

# Package ‘CDsampling’

January 11, 2025

**Type** Package

**Title** 'CDsampling': Constraint Sampling in Paid Research Studies

**Version** 0.1.4

**Maintainer** Yifei Huang <yhuan39@uic.edu>

**Description** In the context of paid research studies and clinical trials, budget considerations and patient sampling from available populations are subject to inherent constraints. We introduce the 'CDsampling' package, which integrates optimal design theories within the framework of constrained sampling. This package offers the possibility to find both D-optimal approximate and exact allocations for samplings with or without constraints. Additionally, it provides functions to find constrained uniform sampling as a robust sampling strategy with limited model information. Our package offers functions for the computation of the Fisher information matrix under generalized linear models (including regular linear regression model) and multinomial logistic models. To demonstrate the applications, we also provide a simulated dataset and a real dataset embedded in the package. Yifei Huang, Liping Tong, and Jie Yang (2025) <[doi:10.5705/ss.202022.0414](https://doi.org/10.5705/ss.202022.0414)>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**Imports** lpSolve, Rglpk, stats

**RoxygenNote** 7.3.1

**Depends** R (>= 2.10)

**Suggests** knitr, rmarkdown, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Yifei Huang [aut, cre],  
Liping Tong [aut],  
Jie Yang [aut]

**Repository** CRAN

**Date/Publication** 2025-01-11 09:40:08 UTC

## Contents

approxtoexact_constrained_func . . . . .	2
approxtoexact_func . . . . .	3
bounded_uniform . . . . .	4
Fdet_func_GLM . . . . .	5
Fdet_func_MLM . . . . .	5
Fdet_func_unif . . . . .	7
Fi_func_MLM . . . . .	8
F_func_GLM . . . . .	9
F_func_MLM . . . . .	9
iset_func_trauma . . . . .	11
iset_func_trial . . . . .	12
liftone_constrained_GLM . . . . .	13
liftone_constrained_MLM . . . . .	15
liftone_GLM . . . . .	19
liftone_MLM . . . . .	20
print.list_output . . . . .	22
print.matrix_list . . . . .	23
print.matrix_output . . . . .	23
trauma_data . . . . .	24
trial_data . . . . .	24
W_func_GLM . . . . .	25
<b>Index</b>	<b>26</b>

---

approxtoexact\_constrained\_func

*Convert the approximate allocation (proportion) to exact allocation (integer) with bounded constraint ( $n_i \leq N_i$ )*

---

### Description

Convert the approximate allocation (proportion) to exact allocation (integer) with bounded constraint ( $n_i \leq N_i$ )

### Usage

```
approxtoexact_constrained_func(
  n,
  w,
  m,
  beta = NULL,
  link = NULL,
  X = NULL,
  Fdet_func = Fdet_func_GLM,
  iset_func = NULL
)
```

**Arguments**

n	Sample size, must be a positive integer
w	Approximate allocation/proportion, must be a real-valued vector, can get from running <code>liftone_constrained_GLM</code> or <code>liftone_constrained_MLM</code>
m	The number of sampling groups
beta	Model parameter coefficients, default to be NULL for use in constrained uniform sampling
link	Link function of GLM or MLM, if used for GLM model (GLM_T is T), options are "identity", "logit", "probit", "cloglog", "loglog". If used for MLM (GLM_T is F), options are "continuation", "cumulative", "adjacent", and "baseline"
X	Design matrix of the model for GLM or MLM, default to be NULL for use in constrained uniform sampling
Fdet_func	determinant of Fisher information matrix function, Fdet_func can be self-defined, or use "Fdet_func_GLM", "Fdet_func_MLM" in the package, default is Fdet_func_GLM
iset_func	self-defined function for checking which index of sampling group fall within constraint if add 1 more subject (I set, see Algorithm 2 in Huang, Tong, Yang (2023)), two example functions are provided in the package, <code>iset_func_trial</code> and <code>iset_func_trauma</code>

**Value**

allocation is the exact allocation or integer value of the number of subjects sampled from the group  
allocation.real is the proportion or the approximate allocation of the number of subjects sampled from the group  
det.maximum is the maximum of |F| from the current exact allocation

**Examples**

```
beta = c(0, 3, 3, 3) #main effect model beta_0, beta_1, beta_21, beta_22
X.liftone=matrix(data=c(1,0,0,0,1,0,1,0,1,0,0,1,1,1,0,0,1,1,1,0,1,1,0,1,1,0,1), ncol=4, byrow=TRUE)
exact_design = approxtoexact_constrained_func(n=200, w=c(0.25, 0.20, 0.05, 0.50, 0.00, 0.00),
m=6, beta=beta, link='logit', X=X.liftone, Fdet_func=Fdet_func_GLM, iset_func=iset_func_trial)
```

