

# Package: MultiDiscreteRNG (via r-universe)

May 27, 2026

**Type** Package

**Title** Generate Multivariate Discrete Data

**Version** 0.1.0

**Maintainer** Chak Kwong (Tommy) Cheng <ccheng46@uic.edu>

**Description** Generate multivariate discrete data with generalized Poisson, negative binomial and binomial marginal distributions using user-specified distribution parameters and a target correlation matrix. The method is described in Cheng and Demirtas (2026) <[doi:10.48550/arXiv.2602.07707](https://doi.org/10.48550/arXiv.2602.07707)>.

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**Imports** GenOrd, Matrix, MultiOrd, matrixcalc, mvtnorm

**URL** <https://github.com/ckchengtommy/MultiDiscreteRNG>

**BugReports** <https://github.com/ckchengtommy/MultiDiscreteRNG/issues>

**RoxygenNote** 7.3.3

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**Config/pak/sysreqs** make

**Repository** <https://ckchengtommy.r-universe.dev>

**Date/Publication** 2026-02-20 14:18:22 UTC

**RemoteUrl** <https://github.com/ckchengtommy/multidiscreterng>

**RemoteRef** HEAD

**RemoteSha** 196b0c716fcc6a6165bb0464460ad11ba491f060

## Contents

BinToB	2
BinToGPD	3
BinToMix	4
BinToNB	6
calc.bin.prob.B	7
calc.bin.prob.GPD	8
calc.bin.prob.NB	8
discrete_cont	9
genB	10
generate.binaryVar	11
genGPD	12
genMix	13
genNB	14
GetGpoisPMF	15
QuantileGpois	16
simBinaryCorr.B	16
simBinaryCorr.GPD	17
simBinaryCorr.Mix	19
simBinaryCorr.NB	20
validation.Bparameters	21
validation.GPDparameters	22
validation.NBparameters	23
<b>Index</b>	<b>24</b>

---

BinToB	<i>Convert multivariate binary data back to the original binomial scale</i>
--------	---

---

### Description

This function maps multivariate binary data to multivariate binomial outcomes, preserving the original marginal distribution characteristics. Given the binary representation of the data, the function assigns binomial values based on the original probability mass functions and the location of the median split.

### Usage

```
BinToB(prop.vec.bin, BProp, Mlocation, bin.data)
```

### Arguments

prop.vec.bin	A vector of binary probabilities
BProp	Binary proportion
Mlocation	Indices of the medians in the vector
bin.data	Generated multivariate binary data.

**Value**

A list containing the multivariate binomial data and its correlation matrix

**Examples**

```
# Generate binary probabilities and probability mass functions for 3 variables
B.n.vec <- c(3, 4, 5)
B.prob.vec <- c(0.5, 0.5, 0.5)
p <- calc.bin.prob.B(B.n.vec, B.prob.vec)
pvec <- p$p
prop <- p$prop
Mlocation <- p$Mlocation

# Select the first two variables for demonstration
pvec.pair <- pvec[1:2]
Mlocation.pair <- Mlocation[1:2]
prop.pair <- list(prop[[1]], prop[[2]])

# Specify a target correlation matrix for two binary variables
del.next <- matrix(c(1.0, 0.3,
                    0.3, 1.0),
                  nrow = 2, byrow = TRUE)

# Simulate N = 100 binary observations with the desired correlation
inter_bin <- generate.binaryVar(100, pvec.pair, del.next)

# Convert back to binomial scale
Mydata <- BinToB(pvec.pair, prop.pair, Mlocation.pair, inter_bin)
```

---

 BinToGPD

*Convert multivariate binary data back to the original generalized Poisson scale*

---

**Description**

This function maps multivariate binary data to multivariate generalized Poisson outcomes, preserving the original marginal distribution characteristics. Given a binary representation, it assigns generalized Poisson values based on the original probability mass functions and the location of the median split for each variable.

