# NAG Toolbox

# nag_rand_subsamp_xyw (g05pw)

## 1 Purpose

nag_rand_subsamp_xyw (g05pw) generates a dataset suitable for use with repeated random sub-sampling validation.

## 2 Syntax

```
[state, sx, sy, sw, errbuf, ifail] = nag_rand_subsamp_xyw(nt, x, state, 'n', n, 'm', m, 'sordx', sordx, 'y', y, 'w', w, 'sordsx', sordsx)
[state, sx, sy, sw, errbuf, ifail] = g05pw(nt, x, state, 'n', n, 'm', m, 'sordx', sordx, 'y', y, 'w', w, 'sordsx', sordsx)
```

## 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.

nag_rand_subsamp_xyw (g05pw) 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 functions nag_rand_init_repeat (g05kf) (for a repeatable sequence if computed sequentially) or nag_rand_init_nonrepeat (g05kg) (for a non-repeatable sequence) must be called prior to the first call to nag_rand_subsamp_xyw (g05pw).

## 4 References

None.

## 5 Parameters

### 5.1 Compulsory Input Parameters

1: **nt** – INTEGER

$n_t$, the number of observations in the training dataset.

*Constraint*: $1 \leq$ **nt** $\leq$ **n**.

2: **x**$(ldx, :)$ – REAL (KIND=nag_wp) array

The first dimension, $ldx$, of the array **x** must satisfy

if **sordx** $= 2$, $ldx \geq$ **m**;
otherwise $ldx \geq$ **n**.

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 $i = 1, 2, \ldots,$ **n** and $j = 1, 2, \ldots,$ **m**.

If **sordx** $= 2$, $\mathbf{x}(j,i)$ contains the $i$th observation for the $j$th variable, for $i = 1,2,\ldots,\mathbf{n}$ and $j = 1,2,\ldots,\mathbf{m}$.

$X_o$, the values of $X$ for the original dataset. This may be the array returned in **sx** by a previous call to nag_rand_subsamp_xyw (g05pw).

3: **state**(:) – INTEGER array

**Note**: the actual argument supplied **must** be the array **state** supplied to the initialization routines nag_rand_init_repeat (g05kf) or nag_rand_init_nonrepeat (g05kg).

Contains information on the selected base generator and its current state.

## 5.2 Optional Input Parameters

1: **n** – INTEGER

*Default*:

if **sordx** $= 2$, the second dimension of **x**;
otherwise the first dimension of **x**.

$n$, the number of observations.

*Constraint*: **n** $\geq 1$.

2: **m** – INTEGER

*Default*:

if **sordx** $= 2$, the first dimension of **x**;
otherwise the second dimension of **x**.

$m$, the number of variables.

*Constraint*: **m** $\geq 1$.

3: **sordx** – INTEGER

*Default*: 1

Determines how variables are stored in **x**.

*Constraint*: **sordx** $= 1$ or 2.

4: $\mathbf{y}(ly)$ – REAL (KIND=nag_wp) array

Optionally, $y_o$, the values of $y$ for the original dataset. This may be the vector returned in **sy** by a previous call to nag_rand_subsamp_xyw (g05pw).

5: $\mathbf{w}(lw)$ – REAL (KIND=nag_wp) array

Optionally, $w_o$, the values of $w$ for the original dataset. This may be the vector returned in **sw** by a previous call to nag_rand_subsamp_xyw (g05pw).

6: **sordsx** – INTEGER

*Default*: **sordx**

Determines how variables are stored in **sx**.

*Constraint*: **sordsx** $= 1$ or 2.

## 5.3 Output Parameters

1: **state**(:) – INTEGER array

Contains updated information on the state of the generator.

2: **sx**$(ldsx,:)$ – REAL (KIND=nag_wp) array

The first dimension, $ldsx$, of the array **sx** will be

if **sordsx** $= 1$, $ldsx =$ **n**;
if **sordsx** $= 2$, $ldsx =$ **m**.

The second dimension of the array **sx** will be **m** if **sordsx** $= 1$ and **n** otherwise.

The way the data is stored in **sx** is defined by **sordsx**.