---

`approxtoexact_func`      *Convert the approximate allocation (proportion) to exact allocation (integer) without constraint*

---

**Description**

Convert the approximate allocation (proportion) to exact allocation (integer) without constraint

**Usage**

```
approxtoexact_func(n, w)
```

**Arguments**

- n                    Sample size, must be a positive integer
- w                    Approximate allocation/proportion, must be a real-valued vector, can get from running `liftone_constrained_GLM` or `liftone_constrained_MLM`

**Value**

allocation is the exact allocation or integer value of the number of subjects sampled from the group

**Examples**

```
exact_design = approxtoexact_func(n=600, w=c(0.2593526, 0.0000000, 0.0000000,
0.1565024, 0.2891565, 0.0000000, 0.0000000, 0.2949885))
```

---

bounded_uniform	<i>Find (constrained) uniform exact allocation of the study for bounded design</i>
-----------------	--

---

**Description**

Find (constrained) uniform exact allocation of the study for bounded design

**Usage**

```
bounded_uniform(Ni, nsample)
```

**Arguments**

- Ni                    a vector with size m, upper bound for exact design of each category/stratification group, if unconstrained, use Inf vector, the function will return unbounded uniform allocation
- nsample              a number, the sample size

**Value**

n is the constrained/unconstrained uniform exact allocation

**Examples**

```
bounded_uniform(Ni=c(50, 40, 10, 200, 150, 50), nsample=200)
```

---

Fdet_func_GLM	<i>Determinant of Fisher information matrix for GLM</i>
---------------	---

---

**Description**

Determinant of Fisher information matrix for GLM

**Usage**

```
Fdet_func_GLM(w, beta, X, link = "logit")
```

**Arguments**

w	allocation (can be exact or approximate)
beta	GLM model covariate coefficient
X	model matrix
link	link function, default "logit", choose from "logit", "cloglog", "loglog", "probit", and "identity"(for regular linear regression)

**Value**

the determinant of Fisher information matrix given X and model parameter beta

**Examples**

```
w = c(1/3, 1/3, 1/3)
beta = c(0.5, 0.5, 0.5)
X = matrix(data=c(1, -1, -1, 1, -1, 1, 1, 1, -1), byrow=TRUE, nrow=3)
Fdet_func_GLM(w=w, beta=beta, X=X, link='logit')
```

---

Fdet_func_MLM	<i>Determinant of Fisher information matrix of multinomial logistic model (MLM)</i>
---------------	---

---

**Description**

Determinant of Fisher information matrix of multinomial logistic model (MLM)

**Usage**

```
Fdet_func_MLM(w, beta, X, link)
```

**Arguments**

w	allocation (can be exact or approximate)
beta	MLM model covariate coefficient
X	MLM model matrix
link	link function of Multinomial logistic regression model, options are "baseline", "cumulative", "adjacent", or "continuation"

**Value**

Determinant of the Fisher information matrix of MLM model

**Examples**

```
w = rep(1/8, 8)
Xi=rep(0,5*12*8) #response levels * num of parameters * num of design points
dim(Xi)=c(5,12,8)
#design matrix
Xi[,,1] = rbind(c( 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0))

Xi[,,2] = rbind(c( 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0))

Xi[,,3] = rbind(c( 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0))

Xi[,,4] = rbind(c( 1, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 4, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 4, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 0))

Xi[,,5] = rbind(c( 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1))

Xi[,,6] = rbind(c( 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1))
```

```

c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[,7] = rbind(c( 1, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 3, 1, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 1),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[,8] = rbind(c( 1, 4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 4, 1, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 4, 1, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 1),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

bvec_temp = c(-4.047, -2.225, -0.302, 1.386, 4.214, 3.519,
              2.420, 1.284, -0.131, -0.376, -0.237, -0.120)
link_temp = "cumulative"

Fdet_func_MLM(w=w, beta=bvec_temp, X=Xi, link=link_temp)

```

---

Fdet_func_unif	<i>Determinant function to be used for finding constrained uniform samplings</i>
----------------	--

---

**Description**

Determinant function to be used for finding constrained uniform samplings

**Usage**

```
Fdet_func_unif(w, beta = NULL, X = NULL, link = NULL)
```

**Arguments**

w	allocation (can be exact or approximate)
beta	use NULL (default to be NULL)
X	use NULL (default to be NULL)
link	use NULL (default to be NULL)