**Usage**

```
BinToGPD(prop.vec.bin, GPDprop, Mlocation, bin.data)
```

**Arguments**

prop.vec.bin	A vector of binary probabilities
GPDprop	Generalized Poisson distribution probability mass functions tables
Mlocation	Indices of the medians in the vector
bin.data	Generated multivariate binary data

**Value**

A list containing the multivariate generalized Poisson data and its correlation matrix

**Examples**

```
# Prepare three GPD parameter vectors
GPD.lambda.vec <- c(0.1, 0.2, 0.3)
GPD.theta.vec  <- c(7, 0.7, 40)

# Compute binary probabilities, PMFs, and thresholds
p      <- calc.bin.prob.GPD(GPD.theta.vec, GPD.lambda.vec)
pvec   <- p$p
prop   <- p$prop
Mloc   <- p$Mlocation

# Use only the first two variables for demonstration
pvec.pair   <- pvec[1:2]
Mlocation.pair <- Mloc[1:2]
prop.pair   <- list(prop[[1]], prop[[2]])

# Define a 2 by2 target correlation matrix
del.next <- matrix(c(1.0, 0.3,
                    0.3, 1.0),
                  nrow = 2, byrow = TRUE)

# Simulate 100 correlated binary observations
inter_bin <- generate.binaryVar(100, pvec.pair, del.next)

# Reconstruct the GPD scaled data
Mydata <- BinToGPD(pvec.pair, prop.pair, Mlocation.pair, inter_bin)
```

**Description**

This function maps multivariate binary data to outcomes from a mixture of generalized Poisson, negative binomial, and binomial distributions while preserving the original marginal distribution characteristics. It assigns appropriate values from each distribution based on binary thresholds and probability mass functions.

**Usage**

```
BinToMix(prop.vec.bin, Mixprop, Mlocation, bin.data)
```

**Arguments**

prop.vec.bin    A vector of binary probabilities for each variable

Mixprop        A list of probability mass functions for each variable's distribution

Mlocation      A vector of threshold values (typically medians) for each variable

bin.data       A matrix of multivariate binary data (0s and 1s)

**Value**

A list containing:

y              Matrix of generated mixed distribution data

Corr           Correlation matrix of the generated data

**Examples**

```
# First simulate intermediate binary correlations
GPD.theta.vec = 2
GPD.lambda.vec = 0.3
NB.r.vec = 10
NB.prob.vec = 0.2

GPD.p = calc.bin.prob.GPD(GPD.theta.vec, GPD.lambda.vec)
NB.p = calc.bin.prob.NB(NB.r.vec, NB.prob.vec)
pvec.pair = c(GPD.p$p, NB.p$p)
Mlocation.pair <- c(GPD.p$Mlocation, NB.p$Mlocation)
prop.pair <- list(GPD.p$prop[[1]], NB.p$prop[[1]])

# Specify a target correlation matrix for two binary variables
del.next <- matrix(c(1.0, 0.3,
                    0.3, 1.0),
                  nrow = 2, byrow = TRUE)

# Generate correlated binary data using the intermediate matrix
inter_bin <- generate.binaryVar(100, pvec.pair, del.next)

# Convert binary data to mixed distribution outcomes
mixed_data <- BinToMix(pvec.pair, prop.pair, Mlocation.pair, inter_bin)$y
```

---

BinToNB	<i>Convert multivariate binary data back to the original negative binomial scale</i>
---------	--

---

### Description

This function maps multivariate binary data to multivariate negative binomial outcomes, preserving the original marginal distribution characteristics. Given a binary representation, the function assigns negative binomial values based on the original probability mass functions and the location of the median split for each variable.

### Usage

```
BinToNB(prop.vec.bin, NBprop, Mlocation, bin.data)
```

### Arguments

prop.vec.bin	A numeric vector of binary probabilities
NBprop	A numeric value or vector of negative binomial proportions
Mlocation	Integer indices of the medians in the vector
bin.data	A data frame or matrix of generated multivariate binary data

### Value

A list containing the multivariate negative binomial data and its correlation matrix

### Examples

```
NB.r.vec <- c(10, 3, 16)
NB.prob.vec <- c(0.65, 0.4, 0.88)

# Compute binary probabilities, PMFs, and thresholds
p <- calc.bin.prob.NB(NB.r.vec, NB.prob.vec)
pvec <- p$p
prop <- p$prop
Mloc <- p$Mlocation

# Use only the first two variables for demonstration
pvec.pair <- pvec[1:2]
Mlocation.pair <- Mloc[1:2]
prop.pair <- list(prop[[1]], prop[[2]])

# Define a 2 by 2 target correlation matrix
del.next <- matrix(c(1.0, -0.3,
                    -0.3, 1.0),
                  nrow = 2, byrow = TRUE)

# Simulate 100 correlated binary observations
```

```
inter_bin <- generate.binaryVar(100, pvec.pair, del.next)

# Reconstruct the negative binomial scaled data
Mydata <- BinToNB(pvec.pair, prop.pair, Mlocation.pair, inter_bin)
```

---

calc.bin.prob.B      *Collapse binomial data outcomes to binary variable*

---

### Description

This function computes the binary probability and identifies a threshold to split discrete binomial outcomes. It summarizes each binomial distribution by providing the probability mass function, the probability of exceeding a median-based threshold, and the location of the threshold for the binary split.

