

# NAG Library Routine Document

## G05PWF

**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

G05PWF generates a dataset suitable for use with repeated random sub-sampling validation.

### 2 Specification

```
SUBROUTINE G05PWF (NT, N, M, SORDX, X, LDX, USEY, Y, USEW, W, STATE,      &
                  IFAIL)
INTEGER          NT, N, M, SORDX, LDX, USEY, USEW, 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.

G05PWF 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 by randomly assigning each observation to either the training dataset or the validation dataset.

The resulting datasets are suitable for use with repeated random sub-sampling validation.

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 G05PWF.

### 4 References

None.

### 5 Parameters

- |    |  |              |
|----|--|--------------|
| 1: | NT – INTEGER   | <i>Input</i> |
|    | <i>On entry:</i> $n_t$ , the number of observations in the training dataset. |              |
|    | <i>Constraint:</i> $1 \leq NT \leq N$ .                                      |              |
| 2: | N – INTEGER  | <i>Input</i> |
|    | <i>On entry:</i> $n$ , the number of observations.                           |              |
|    | <i>Constraint:</i> $N \geq 1$ .  |              |
| 3: | M – INTEGER  | <i>Input</i> |
|    | <i>On entry:</i> $m$ , the number of variables.                              |              |
|    | <i>Constraint:</i> $M \geq 1$ .  |              |

- 4: SORDX – INTEGER *Input*  
*On entry:* determines how variables are stored in X.  
*Constraint:* SORDX = 1 or 2.
- 5: X(LDX,\*) – REAL (KIND=nag\_wp) array *Input/Output*  
**Note:** the second dimension of the array X must be at least M if SORDX = 1 and at least N if SORDX = 2.  
 The way the data is stored in X is defined by SORDX.  
 If SORDX = 1,  $X(i, j)$  contains the  $i$ th observation for the  $j$ th variable, for  $i = 1, 2, \dots, N$  and  $j = 1, 2, \dots, M$ .  
 If SORDX = 2,  $X(j, i)$  contains the  $i$ th observation for the  $j$ th variable, for  $i = 1, 2, \dots, N$  and  $j = 1, 2, \dots, M$ .  
*On entry:* X must hold  $X_o$ , the values of X for the original dataset. This may be the same X as returned by a previous call to G05PWF.  
*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 NT + 1 to N.
- 6: LDX – INTEGER *Input*  
*On entry:* the first dimension of the array X as declared in the (sub)program from which G05PWF is called.  
*Constraints:*  
     if SORDX = 2,  $LDX \geq M$ ;  
     otherwise  $LDX \geq N$ .
- 7: USEY – INTEGER *Input*  
*On entry:* if USEY = 1, the original dataset includes  $y_o$  and  $y_o$  will be processed alongside  $X_o$ .  
*Constraint:* USEY = 0 or 1.
- 8: Y(\*) – REAL (KIND=nag\_wp) array *Input/Output*  
**Note:** the dimension of the array Y must be at least N if USEY = 1.  
 If USEY = 0, Y is not referenced on entry and will not be modified on exit.  
*On entry:* Y must hold  $y_o$ , the values of y for the original dataset. This may be the same Y as returned by a previous call to G05PWF.  
*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 NT + 1 to N.
- 9: USEW – INTEGER *Input*  
*On entry:* if USEW = 1, the original dataset includes  $w_o$  and  $w_o$  will be processed alongside  $X_o$ .  
*Constraint:* USEW = 0 or 1.
- 10: W(\*) – REAL (KIND=nag\_wp) array *Input/Output*  
**Note:** the dimension of the array W must be at least N if USEW = 1.  
 If USEW = 0, W is not referenced on entry or and will not be modified on exit.  
*On entry:* W must hold  $w_o$ , the values of w for the original dataset. This may be the same W as returned by a previous call to G05PWF.  
*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 NT + 1 to N.

11: 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.

12: IFAIL – INTEGER *Input/Output*

*On entry:* IFAIL must be set to 0, -1 or 1. If you are unfamiliar with this parameter you should refer to Section 3.3 in the Essential Introduction 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, if you are not familiar with this parameter, the recommended value is 0. **When the value -1 or 1 is used it is essential to test the value of IFAIL on exit.**

*On exit:* 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 IFAIL = 0 or -1, explanatory error messages are output on the current error message unit (as defined by X04AAF).

Errors or warnings detected by the routine:

IFAIL = 11

On entry, NT =  $\langle value \rangle$  and N =  $\langle value \rangle$ .  
Constraint:  $1 \leq NT \leq N$ .

IFAIL = 21

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

IFAIL = 31

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

IFAIL = 41

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

IFAIL = 61

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

IFAIL = 62

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

IFAIL = 71

Constraint: USEY = 0 or 1.

IFAIL = 91

Constraint: USEW = 0 or 1.

IFAIL = 111

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 3.8 in the Essential Introduction for further information.

IFAIL = -399

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

See Section 3.7 in the Essential Introduction for further information.

IFAIL = -999

Dynamic memory allocation failed.

See Section 3.6 in the Essential Introduction for further information.

## 7 Accuracy

Not applicable.

## 8 Parallelism and Performance

G05PWF is threaded by NAG for parallel execution in multithreaded implementations of the NAG Library.

G05PWF 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

G05PWF will be computationally more efficient if each observation in X is contiguous, that is SORDX = 2.

## 10 Example

This example uses G05PWF to facilitate repeated random sub-sampling cross-validation.

A set of simulated data is randomly split into a 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. This process is repeated 10 times.

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

## 10.1 Program Text

```

Program g05pwfe

!      G05PWF Example Program Text

!      Mark 25 Release. NAG Copyright 2014.

!      .. Use Statements ..
Use nag_library, Only: g02gbf, g02gpf, g05kff, g05pwf, 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, fp, genid, i, idf, ifail, ip,      &
                             iprint, irank, ldv, ldx, lstate,      &
                             lwk, m, maxit, n, nn, np, nsamp, nt, &
                             nv, obs_val, pred_val, samp, sordx, &
                             subsid, 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                  :: count, int, real
!      .. Executable Statements ..
Write (nout,*) 'G05PWF 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 G05PWF (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 (G05PWF) ...

!      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 size of the training set required
      Read (nin,*) nt

!      Read in the number of sub-samples we will use */
      Read (nin,*) nsamp
!      ... 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

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

!      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(nv),seeta(nv),pred(nv),sepred(nv))

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

!      Loop over each sample
      Do samp = 1, nsamp
!      Split the data into training and validation datasets
      ifail = 0
      Call g05pwf(nt,n,m,sordx,x,ldx,usey,y,uset,t,state,ifail)

!      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) '
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 g05pwfe
    
```

### 10.2 Program Data

```

G05PWF 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
  32          :: NT
  10          :: NSAMP
    
```

### 10.3 Program Results

G05PWF Example Program Results

Predicted	Observed		
	Negative	Positive	Total
Negative	38	20	58



Positive		8	14	22
Total		46	34	80

True Positive Rate (Sensitivity): 0.41  
True Negative Rate (Specificity): 0.83

---