**Value**

product of all allocation

**Examples**

```
Fdet_func_unif(w=c(0.2,0.2,0.2,0.2,0.2))
```

---

Fi_func_MLM	<i>Generate Fisher information matrix <math>F_x</math> at a design point <math>x_i</math> for Multinomial logistic regression model</i>
-------------	---

---

### Description

Generate Fisher information matrix  $F_x$  at a design point  $x_i$  for Multinomial logistic regression model

### Usage

```
Fi_func_MLM(X_x, beta, link)
```

### Arguments

<code>X_x</code>	model matrix given design point $x_i$ (for example, $X_x = h.func(x_i)$ , where $h.func$ transforms a design point to a model matrix)
<code>beta</code>	parameter coefficients in the Multinomial logistic regression model, the order of coefficients in <code>bvec</code> and the order of design points in <code>X_x</code> should be consistent
<code>link</code>	link function of Multinomial logistic regression model, options are "baseline", "cumulative", "adjacent", or "continuation"

### Value

$F_x$  is the Fisher information matrix at design point  $x_i$  (with model matrix  $X_x$ );

$U_x$  is a middle step matrix for calculation of  $F_x$ , details see Corollary 3.1 in Bu, X., Majumdar, D., & Yang, J. (2020). D-optimal designs for multinomial logistic models

### Examples

```
X_x_temp = rbind(c( 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                 c( 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0),
                 c( 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0),
                 c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0),
                 c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
bvec_temp = c(-4.047, -2.225, -0.302, 1.386, 4.214, 3.519,
              2.420, 1.284, -0.131, -0.376, -0.237, -0.120)
link_temp = "cumulative"
Fi_func_MLM(X_x=X_x_temp, beta=bvec_temp, link=link_temp)
```



---

F\_func\_GLM

*Fisher information matrix of generalized linear model (GLM)*


---

**Description**

Fisher information matrix of generalized linear model (GLM)

**Usage**

```
F_func_GLM(w, beta, X, link = "logit")
```

**Arguments**

w	allocation (can be exact or approximate)
beta	GLM model covariate coefficient
X	model matrix
link	link function, default "logit", choose from "logit", "cloglog", "loglog", "probit", and "identity"(for regular linear regression)

**Value**

object of class "matrix\_output", Fisher information matrix given X and model parameter beta

**Examples**

```
w = c(1/3,1/3, 1/3)
beta = c(0.5, 0.5, 0.5)
X = matrix(data=c(1,-1,-1,1,-1,1,1,1,-1), byrow=TRUE, nrow=3)
F_func_GLM(w=w, beta=beta, X=X, link='logit')
```

---

F\_func\_MLM

*The Fisher information matrix of multinomial logistic model (MLM)*


---

**Description**

The Fisher information matrix of multinomial logistic model (MLM)

**Usage**

```
F_func_MLM(w, beta, X, link)
```

**Arguments**

w	allocation (can be exact or approximate)
beta	MLM model covariate coefficient
X	MLM model matrix
link	link function of Multinomial logistic regression model, options are "baseline", "cumulative", "adjacent", or "continuation"

**Value**

The Fisher information matrix of MLM model

**Examples**

```
w = rep(1/8, 8)
Xi=rep(0,5*12*8) #response levels * num of parameters * num of design points
dim(Xi)=c(5,12,8)
#design matrix
Xi[,,1] = rbind(c( 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0))

Xi[,,2] = rbind(c( 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0))

Xi[,,3] = rbind(c( 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0))

Xi[,,4] = rbind(c( 1, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 4, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 4, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 0))

Xi[,,5] = rbind(c( 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1))

Xi[,,6] = rbind(c( 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 1))
```

```

c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[,7] = rbind(c( 1, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 3, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 1),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[,8] = rbind(c( 1, 4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 4, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 4, 1, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 1),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

bvec_temp = c(-4.047, -2.225, -0.302, 1.386, 4.214, 3.519, 2.420,
              1.284, -0.131, -0.376, -0.237, -0.120)
link_temp = "cumulative"

F_func_MLM(w=w, beta=bvec_temp, X=Xi, link=link_temp)

```

---

iset_func_trauma	<i>trauma_data</i> example (see Huang, Tong, Yang (2023)) specific function for finding index set that if allocation of that index add "1", the new allocation still falls within the constraint Used in <i>approxtoexact_constrained_func()</i>
------------------	--

---

## Description

*trauma\_data* example (see Huang, Tong, Yang (2023)) specific function for finding index set that if allocation of that index add "1", the new allocation still falls within the constraint Used in *approxtoexact\_constrained\_func()*