### Usage

```
calc.bin.prob.B(B.n.vec, B.prob.vec)
```

### Arguments

B.n.vec            Numeric vector of trial counts for each variable  
B.prob.vec        Numeric vector of success probabilities for each variable

### Value

A list containing the binary probability vector, the list of binomial probability mass functions for each variable, and the corresponding threshold indices

### Examples

```
B.n.vec <- c(3, 4, 5)
B.prob.vec <- c(0.5, 0.5, 0.5)
p <- calc.bin.prob.B(B.n.vec, B.prob.vec)
```

---

calc.bin.prob.GPD      *Collapse discrete generalized Poisson outcomes to binary variables*

---

### Description

This function implements Step 1 of the algorithm. It collapses each discrete outcome from the generalized Poisson distribution (GPD) into a binary probability and determines a dichotomous threshold for each variable.

### Usage

```
calc.bin.prob.GPD(GPD.theta.vec, GPD.lambda.vec)
```

### Arguments

GPD.theta.vec    Numeric vector of GPD theta parameters.  
GPD.lambda.vec   Numeric vector of GPD lambda parameters.

### Value

A list containing the binary probability vector, the list of GPD probability mass functions for each variable, and the corresponding threshold indices.

### Examples

```
# Prepare three GPD parameter vectors  
GPD.lambda.vec <- c(0.1, 0.2, 0.3)  
GPD.theta.vec  <- c(7, 0.7, 40)  
  
# Compute binary probabilities, PMFs, and thresholds  
p <- calc.bin.prob.GPD(GPD.theta.vec, GPD.lambda.vec)
```

---

calc.bin.prob.NB      *Collapse discrete negative binomial outcomes to binary variables*

---

### Description

This function implements Step 1 of the algorithm. It collapses each discrete outcome from the negative binomial distribution into a binary probability and determines a dichotomous threshold for each variable.

### Usage

```
calc.bin.prob.NB(NB.r.vec, NB.prob.vec)
```

**Arguments**

NB.r.vec            vector of number of trials  
 NB.prob.vec        vector of probabilities

**Value**

vector of binary probability, dichotomous threshold

**Examples**

```
NB.r.vec <- c(10, 3, 16)
NB.prob.vec <- c(0.65, 0.4, 0.88)

# Compute binary probabilities, PMFs, and thresholds
p <- calc.bin.prob.NB(NB.r.vec, NB.prob.vec)
```

---

discrete_cont	<i>Compute the tetrachoric correlation matrix for a multivariate standard normal distribution</i>
---------------	---

---

**Description**

This function calculates the intermediate correlation matrix of a multivariate standard normal distribution in Step 2 of the algorithm. If the resulting matrix is not positive definite, the nearest positive definite matrix is returned and a warning is issued.

**Usage**

```
discrete_cont(
  marginal,
  Sigma,
  support = list(),
  Spearman = FALSE,
  epsilon = 1e-06,
  maxit = 100
)
```

**Arguments**

marginal            a list of  $k$  elements, where  $k$  is the number of variables. The  $i$ -th element of marginal is the vector of the cumulative probabilities defining the marginal distribution of the  $i$ -th component of the multivariate variable. If the  $i$ -th component can take  $k_i$  values, the  $i$ -th element of marginal will contain  $k_i - 1$  probabilities (the  $k_i$ -th is obviously 1 and shall not be included).

Sigma                the target correlation matrix of the discrete variables

support	a list of $k$ elements, where $k$ is the number of variables. The $i$ -th element of support is the vector containing the ordered values of the support of the $i$ -th variable. By default, the support of the $i$ -th variable is $1, 2, \dots, k_i$
Spearman	A logical flag indicating whether Spearman correlation should be used
epsilon	tolerance of the algorithm convergence
maxit	maximum iterations of the algorithm to correct PD matrix

### Value

No return values; called it to check parameter inputs

### References

Ferrari and Barbiero 2012 (<<https://doi.org/10.1080/00273171.2012.692630>>)