If **sordsx** $= 1$, **sx**$(i,j)$ contains the $i$th observation for the $j$th variable, for $i = 1,2,\ldots,$ **n** and $j = 1,2,\ldots,$ **m**.

If **sordsx** $= 2$, **sx**$(j,i)$ contains the $i$th observation for the $j$th variable, for $i = 1,2,\ldots,$ **n** and $j = 1,2,\ldots,$ **m**.

**sx** holds the 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**.

3: **sy**$(lsy)$ – REAL (KIND=nag_wp) array

If **y** is not **NULL** then **sy** holds the 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**.

4: **sw**$(lsw)$ – REAL (KIND=nag_wp) array

If **w** is not **NULL** then **sw** holds the 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**.

5: **errbuf** – CHARACTER(200)

6: **ifail** – INTEGER

**ifail** $= 0$ unless the function detects an error (see Section 5).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**ifail** $= 11$

Constraint: $1 \leq$ **nt** $\leq$ **n**.

**ifail** $= 21$

Constraint: **n** $\geq 1$.

**ifail** $= 31$

Constraint: **m** $\geq 1$.

**ifail** $= 41$

Constraint: **sordx** $= 1$ or 2.

**ifail** $= 61$

Constraint: if **sordx** $= 1$, $ldx \geq$ **n**.

**ifail** $= 62$

Constraint: if **sordx** $= 2$, $ldx \geq$ **m**.

**ifail** $= 111$

On entry, **state** vector has been corrupted or not initialized.

**ifail** $= 141$

Constraint: **sordsx** $= 1$ or $2$.

**ifail** $= -99$

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

**ifail** $= -399$

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

**ifail** $= -999$

Dynamic memory allocation failed.

## 7 Accuracy

Not applicable.

## 8 Further Comments

nag_rand_subsamp_xyw (g05pw) will be computationality more efficient if each observation in **x** are contiguous, that is **sordx** $= 2$ and **sordsx** $= 2$.

## 9 Example

This example uses nag_rand_subsamp_xyw (g05pw) to facilitate repeated random sub-sampling cross-validation.

A set of simulated data is randomly split into a training and validation datasets. nag_correg_glm_binomial (g02gb) is used to fit a logistic regression model to each training dataset and then nag_correg_glm_predict (g02gp) 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.