## Usage

```
iset_func_trauma(allocation)
```

## Arguments

`allocation`      the exact allocation

## Value

list of TRUE and FALSE, if TRUE, it means the allocation of this index will fall out of the constraint with more subject; if TURE, it means the allocation of this index can add more subjects

**Examples**

```
iset_func_trauma(allocation=c(50,30,10,10,100,100,200,10))
```

---

iset_func_trial	<i>trial_data example (see Huang, Tong, Yang (2023)) specific function for finding index set that if allocation of that index add "1", the new allocation still falls within the constraint Used in approx-toexact_constrained_func()</i>
-----------------	---

---

**Description**

trial\_data example (see Huang, Tong, Yang (2023)) specific function for finding index set that if allocation of that index add "1", the new allocation still falls within the constraint Used in approx-toexact\_constrained\_func()

**Usage**

```
iset_func_trial(allocation)
```

**Arguments**

allocation      the exact allocation

**Value**

list of TRUE and FALSE, if TRUE, it means the allocation of this index will fall out of the constraint with more subject; if TURE, it means the allocation of this index can add more subjects

**Examples**

```
iset_func_trial(allocation=c(50,30,10,100,100,40))
```

---

 liftone\_constrained\_GLM

*Find constrained D-optimal approximate design for generalized linear models (GLM)*

---

## Description

Find constrained D-optimal approximate design for generalized linear models (GLM)

## Usage

```
liftone_constrained_GLM(
  X,
  W,
  g.con,
  g.dir,
  g.rhs,
  lower.bound,
  upper.bound,
  reltol = 1e-05,
  maxit = 500,
  random = TRUE,
  nram = 3,
  w00 = NULL,
  epsilon = 1e-12
)
```

## Arguments

X	Model matrix, with nrow = num of design points and ncol = num of parameters
W	Diagonal of W matrix in Fisher information matrix, can be calculated from W_func_GLM in package
g.con	A matrix of numeric constraint coefficients, one row per constraint, on column per variable (to be used in as const.mat lp() and mat in Rglpk_solve_LP())
g.dir	Vector of character strings giving the direction of the constraint: each value should be one of "<," "<=," "=", "==" ,">," or ">=" . (In each pair the two values are identical.) to be used as const.dir in lp() and dir in Rglpk_solve_LP()
g.rhs	Vector of numeric values for the right-hand sides of the constraints. to be used as const.rhs in lp() and rhs in Rglpk_solve_LP().
lower.bound	A function to determine lower bound r_i1 in Step 3 of Constrained lift-one algorithm from Yifei, H., Liping, T., Yang, J. (2023) Constrained D-optimal design for paid research study
upper.bound	A function to determine upper bound r_i2 in Step 3 of Constrained lift-one algorithm from Yifei, H., Liping, T., Yang, J. (2023) Constrained D-optimal design for paid research study

reltol	The relative convergence tolerance, default value 1e-5
maxit	The maximum number of iterations, default value 500
random	TRUE or FALSE, if TRUE then the function will run with additional "nram" number of random initial points, default to be TRUE
nram	When random == TRUE, the function will generate nram number of initial points, default is 3
w00	Specified initial design proportion; default to be NULL, this will generate a random initial design
epsilon	A very small number, for comparison of >0, <0, ==0, to reduce errors, default 1e-12

### Value

w is the approximate D-optimal design

w0 is the initial design used to get optimal design w

maximum is the maximized |f| value

itmax is the number of iterations

convergence is TRUE or FALSE, if TRUE means the reported design is converged

deriv.ans is the derivative from step 6 of constrained lift-one algorithm

gmax is the maximum g function in step 8 of constrained lift-one algorithm

reason is the lift-one loops break reason, either "all derivatives <=0" or "gmax <=0"