### Examples

```
prop.vec.bin = c(0.5037236, 0.5034147)
cor.mat = matrix(c(1, 0.3, 0.3, 1), nrow = 2, byrow = TRUE)
InterMVN_Sigma = discrete_cont(marginal = prop.vec.bin, Sigma = cor.mat)$SigmaC
```

---

genB

*Generate multivariate binomial data*

---

### Description

This function is the engine for simulating correlated binomial outcomes once the intermediate binary parameters have been computed. It first generates correlated multivariate binary data using a latent normal approach implemented in `generate.binaryVar` and then maps the binary outcomes back to the original binomial scales via a reverse-collapsing step implemented in `BinToB`, using the binomial probability mass function and dichotomization locations stored in `binObj`

### Usage

```
genB(no.rows, binObj)
```

### Arguments

no.rows	integer; number of observations to generate (sample size $N$ ).
binObj	list; intermediate object produced by the binary-correlation calibration step for binomial margins (e.g., <code>simBinaryCorr.B</code> ). It must contain: <p>pvec numeric vector of binary success probabilities <math>p_j^b</math>.</p> <p>intermat intermediate/tetrachoric correlation matrix for the latent normal model used to generate correlated binary data.</p> <p>BProp list of binomial PMFs (probability masses over the support) used in the reverse-collapsing step.</p> <p>Mlocation numeric vector of dichotomization thresholds (median locations) used to split each marginal distribution into binary categories.</p>

**Value**

A generated multivariate binomial dataset (returned as a list containing the simulated data matrix and its empirical correlation matrix).

**Examples**

```
n.vec <- c(3, 4)
p.vec <- c(0.5, 0.5)

M<- c(0.3, 0.4)
N <- diag(2)
N[lower.tri(N)] <- M
cmat<- N + t(N)
diag(cmat) <- 1
#In real data simulation, no.rows should set to 100000 for accurate data generation
#in the intermediate step
binObj = simBinaryCorr.B(B.n.vec = n.vec, B.prob.vec = p.vec,
CorrMat = cmat, no.rows = 20000, steps= 0.025)

data = genB(no.rows = 100, binObj = binObj)$y
```

---

generate.binaryVar	<i>Generate multivariate Binary data using the Emrich and Piedmonte (1991) approach Approach</i>
--------------------	--

---

**Description**

Generate multivariate Binary data using the Emrich and Piedmonte (1991) approach Approach

**Usage**

```
generate.binaryVar(nObs, prop.vec.bin, corr.mat)
```

**Arguments**

nObs	number of observations
prop.vec.bin	probability of binary variables in a vector
corr.mat	Correlation matrix

**Value**

multivariate Binary Data

genGPD

*Generate multivariate generalized Poisson data***Description**

This function is the engine for simulating correlated generalized Poisson outcomes once the intermediate binary parameters have been computed. It first generates correlated multivariate binary data using a latent normal approach implemented in `generate.binaryVar` and then maps the binary outcomes back to the original generalized Poisson scales via a reverse-collapsing step implemented in `BinToGPD`, using the generalized Poisson probability mass function and dichotomization locations stored in `binObj`.

**Usage**

```
genGPD(no.rows, binObj)
```

**Arguments**

<code>no.rows</code>	integer; number of observations to generate (sample size $N$ ).
<code>binObj</code>	list; intermediate object produced by the binary-correlation calibration step for generalized Poisson margins (e.g., <code>simBinaryCorr.GPD</code> ). It must contain: <ul style="list-style-type: none"> <li><code>pvec</code> numeric vector of binary success probabilities <math>p_j^b</math>.</li> <li><code>intermat</code> intermediate/tetrachoric correlation matrix for the latent normal model used to generate correlated binary data.</li> <li><code>GPDprop</code> list of generalized Poisson PMFs (probability masses over the support) used in the reverse-collapsing step.</li> <li><code>Mlocation</code> numeric vector of dichotomization thresholds (median locations) used to split each marginal distribution into binary categories.</li> </ul>

**Value**

A generated multivariate generalized Poisson dataset (returned as a list containing the simulated data matrix and its empirical correlation matrix).

