive
              fp = fp + 1
            End Select
          Case (1)
!           Positive
            Select Case (pred_val)
            Case (0)
!             False negative
              fn = fn + 1
            Case (1)
!             True positive
              tp = tp + 1
            End Select
          End Select
        End Do
      End Do

!     Display results
      np = tp + fn
      nn = fp + tn

      Write (*,99998) '                      Observed'
      Write (*,99998) '            --------------------------'
      Write (*,99998) 'Predicted | Negative  Positive   Total'
      Write (*,99998) '--------------------------------------'
      Write (*,99997) 'Negative  |', tn, fn, tn + fn
      Write (*,99997) 'Positive  |', fp, tp, fp + tp
      Write (*,99997) 'Total     |', nn, np, nn + np
      Write (*,*)

      If (np/=0) Then
        Write (nout,99999) 'True Positive Rate (Sensitivity):',                &
          real(tp,kind=nag_wp)/real(np,kind=nag_wp)
      Else
        Write (nout,99998)                                                     &
          'True Positive Rate (Sensitivity): No positives in data'
      End If
      If (nn/=0) Then
        Write (nout,99999) 'True Negative Rate (Specificity):',                &
          real(tn,kind=nag_wp)/real(nn,kind=nag_wp)
      Else
        Write (nout,99998)                                                     &
          'True Negative Rate (Specificity): No negatives in data'
      End If

100   Continue
99999 Format (1X,A,F5.2)
99998 Format (1X,A)
99997 Format (1X,A,1X,I5,5X,I5,5X,I5)
    End Program g05pvfe
```

10.2 Program Data

```
G05PVF Example Program Data
 'G'  'M'  40  4                    :: LINK, MEAN, N, M
 0.0 -0.1  0.0  1.0      0.0  1.0
 0.4 -1.1  1.0  1.0      1.0  1.0
-0.5  0.2  1.0  0.0      0.0  1.0
 0.6  1.1  1.0  0.0      0.0  1.0
-0.3 -1.0  1.0  1.0      0.0  1.0
 2.8 -1.8  0.0  1.0      0.0  1.0
 0.4 -0.7  0.0  1.0      1.0  1.0
-0.4 -0.3  1.0  0.0      1.0  1.0
 0.5 -2.6  0.0  0.0      1.0  1.0
-1.6 -0.3  1.0  1.0      0.0  1.0
 0.4  0.6  1.0  0.0      0.0  1.0
-1.6  0.0  1.0  1.0      1.0  1.0
 0.0  0.4  1.0  1.0      1.0  1.0
-0.1  0.7  1.0  1.0      0.0  1.0
-0.2  1.8  1.0  1.0      0.0  1.0
-0.9  0.7  1.0  1.0      0.0  1.0
-1.1 -0.5  1.0  1.0      0.0  1.0
-0.1 -2.2  1.0  1.0      1.0  1.0
-1.8 -0.5  1.0  1.0      1.0  1.0
-0.8 -0.9  0.0  1.0      1.0  1.0
 1.9 -0.1  1.0  1.0      1.0  1.0
 0.3  1.4  1.0  1.0      0.0  1.0
 0.4 -1.2  1.0  0.0      1.0  1.0
 2.2  1.8  1.0  0.0      1.0  1.0
 1.4 -0.4  0.0  1.0      1.0  1.0
 0.4  2.4  1.0  1.0      0.0  1.0
-0.6  1.1  1.0  1.0      0.0  1.0
 1.4 -0.6  1.0  1.0      1.0  1.0
-0.1 -0.1  0.0  0.0      0.0  1.0
-0.6 -0.4  0.0  0.0      0.0  1.0
 0.6 -0.2  1.0  1.0      1.0  1.0
-1.8 -0.3  1.0  1.0      1.0  1.0
-0.3  1.6  1.0  1.0      0.0  1.0
-0.6  0.8  0.0  1.0      0.0  1.0
 0.3 -0.5  0.0  0.0      1.0  1.0
 1.6  1.4  1.0  1.0      0.0  1.0
-1.1  0.6  1.0  1.0      0.0  1.0
-0.3  0.6  1.0  1.0      0.0  1.0
-0.6  0.1  1.0  1.0      0.0  1.0
 1.0  0.6  1.0  1.0      1.0  1.0 :: End of X, Y, T
 1    1    1    1                   :: ISX
0  0.0  0.0  0                      :: IPRINT, EPS, TOL, MAXIT
6 0 42321                           :: GENID, SUBID, SEED
5                                   :: K
```

10.3 Program Results

```
 G05PVF Example Program Results

                         Observed
             --------------------------
 Predicted | Negative  Positive   Total
 --------------------------------------
 Negative  |    18         8        26
 Positive  |     4        10        14
 Total     |    22        18        40

 True Positive Rate (Sensitivity): 0.56
 True Negative Rate (Specificity): 0.82
```