### Examples

```
#Example 6 in Section 3.4 of Yifei, H., Liping, T., Yang, J. (2025)
#Constrained D-optimal design for paid research study

#main effect model beta_0, beta_1, beta_21, beta_22
beta = c(0, -0.1, -0.5, -2)

#gives the 6 categories (0,0,0), (0,1,0),(0,0,1),(1,0,0),(1,1,0),(1,0,1)
X.liftone=matrix(data=c(1,0,0,0,1,0,1,0,1,0,0,1,1,1,0,0,1,1,1,0,1,1,0,1),
ncol=4, byrow=TRUE)

#calculate W matrix based on beta's under logit link
W_matrix=W_func_GLM(X= X.liftone, b=beta)

m=6 #number of categories
nsample = 200
rc = c(50, 40, 10, 200, 150, 50)/nsample
g.con = matrix(0,nrow=(2*m+1), ncol=m)
g.con[1,] = rep(1, m)
g.con[2:(m+1),] = diag(m)
g.con[(m+2):(2*m+1), ] = diag(m)
g.dir = c("==", rep("<=", m), rep(">=", m))
g.rhs = c(1, rc, rep(0, m))
```

```

lower.bound=function(i, w){
  nsample = 200
  rc = c(50, 40, 10, 200, 150, 50)/nsample
  m=length(w) #num of categories
  temp = rep(0,m)
  temp[w>0]=1-pmin(1,rc[w>0])*(1-w[i])/w[w>0];
  temp[i]=0;
  max(0,temp);
}
upper.bound=function(i, w){
  nsample = 200
  rc = c(50, 40, 10, 200, 150, 50)/nsample
  m=length(w) #num of categories
  rc[i];
  min(1,rc[i]);
}

approximate_design = liftone_constrained_GLM(X=X.liftone, W=W_matrix,
g.con=g.con, g.dir=g.dir, g.rhs=g.rhs, lower.bound=lower.bound,
upper.bound=upper.bound, reltol=1e-10, maxit=100, random=TRUE, nram=4,
w00=NULL, epsilon = 1e-8)

```

---

liftone\_constrained\_MLM

*Find constrained D-optimal designs for Multinomial Logit Models (MLM)*

---

## Description

Find constrained D-optimal designs for Multinomial Logit Models (MLM)

## Usage

```

liftone_constrained_MLM(
  m,
  p,
  Xi,
  J,
  beta,
  lower.bound,
  upper.bound,
  g.con,
  g.dir,
  g.rhs,
  w00 = NULL,
  link = "cumulative",

```

```

    Fi.func = Fi_func_MLM,
    reltol = 1e-05,
    maxit = 500,
    delta = 1e-06,
    epsilon = 1e-08,
    random = TRUE,
    nram = 3
)

```

### Arguments

<code>m</code>	The number of design points; it is usually the number of combinations of all the stratification factors
<code>p</code>	The number of parameters in the MLM model
<code>Xi</code>	Model matrix, a J by p by m 3D array of predictors for separate response category at all design points(input to determine ppo, npo, po)
<code>J</code>	The number of response levels
<code>beta</code>	A p*1 vector, parameter coefficients for MLM, the order of beta should be consistent with Xi
<code>lower.bound</code>	A function to determine lower bound $r_{i1}$ in Step 3 of Constrained lift-one algorithm from Yifei, H., Liping, T., Yang, J. (2023) Constrained D-optimal design for paid research study
<code>upper.bound</code>	A function to determine upper bound $r_{i2}$ in Step 3 of Constrained lift-one algorithm from Yifei, H., Liping, T., Yang, J. (2023) Constrained D-optimal design for paid research study
<code>g.con</code>	A matrix of numeric constraint coefficients, one row per constraint, on column per variable (to be used in as <code>const.mat</code> in <code>lp()</code> and <code>mat</code> in <code>Rglpk_solve_LP()</code> )
<code>g.dir</code>	Vector of character strings giving the direction of the constraint: each value should be one of "<," "<=," "=", "==" ,">," or ">=" . (In each pair the two values are identical.) to be used as <code>const.dir</code> in <code>lp()</code> and <code>dir</code> in <code>Rglpk_solve_LP()</code>
<code>g.rhs</code>	Vector of numeric values for the right-hand sides of the constraints. to be used as <code>const.rhs</code> in <code>lp()</code> and <code>rhs</code> in <code>Rglpk_solve_LP()</code>
<code>w00</code>	Specified initial design proportion; default to be NULL, this will generate a random initial design
<code>link</code>	Link function of MLM, default to be "cumulative", options from "continuation", "cumulative", "adjacent", and "baseline"
<code>Fi.func</code>	A function for calculating Fisher information at a specific design point, default to be <code>Fi_func_MLM</code> function in the package
<code>reltol</code>	The relative convergence tolerance, default value 1e-5
<code>maxit</code>	The maximum number of iterations, default value 500
<code>delta</code>	A very small number, used in <code>alpha_star</code> calculation, default to be 1e-6.
<code>epsilon</code>	A very small number, for comparison of >0, <0, ==0, to reduce errors, default 1e-12



random	TRUE or FALSE, if TRUE then the function will run with additional "nram" number of random initial points, default to be TRUE
nram	When random == TRUE, the function will generate nram number of initial points, default is 3