**Examples**

```
lambda.vec <- c(0.1, 0.05)
theta.vec <- c(7, 12)
M<- c(0.3, 0.3, 0.3)
N <- diag(2)
N[lower.tri(N)] <- M
cmat<- N + t(N)
diag(cmat) <- 1

# In real data simulation, no.rows should set to 100000 for accurate data generation
# in the intermediate step.
binObj = simBinaryCorr.GPD(GPD.theta.vec = theta.vec, GPD.lambda.vec = lambda.vec,
```

```

                                CorrMat = cmat, no.rows = 20000, steps= 0.025)
data = genGPD(no.rows = 100, binObj = binObj)$y

```

---

genMix

*Generate multivariate mixed discrete data*


---

### Description

This function is the engine for simulating correlated mixed discrete outcomes once the intermediate binary parameters have been computed. It first generates correlated multivariate binary data using a latent normal approach implemented in `generate.binaryVar` and then maps the binary outcomes back to the original mixed discrete scales via a reverse-collapsing step implemented in `BinToMix`, using the component probability mass functions and dichotomization locations stored in `binObj`.

### Usage

```
genMix(no.rows, binObj)
```

### Arguments

<code>no.rows</code>	integer; number of observations to generate (sample size $N$ ).
<code>binObj</code>	list; intermediate object produced by the binary-correlation calibration step for mixed discrete margins (e.g., <code>simBinaryCorr.Mix</code> ). It must contain: <ul style="list-style-type: none"> <li><code>pvec</code> numeric vector of binary success probabilities.</li> <li><code>intermat</code> intermediate/tetrachoric correlation matrix for the latent normal model used to generate correlated binary data.</li> <li><code>Mixprop</code> list of probability mass functions (probability masses over the support) for each discrete margin, used in the reverse-collapsing step.</li> <li><code>Mlocation</code> numeric vector of dichotomization thresholds (median locations) used to split each marginal distribution into binary categories.</li> </ul>

### Value

A generated multivariate mixed discrete dataset (returned as a list containing the simulated data matrix and its empirical correlation matrix).

### Examples

```

GPD.theta = 4
GPD.lambda = 0.2
NB.r = 15
NB.prob = 0.42
M<- c(0.15, 0.2)
N <- diag(2)
N[lower.tri(N)] <- M
cmat<- N + t(N)

```

```
diag(cmat) <- 1
binObj = simBinaryCorr.Mix(GPD.theta.vec = GPD.theta, GPD.lambda.vec = GPD.lambda,
                          NB.r.vec = NB.r, NB.prob.vec = NB.prob,
                          CorrMat =cmat, no.rows = 20000, steps= 0.025)
Mix.data = genMix(no.rows = 100, binObj)$y
```

---

genNB

*Generate multivariate negative binomial data*


---

### Description

This function is the engine for simulating correlated negative binomial outcomes once the intermediate binary parameters have been computed. It first generates correlated multivariate binary data using a latent normal approach implemented in `generate.binaryVar` and then maps the binary outcomes back to the original negative binomial scales via a reverse-collapsing step implemented in `BinToNB`, using the negative binomial probability mass function and dichotomization locations stored in `binObj`.

### Usage

```
genNB(no.rows, binObj)
```

### Arguments

<code>no.rows</code>	integer; number of observations to generate (sample size $N$ ).
<code>binObj</code>	list; intermediate object produced by the binary-correlation calibration step for negative binomial margins (e.g., <code>simBinaryCorr.NB</code> ). It must contain: <ul style="list-style-type: none"> <li><code>pvec</code> numeric vector of binary success probabilities <math>p_j^b</math>.</li> <li><code>intermat</code> intermediate/tetrachoric correlation matrix for the latent normal model used to generate correlated binary data.</li> <li><code>NBprop</code> list of negative binomial PMFs (probability masses over the support) used in the reverse-collapsing step.</li> <li><code>Mlocation</code> numeric vector of dichotomization thresholds (median locations) used to split each marginal distribution into binary categories.</li> </ul>

### Value

A generated multivariate negative binomial dataset (returned as a list containing the simulated data matrix and its empirical correlation matrix).

**Examples**

```

r.vec <- c(3, 5)
p.vec <- c(0.7, 0.5)

M<- c(0.2, 0.3)
N <- diag(2)
N[lower.tri(N)] <- M
cmat<- N + t(N)
diag(cmat) <- 1

# In real data simulation, no.rows should set to 100000 for accurate data generation
# in the intermediate step.
binObj = simBinaryCorr.NB(NB.r.vec = r.vec, NB.prob.vec = p.vec, CorrMat = cmat,
no.rows = 20000, steps= 0.025)

data = genNB(no.rows = 100, binObj = binObj)$y

```

---

GetGpoisPMF

*Get probability mass function of generalized Poisson distribution*


---

**Description**

This function returns a table of the probability mass function of generalized Poisson distribution.