### 9.1 Program Text

```
function g05pw_example

fprintf('g05pw example results\n\n');

% Fit a logistic regression model using g02gb and predict values using g02gp
% (binomial error, logistic link, with an intercept)
link = 'G';
mean = 'M';
errfn = 'B';

% Not usind the predicted standard errors
vfobs = false;

% Independent variables
x = [ 0.0 -0.1  0.0  1.0;   0.4 -1.1  1.0  1.0;  -0.5  0.2  1.0  0.0;
      0.6  1.1  1.0  0.0;  -0.3 -1.0  1.0  1.0;   2.8 -1.8  0.0  1.0;
      0.4 -0.7  0.0  1.0;  -0.4 -0.3  1.0  0.0;   0.5 -2.6  0.0  0.0;
     -1.6 -0.3  1.0  1.0;   0.4  0.6  1.0  0.0;  -1.6  0.0  1.0  1.0;
      0.0  0.4  1.0  1.0;  -0.1  0.7  1.0  1.0;  -0.2  1.8  1.0  1.0;
     -0.9  0.7  1.0  1.0;  -1.1 -0.5  1.0  1.0;  -0.1 -2.2  1.0  1.0;
     -1.8 -0.5  1.0  1.0;  -0.8 -0.9  0.0  1.0;   1.9 -0.1  1.0  1.0;
```

```
      0.3  1.4  1.0  1.0;   0.4 -1.2  1.0  0.0;   2.2  1.8  1.0  0.0;
      1.4 -0.4  0.0  1.0;   0.4  2.4  1.0  1.0;  -0.6  1.1  1.0  1.0;
      1.4 -0.6  1.0  1.0;  -0.1 -0.1  0.0  0.0;  -0.6 -0.4  0.0  0.0;
      0.6 -0.2  1.0  1.0;  -1.8 -0.3  1.0  1.0;  -0.3  1.6  1.0  1.0;
     -0.6  0.8  0.0  1.0;   0.3 -0.5  0.0  0.0;   1.6  1.4  1.0  1.0;
     -1.1  0.6  1.0  1.0;  -0.3  0.6  1.0  1.0;  -0.6  0.1  1.0  1.0;
      1.0  0.6  1.0  1.0];

% Dependent variable
y = [0;1;0;0;0;0;1;1;1;0;0;1;1;0;0;0;0;1;1;1;
     1;0;1;1;1;0;0;1;0;0;1;1;0;0;1;0;0;0;0;1];

% Each observation represents a single trial
t = ones(size(x,1));

% Include all independent variables in the model
isx = nag_int(ones(size(x,2),1));
ip = nag_int(sum(isx) + (upper(mean(1:1)) == 'M'));

% In order to use cross-validation we need to initialise the random
% number generator (using L'Ecuyers MRG32k3a and a repeatable sequence)
seed = nag_int(42321);
genid = nag_int(6);
subid = nag_int(0);
[state,ifail] = g05kf(genid,subid,seed);

% generate 10 random sub-samples
nsamp = nag_int(10);

% size of sub-samples to generate (size of training dataset)
nt = nag_int(32);
% size validation dataset is n - nt

% Some of the routines used in this example issue warnings, but return
% sensible results, so save current warning state and turn warnings on
warn_state = nag_issue_warnings();
nag_issue_warnings(true);

tn = 0;
fn = 0;
fp = 0;
tp = 0;

%  Loop over each sample
for i = 1:nsamp

  % Split the data into training and validation datasets
  [state,x,y,t,ifail] = g05pw( ...
                               nt,x,state,'y',y,'w',t);
  if (ifail~=0)
    break
  end

  % Call routine to fit generalized linear model, with Binomial errors
  % to training data (the first nt values in x)
  [~,~,b,~,~,cov,~,ifail] = g02gb( ...
                                   link,mean,x,isx,ip,y,t,'n',nt);
  if (ifail~=0 & ifail < 6)
    break
  end

  % Predict the response for the observations in the validation dataset
  [~,~,pred,~,ifail] = g02gp( ...
                              errfn,link,mean,x(nt+1:end,:),isx,b,cov, ...
                              vfobs, 't',t(nt+1:end));
  if (ifail~=0)
    break
  end

  % Cross-tabulate the observed and predicted values
  obs_val = ceil(y(nt+1:end) + 0.5);
```

```
  pred_val = (pred >= 0.5) + 1;
  count = zeros(2,2);
  for i = 1:size(pred_val,1)
    count(pred_val(i),obs_val(i)) = count(pred_val(i),obs_val(i)) + 1;
  end

  % Extract the true/false negatives/positives
  tn = tn + count(1,1);
  fn = fn + count(1,2);
  fp = fp + count(2,1);
  tp = tp + count(2,2);
end

% Reset the warning state to its initial value
nag_issue_warnings(warn_state);

np = tp + fn;
nn = fp + tn;

fprintf('                       Observed\n');
fprintf('             -------------------------\n');
fprintf(' Predicted | Negative  Positive   Total\n');
fprintf(' ------------------------------------\n');
fprintf(' Negative  | %5d     %5d     %5d\n', tn, fn, tn + fn);
fprintf(' Positive  | %5d     %5d     %5d\n', fp, tp, fp + tp);
fprintf(' Total     | %5d     %5d     %5d\n', nn, np, nn + np);
fprintf('\n');

if (np~=0)
  fprintf(' True Positive Rate (Sensitivity): %4.2f\n', tp / np);
else
  fprintf(' True Positive Rate (Sensitivity): No positives in data\n');
end
if (nn~=0)
  fprintf(' True Negative Rate (Specificity): %4.2f\n', tn / nn);
else
  fprintf(' True Negative Rate (Specificity): No negatives in data\n');
end
```

## 9.2 Program Results

```
     g05pw example results

                    Observed
             -------------------------
 Predicted | 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
```