### Value

w is the approximate D-optimal design

w0 is the initial design used to get optimal design w

Maximum is the maximized |F| value

itmax is the number of iterations

convergence is TRUE or FALSE, if TRUE means the reported design is converged

deriv.ans is the derivative from step 6 of constrained lift-one algorithm

gmax is the maximum g function in step 8 of constrained lift-one algorithm

reason is the lift-one loops break reason, either "all derivatives <=0" or "gmax <=0"

### Examples

```
#Example 8 of Trauma data example in Yifei, H., Liping, T., Yang, J. (2025)
#Constrained D-optimal design for paid research study
```

```
J = 5 # number of categories, >= 3
p = 12 # number of parameters
m = 8 # number of design points
nsample=600 #collect 600 samples finally from the 802 subjects
lower.bound <- function(i, w0){
  n = 600
  constraint = c(392,410)
  if(i <= 4){
    a.lower <- (sum(w0[5:8])-(constraint[2]/n)*(1-w0[i]))/(sum(w0[5:8]))
  }
  else{
    a.lower <- (sum(w0[1:4])-(constraint[1]/n)*(1-w0[i]))/(sum(w0[1:4]))
  }
  a.lower
}
upper.bound <- function(i, w0){
  n = 600
  constraint = c(392,410)
  if(i <= 4){
    b.upper <- ((constraint[1]/n)*(1-w0[i]) - (sum(w0[1:4])-w0[i]))/(1-sum(w0[1:4]))
  }
  else{
    b.upper <- ((constraint[2]/n)*(1-w0[i]) - (sum(w0[5:8])-w0[i]))/(1-sum(w0[5:8]))
  }
  b.upper
}
```

```

constraint = c(392,410)
g.con = matrix(0,nrow=length(constraint)+1+m, ncol=m)
g.con[2:3,] = matrix(data=c(1,1,1,1,0,0,0,0,0,0,0,0,1,1,1,1), ncol = m, byrow=TRUE)
g.con[1,] = rep(1, m)
g.con[4:(length(constraint)+1+m), ] = diag(1, nrow=m)
g.dir = c("==", "<=", "<=", rep(">=",m))
g.rhs = c(1, ifelse((constraint/nsample<1),constraint/nsample,1), rep(0, m))
Xi=rep(0,J*p*m)
dim(Xi)=c(J,p,m)
Xi[,1] = rbind(c( 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
Xi[,2] = rbind(c( 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
Xi[,3] = rbind(c( 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
Xi[,4] = rbind(c( 1, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 4, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
Xi[,5] = rbind(c( 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
Xi[,6] = rbind(c( 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
Xi[,7] = rbind(c( 1, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
Xi[,8] = rbind(c( 1, 4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 1, 4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 1, 4, 1, 0, 0, 0, 0, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 1, 0, 0, 0),
               c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))
thetavec = c(-4.3050, -0.0744, 4.3053, -2.3334, -0.3290, 3.4773,
             -0.1675, -0.3609, 2.7358, 1.2935, -0.1612, 1.4899)
set.seed(123)
liftone_constrained_MLM(m=m, p=p, Xi=Xi, J=J, beta=thetavec, lower.bound=lower.bound,

```

```
upper.bound=upper.bound, g.con=g.con,g.dir=g.dir, g.rhs=g.rhs, w00=NULL,
link='cumulative', Fi.func = Fi_func_MLM, reltol=1e-5, maxit=500,
delta = 1e-6, epsilon=1e-8, random=TRUE, nram=1)
```

---

liftone_GLM	<i>Unconstrained lift-one algorithm to find D-optimal allocations for GLM</i>
-------------	---

---

### Description

Unconstrained lift-one algorithm to find D-optimal allocations for GLM

### Usage

```
liftone_GLM(
  X,
  W,
  reltol = 1e-05,
  maxit = 500,
  random = TRUE,
  nram = 3,
  w00 = NULL
)
```

### Arguments

X	Model matrix, with nrow = num of design points and ncol = num of parameters
W	Diagonal of W matrix in Fisher information matrix, can be calculated from W_func_GLM in package
reltol	The relative convergence tolerance, default value 1e-5
maxit	The maximum number of iterations, default value 500
random	TRUE or FALSE, if TRUE then the function will run with additional "nram" number of random initial points, default to be TRUE
nram	When random == TRUE, the function will generate nram number of initial points, default is 3
w00	Specified initial design proportion; default to be NULL, this will generate a random initial design