**Usage**

```
GetGpoisPMF(p, theta, lambda, details = FALSE)
```

**Arguments**

p	Probability level(s) used to determine how far to compute the support (the PMF is generated until the CDF exceeds $\max(p)$ ).
theta	Theta parameter(s) for the generalized Poisson distribution.
lambda	Lambda parameter(s) for the generalized Poisson distribution.
details	Logical; if TRUE, additional computation information is printed.

**Value**

A named numeric vector containing the generalized Poisson PMF values over the computed support (names correspond to support points  $x = 0, 1, 2, \dots$ ). Very small probabilities (below  $1e-10$ ) are removed.

**Examples**

```

# PMF values computed until the CDF exceeds p = 1
gpd_pmf <- GetGpoisPMF(p = 1, theta = 0.2, lambda = 0.3)
gpd_pmf

```

---

<code>QuantileGpois</code>	<i>Compute the quantile function of the generalized Poisson distribution</i>
----------------------------	--

---

**Description**

This function evaluates the generalized Poisson quantile  $Q(p)$  by incrementally constructing the PMF and CDF from  $x = 0$  upward until the CDF exceeds the largest requested probability in  $p$ . The returned quantile(s) are the smallest integer  $x$  such that  $P(X \leq x) \geq p$ .

**Usage**

```
QuantileGpois(p, theta, lambda, details = FALSE)
```

**Arguments**

<code>p</code>	vector of probabilities
<code>theta</code>	vector of theta
<code>lambda</code>	vector of lambda
<code>details</code>	A logical flag to return the computational details

**Value**

An integer vector of generalized Poisson quantiles corresponding to  $p$ .

**Examples**

```
QuantileGpois(p = 0.95, theta = 2, lambda = 0.1)
```

---

<code>simBinaryCorr.B</code>	<i>Compute intermediate binary correlations for multivariate binomial data</i>
------------------------------	--

---

**Description**

This function implements Step 2 of the algorithm to calibrate the intermediate latent-normal correlation matrix used to generate correlated binary variables. For each pair of variables, it iteratively updates the latent correlation so that, after (i) generating correlated binary data via `generate.binaryVar` and (ii) mapping back to binomial outcomes via `BinToB`, the empirical correlation of the resulting binomial pair matches the user-specified target correlation in `CorrMat`. The calibrated pairwise latent correlations are then assembled into a full intermediate matrix, which is adjusted to be positive definite if needed.

**Usage**

```
simBinaryCorr.B(B.n.vec, B.prob.vec, CorrMat, no.rows, steps = 0.025)
```

**Arguments**

B.n.vec	vector of number of trials
B.prob.vec	vector of probabilities
CorrMat	specified Correlation matrix
no.rows	number of observations for generating Multivariate Binary data
steps	Fraction of difference between the current and target matrix to be added in each iteration.

**Value**

intermediate multivariate binary Correlation matrix

**Examples**

```
n.vec <- c(3, 4)
p.vec <- c(0.5, 0.5)

M<- c(0.3, 0.2)
N <- diag(2)
N[lower.tri(N)] <- M
cmat<- N + t(N)
diag(cmat) <- 1
# In real data simulation, no.rows should set to 100000 for accurate data
# generation in the intermediate step
binObj = simBinaryCorr.B(B.n.vec = n.vec, B.prob.vec = p.vec, CorrMat = cmat,
no.rows = 20000, steps= 0.025)
```

---

simBinaryCorr.GPD	<i>Compute intermediate binary correlations for multivariate generalized Poisson data</i>
-------------------	---

---

**Description**

This function implements Step 2 of the algorithm to calibrate the intermediate latent-normal correlation matrix used to generate correlated binary variables for generalized Poisson (GPD) margins. For each pair of variables, it iteratively updates the latent correlation so that, after (i) generating correlated binary data via `generate.binaryVar` and (ii) mapping back to GPD outcomes via `BinToGPD`, the empirical correlation of the resulting GPD pair matches the user-specified target correlation in `CorrMat`. The calibrated pairwise latent correlations are then assembled into a full intermediate matrix, which is adjusted to be positive definite if needed.

**Usage**

```
simBinaryCorr.GPD(
  GPD.theta.vec,
  GPD.lambda.vec,
  CorrMat,
  no.rows,
  steps = 0.025
)
```

**Arguments**

<code>GPD.theta.vec</code>	vector of theta values
<code>GPD.lambda.vec</code>	vector of lambda values
<code>CorrMat</code>	specified Correlation matrix
<code>no.rows</code>	number of observations for generating Multivariate Binary data
<code>steps</code>	Fraction of difference between the current and target matrix to be added in each iteration.

**Value**

intermediate multivariate binary Correlation matrix

**Examples**

```
lambda.vec <- c(0.1, 0.13)
theta.vec <- c(7, 40)
M<- c(0.3, 0.3)
N <- diag(2)
N[lower.tri(N)] <- M
cmat<- N + t(N)
diag(cmat) <- 1

# In real-data simulation, no.rows is often set to 100000 in this intermediate step
# for more accurate calibration.
binObj <- simBinaryCorr.GPD(
  GPD.theta.vec = theta.vec,
  GPD.lambda.vec = lambda.vec,
  CorrMat       = cmat,
  no.rows       = 20000,
  steps         = 0.025)
```

---

simBinaryCorr.Mix      *Calculate intermediate binary correlations for mixed data*

---

### Description

This function implements Step 2 of the algorithm to calibrate the intermediate latent-normal correlation matrix used to generate correlated binary variables for a mixture of generalized Poisson (GPD), negative binomial (NB), and binomial (B) margins. For each pair of variables, it iteratively updates the latent correlation so that, after (i) generating correlated binary data via `generate.binaryVar` and (ii) mapping back to the mixed discrete scales via `BinToMix`, the empirical correlation of the resulting mixed pair matches the user-specified target correlation in `CorrMat`. The calibrated pairwise latent correlations are then assembled into a full intermediate matrix, which is adjusted to be positive definite if needed (via `Matrix::nearPD`).

### Usage

```
simBinaryCorr.Mix(
  GPD.theta.vec = NULL,
  GPD.lambda.vec = NULL,
  NB.r.vec = NULL,
  NB.prob.vec = NULL,
  B.n.vec = NULL,
  B.prob.vec = NULL,
  CorrMat,
  no.rows,
  steps = 0.025
)
```

### Arguments

<code>GPD.theta.vec</code>	Numeric vector of theta parameters for GPD variables (or 'NULL' if none).
<code>GPD.lambda.vec</code>	Numeric vector of lambda parameters for GPD variables (must match length of 'GPD.theta.vec').
<code>NB.r.vec</code>	Numeric vector of dispersion parameters ('r') for NB variables (or 'NULL' if none).
<code>NB.prob.vec</code>	Numeric vector of success probabilities for NB variables (must match length of 'NB.r.vec').
<code>B.n.vec</code>	Numeric vector of number of trials for Binomial variables (or 'NULL' if none).
<code>B.prob.vec</code>	Numeric vector of success probabilities for Binomial variables (must match length of 'B.n.vec').
<code>CorrMat</code>	Correlation matrix (must be symmetric positive definite with dimensions matching total variables).
<code>no.rows</code>	Integer specifying the number of rows (samples) to generate during intermediate binary sampling.
<code>steps</code>	Numeric step size (default = 0.025) for correlation adjustment in later iterations.

**Details**

The function first calculates binary probabilities and properties for each distribution family (GPD, NB, Binomial) using helper functions `'calc.bin.prob.GPD'`, `'calc.bin.prob.NB'`, and `'calc.bin.prob.B'`. It then iteratively adjusts pairwise correlations in binary space to match the target correlation structure, using a step size for convergence. If the intermediate matrix is not positive definite, it is adjusted using `'Matrix::nearPD'`.

**Value**

A list containing:

Mixprop	List of proportions for each variable's binary components.
intermat	Intermediate correlation matrix for binary variables (adjusted to be positive definite if needed).
Mlocation	List of location parameters for each variable.
pvec	Vector of binary probabilities for each variable.

**Examples**

```
GPD.theta = 4
GPD.lambda = 0.03
NB.r = 15
NB.prob = 0.61
M<- c(0.15, 0.2)
N <- diag(2)
N[lower.tri(N)] <- M
cmat<- N + t(N)
diag(cmat) <- 1
binObj = simBinaryCorr.Mix(GPD.theta.vec = GPD.theta, GPD.lambda.vec = GPD.lambda,
                           NB.r.vec = NB.r, NB.prob.vec = NB.prob,
                           CorrMat = cmat, no.rows = 20000, steps= 0.025)
```

---

simBinaryCorr.NB	<i>Compute intermediate binary correlations for multivariate negative binomial data</i>
------------------	---

---

**Description**

This function implements Step 2 of the algorithm to calibrate the intermediate latent-normal correlation matrix used to generate correlated binary variables for negative binomial margins. For each pair of variables, it iteratively updates the latent correlation so that, after (i) generating correlated binary data via `generate.binaryVar` and (ii) mapping back to negative binomial outcomes via `BinToNB`, the empirical correlation of the resulting NB pair matches the user-specified target correlation in `CorrMat`. The calibrated pairwise latent correlations are then assembled into a full intermediate matrix, which is adjusted to be positive definite if needed.