### Value

w is the approximate D-optimal design

w0 is the initial design used to get optimal design w

Maximum is the maximized |F| value

itmax is the number of iterations

convergence is TRUE or FALSE, if TRUE means the reported design is converged

**Examples**

```

beta = c(0.5, 0.5, 0.5)
X = matrix(data=c(1,-1,-1,1,-1,1,1,1,-1), byrow=TRUE, nrow=3)
W_matrix = W_func_GLM(X=X, beta=beta)
w00 = c(1/6, 1/6, 2/3)
approximate_design = liftone_GLM(X=X, W=W_matrix, reltol=1e-10, maxit=100,
random=FALSE, nram=3, w00=w00)

```

---

liftone_MLM	<i>Unconstrained lift-one algorithm to find D-optimal allocations for MLM</i>
-------------	---

---

**Description**

Unconstrained lift-one algorithm to find D-optimal allocations for MLM

**Usage**

```

liftone_MLM(
  m,
  p,
  Xi,
  J,
  beta,
  link = "continuation",
  Fi.func = Fi_func_MLM,
  reltol = 1e-05,
  maxit = 500,
  w00 = NULL,
  random = TRUE,
  nram = 3
)

```

**Arguments**

m	The number of design points; it is usually the number of combinations of all the stratification factors
p	The number of parameters in the MLM model
Xi	Model matrix, a J by p by m 3D array of predictors for separate response category at all design points(input to determine ppo,npo,po)
J	The number of response levels
beta	A p*1 vector, parameter coefficients for MLM, the order of beta should be consistent with Xi
link	Link function of MLM, default to be "cumulative", options from "continuation", "cumulative", "adjacent", and "baseline"

Fi.func	A function for calculating Fisher information at a specific design point, default to be Fi_func_MLM function in the package
reltol	The relative convergence tolerance, default value 1e-5
maxit	The maximum number of iterations, default value 500
w00	Specified initial design proportion; default to be NULL, this will generate a random initial design
random	TRUE or FALSE, if TRUE then the function will run with additional "nram" number of random initial points, default to be TRUE
nram	When random == TRUE, the function will generate nram number of initial points, default is 3

### Value

w is the approximate D-optimal design

w0 is the initial design used to get optimal design

Maximum is the maximized |F| value

itmax is the number of iterations

convergence is TRUE or FALSE, if TRUE means the reported design is converged

### Examples

```
J = 5 # number of categories, >= 3
p = 12 # number of parameters
m = 8 # number of design points
Xi=rep(0,J*p*m) #J*p*m=5*12*8
dim(Xi)=c(J,p,m)
#design matrix
Xi[, ,1] = rbind(c( 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0),
                c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[, ,2] = rbind(c( 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 0),
                c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[, ,3] = rbind(c( 1, 3, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 1, 3, 0, 0, 0, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 0, 0, 0, 1, 3, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 0),
                c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[, ,4] = rbind(c( 1, 4, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 1, 4, 0, 0, 0, 0, 0, 0, 0, 0),
                c( 0, 0, 0, 0, 0, 0, 1, 4, 0, 0, 0, 0, 0),
```

```

c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 0),
c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[,5] = rbind(c( 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
c( 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0),
c( 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0),
c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1),
c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[,6] = rbind(c( 1, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
c( 0, 0, 0, 1, 2, 1, 0, 0, 0, 0, 0, 0, 0),
c( 0, 0, 0, 0, 0, 0, 1, 2, 1, 0, 0, 0, 0),
c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1),
c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[,7] = rbind(c( 1, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
c( 0, 0, 0, 1, 3, 1, 0, 0, 0, 0, 0, 0, 0),
c( 0, 0, 0, 0, 0, 0, 1, 3, 1, 0, 0, 0, 0),
c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 3, 1),
c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

Xi[,8] = rbind(c( 1, 4, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),
c( 0, 0, 0, 1, 4, 1, 0, 0, 0, 0, 0, 0, 0),
c( 0, 0, 0, 0, 0, 0, 1, 4, 1, 0, 0, 0, 0),
c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 4, 1),
c( 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0))

thetavec = c(-4.047, -0.131, 4.214, -2.225, -0.376, 3.519,
-0.302, -0.237, 2.420, 1.386, -0.120, 1.284)

liftone_MLM(m=m, p=p, Xi=Xi, J=J, beta=thetavec, link = "cumulative",
Fi.func=Fi_func_MLM, reltol=1e-5, maxit=5000, w00=NULL, random=TRUE, nram=3)

```

---

`print.list_output`      *Print Method for list\_output Objects*

---

### Description

Custom print method for objects of class 'list\_output'.

### Usage

```
## S3 method for class 'list_output'
print(x, ...)
```

### Arguments

`x`                    An object of class 'list\_output'.

`...`                  Additional arguments (ignored).

**Value**

Invisibly returns 'x' (the input object).

---

`print.matrix_list`      *Print Method for matrix\_list Objects*

---

**Description**

Custom print method for objects of class 'matrix\_list'.

**Usage**

```
## S3 method for class 'matrix_list'  
print(x, ...)
```

**Arguments**

x                    An object of class 'matrix\_list'.  
...                   Additional arguments (ignored).

**Value**

Invisibly returns 'x' (the input object).

---

`print.matrix_output`      *Print Method for matrix\_output Objects*

---

**Description**

Custom print method for objects of class 'matrix\_output'.

**Usage**

```
## S3 method for class 'matrix_output'  
print(x, ...)
```

**Arguments**

x                    An object of class 'matrix\_output'.  
...                   Additional arguments (ignored).

**Value**

Invisibly returns 'x' (the input object).

---

 trauma\_data

*Trauma data with multinomial response*


---

### Description

The data frame saves data from the trauma trial data from Chuang-Stein and Agresti (1997).

### Usage

```
trauma_data
```

### Format

A data frame with 802 rows and 5 variables:

**Severity** severity of the trauma symptoms, mild or moderate/severe

**Dose** dose levels applied to the patients, 4 levels, placebo, low, medium and high

**Label** stratification group in terms of severity and dose

**Outcome** treatment outcome, 5 levels, death, vegetative state, major disability, minor disability and good recovery

**ID** patient ID, 1-802

### Source

Chuang-Stein and Agresti (1997)

### Examples

```
data(trauma_data) #lazy loading
```

---

trial\_data

*Generated clinical trial data with binary response*


---

### Description

Generated with logistic regression model:

$$\text{logit}(P(Y_{ij} = 1 | \text{gender}_i, \text{age}_{i1}, \text{age}_{i2})) = 3 * \text{gender}_i + 3 * \text{age}_{i1} + 3 * \text{age}_{i2}$$

The data frame can be used to run GLM clinical trial example in Huang, Tong, Yang (2023)

### Usage

```
trial_data
```



**Format**

A data frame with 500 rows and 6 variables:

**gender** gender of the patients

**age\_1** 1 or 0, whether or not the patient belongs to 18-25 age group

**age\_2** 1 or 0, whether or not the patient belongs to 26-64 age group

**label** stratification group in terms of gender and age, 1 to 6

**Y** treatment effective or not, Y=1 means treatment is effective to the patient

**ID** patient ID, 1-500

**Source**

Generated pseudo clinical trial data to serve as an example.

**Examples**

```
data(trial_data) #lazy loading
```

---

W\_func\_GLM

*Calculate the diagonal elements nu of Fisher information matrix*

---

**Description**

Calculate the diagonal elements nu of Fisher information matrix

**Usage**

```
W_func_GLM(X, beta, link = "logit")
```

**Arguments**

X	Model matrix
beta	Parameters of GLM model
link	GLM link function, default is "logit", options are "logit", "probit", "cloglog", "loglog", "identity", identity is the same as ordinary linear regression

**Value**

the diagonal element nu of GLM Fisher information matrix, can be used as w in `liftone_constrained_GLM`

**Examples**

```
beta = c(0, 3, 3, 3) #main effect model beta_0, beta_1, beta_21, beta_22
#gives the 6 categories (0,0,0), (0,1,0),(0,0,1),(1,0,0),(1,1,0),(1,0,1)
X.liftone=matrix(data=c(1,0,0,0,1,0,1,0,1,0,0,1,1,1,0,0,1,1,1,0,1,1,0,1,1,0,1), ncol=4, byrow=TRUE)
#calculate diagonal elements of W based on beta's under logit link
W=W_func_GLM(X= X.liftone, beta=beta, link="logit")
```

# Index

## \* datasets

trauma\_data, [24](#)

trial\_data, [24](#)

approxtoexact\_constrained\_func, [2](#)

approxtoexact\_func, [3](#)

bounded\_uniform, [4](#)

F\_func\_GLM, [9](#)

F\_func\_MLM, [9](#)

Fdet\_func\_GLM, [5](#)

Fdet\_func\_MLM, [5](#)

Fdet\_func\_unif, [7](#)

Fi\_func\_MLM, [8](#)

iset\_func\_trauma, [11](#)

iset\_func\_trial, [12](#)

liftone\_constrained\_GLM, [13](#)

liftone\_constrained\_MLM, [15](#)

liftone\_GLM, [19](#)

liftone\_MLM, [20](#)

print.list\_output, [22](#)

print.matrix\_list, [23](#)

print.matrix\_output, [23](#)

trauma\_data, [24](#)

trial\_data, [24](#)

W\_func\_GLM, [25](#)