**Usage**

```
simBinaryCorr.NB(NB.r.vec, NB.prob.vec, CorrMat, no.rows, steps = 0.025)
```

**Arguments**

NB.r.vec	vector of number of trials
NB.prob.vec	vector of probabilities
CorrMat	specified correlation matrix
no.rows	number of observations for generating multivariate binary data
steps	fraction of difference between the current and target matrix to be added in each iteration.

**Value**

intermediate multivariate binary Correlation matrix

**Examples**

```
r.vec <- c(3, 5)
p.vec <- c(0.7, 0.5)

M<- c(0.45, 0.45)
N <- diag(2)
N[lower.tri(N)] <- M
cmat<- N + t(N)
diag(cmat) <- 1

# In real data simulation, no.rows should set to 100000 for accurate data generation
# in the intermediate step.
binObj = simBinaryCorr.NB(NB.r.vec = r.vec, NB.prob.vec = p.vec, CorrMat = cmat,
no.rows = 20000, steps= 0.025)
```

---

validation.Bparameters

*Validate binomial parameters are within feasible ranges*

---

**Description**

This helper function checks that binomial inputs are valid before downstream probability calculations and data generation. If any check fails, the function stops with an informative error message.

**Usage**

```
validation.Bparameters(B.n.vec, B.prob.vec)
```

**Arguments**

B.n.vec            Vector of number of trials  
B.prob.vec        Vector of probability

**Value**

No return values; called it to check parameter inputs

**Examples**

```
validation.Bparameters(B.n.vec = c(10, 15), B.prob.vec = c(0.4, 0.2))
```

---

```
validation.GPDparameters
```

*Validate generalized Poisson parameters are within feasible ranges*

---

**Description**

This helper function checks that generalized Poisson (GPD) inputs are valid before downstream probability calculations and data generation. If any check fails, the function stops with an informative error message.

**Usage**

```
validation.GPDparameters(GPD.theta.vec, GPD.lambda.vec)
```

**Arguments**

GPD.theta.vec    Vector of theta values  
GPD.lambda.vec   Vector of lambda values

**Value**

No return values; called it to check parameter inputs

**Examples**

```
validation.GPDparameters(GPD.theta.vec = c(3, 2), GPD.lambda.vec = c(0.4, 0.2))
```

---

validation.NBparameters

*Validate if the input negative binomial parameters are within feasible range*

---

**Description**

Validate if the input negative binomial parameters are within feasible range

**Usage**

```
validation.NBparameters(NB.r.vec, NB.prob.vec)
```

**Arguments**

NB.r.vec            Vector of number of trials parameters  
NB.prob.vec        Vector of probabilities

**Value**

No return values; called it to check parameter inputs

**Examples**

```
validation.NBparameters(NB.r.vec = c(10, 15), NB.prob.vec = c(0.7, 0.5))
```

# Index

[BinToB](#), [2](#)  
[BinToGPD](#), [3](#)  
[BinToMix](#), [4](#)  
[BinToNB](#), [6](#)

[calc.bin.prob.B](#), [7](#)  
[calc.bin.prob.GPD](#), [8](#)  
[calc.bin.prob.NB](#), [8](#)

[discrete\\_cont](#), [9](#)

[genB](#), [10](#)  
[generate.binaryVar](#), [11](#)  
[genGPD](#), [12](#)  
[genMix](#), [13](#)  
[genNB](#), [14](#)  
[GetGpoisPMF](#), [15](#)

[QuantileGpois](#), [16](#)

[simBinaryCorr.B](#), [16](#)  
[simBinaryCorr.GPD](#), [17](#)  
[simBinaryCorr.Mix](#), [19](#)  
[simBinaryCorr.NB](#), [20](#)

[validation.Bparameters](#), [21](#)  
[validation.GPDparameters](#), [22](#)  
[validation.NBparameters](#), [23](#)