

# Package ‘KBoost’

December 19, 2024

**Type** Package

**Title** Inference of gene regulatory networks from gene expression data

**Version** 1.15.0

**Description** Reconstructing gene regulatory networks and transcription factor activity is crucial to understand biological processes and holds potential for developing personalized treatment. Yet, it is still an open problem as state-of-art algorithm are often not able to handle large amounts of data. Furthermore, many of the present methods predict numerous false positives and are unable to integrate other sources of information such as previously known interactions. Here we introduce KBoost, an algorithm that uses kernel PCA regression, boosting and Bayesian model averaging for fast and accurate reconstruction of gene regulatory networks. KBoost can also use a prior network built on previously known transcription factor targets. We have benchmarked KBoost using three different datasets against other high performing algorithms. The results show that our method compares favourably to other methods across datasets.

**Depends** R (>= 4.1), stats, utils

**License** GPL-2 | GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.1.1

**URL** <https://github.com/Luisiglm/KBoost>

**biocViews** Network, GraphAndNetwork, Bayesian, NetworkInference, GeneRegulation, Transcriptomics, SystemsBiology, Transcription, GeneExpression, Regression, PrincipalComponent

**Suggests** knitr, rmarkdown, testthat

**VignetteBuilder** knitr

**git\_url** <https://git.bioconductor.org/packages/KBoost>

**git\_branch** devel

**git\_last\_commit** b4d1509

**git\_last\_commit\_date** 2024-10-29

**Repository** Bioconductor 3.21

**Date/Publication** 2024-12-18

**Author** Luis F. Iglesias-Martinez [aut, cre] (ORCID:  
<https://orcid.org/0000-0002-9110-2189>),  
 Barbara de Kegel [aut],  
 Walter Kolch [aut]

**Maintainer** Luis F. Iglesias-Martinez <luis.iglesiasmartinez@ucd.ie>

## Contents

add_names . . . . .	3
AUPR_AUROC_matrix . . . . .	3
d4_mfac . . . . .	4
D4_multi_1 . . . . .	5
D4_multi_2 . . . . .	5
D4_multi_3 . . . . .	6
D4_multi_4 . . . . .	7
D4_multi_5 . . . . .	7
Gerstein_Prior_ENET_2 . . . . .	8
get_prior_Gerstein . . . . .	8
get_tfs_human . . . . .	9
grid_search_kboost . . . . .	10
G_D4_multi_1 . . . . .	10
G_D4_multi_2 . . . . .	11
G_D4_multi_3 . . . . .	12
G_D4_multi_4 . . . . .	12
G_D4_multi_5 . . . . .	13
Human_TFs . . . . .	14
irma_check . . . . .	14
IRMA_Gold . . . . .	15
irma_off . . . . .	15
irma_on . . . . .	16
kboost . . . . .	17
KBoost_human_symbol . . . . .	17
kernel_normal . . . . .	18
kernel_pc_boosting . . . . .	19
KPC . . . . .	19
net_dist_bin . . . . .	20
net_refine . . . . .	21
net_summary_bin . . . . .	21
RBF_K . . . . .	22
tab_2_matrix_D4 . . . . .	23
write_GRN_D4 . . . . .	23

**Index**

**25**

---

add_names	<i>Function to add names to network for the user.</i>
-----------	---

---

**Description**

Function to add names to network for the user.

**Usage**

```
add_names(grn, gen_names)
```

**Arguments**

grn	a GRN object from KBoost.
gen_names	a vector with the gene names.

**Value**

grn a GRN object with elements with user-defined gene names.

**Examples**

```
data(D4_multi_1)
Net = kboost(D4_multi_1)
g_names = matrix("G",100,1)
for (i in seq_along(g_names)){
  g_names[i] = paste(g_names[i],toString(i), sep = "")
}
Net = add_names(Net,g_names)
```

---

AUPR_AUROC_matrix	<i>Function to calculate the AUROC and AUPR of a known network. This function was made to test the R implementation of the KBoost Package.</i>
-------------------	--

---

**Description**

Function to calculate the AUROC and AUPR of a known network. This function was made to test the R implementation of the KBoost Package.

**Usage**

```
AUPR_AUROC_matrix(Net, G_mat, auto_remove, TFs, upper_limit)
```

**Arguments**

Net	An inferred gene regulatory network
G_mat	A matrix with the gold standard network.
auto_remove	TRUE if the auto-regulation is to be discarded.
TFs	the indexes of the rows of Net that are TFs.
upper_limit	Top number of edges to use.

**Value**

list object with AUPR and AUROC of gold standard in matrix format.

**Examples**

```
data(D4_multi_1)
Net = kboost(D4_multi_1)
g_mat1 = tab_2_matrix_D4(KBoost::G_D4_multi_1,100)
aupr_auroc = AUPR_AUROC_matrix(Net$GRN,g_mat1,auto_remove = TRUE, seq_len(100))
```

---

d4_mfac	<i>Function to obtain the AUPR and AUROC in the DREAM4 Multifactorial Challenge.</i>
---------	--

---

**Description**

Function to obtain the AUPR and AUROC in the DREAM4 Multifactorial Challenge.

**Usage**

```
d4_mfac(v, g, ite, write_res)
```

**Arguments**

v	a number between 0 and 1 that is the shrinkage parameter
g	a number larger than 0, width parameter for the RBF Kernel
ite	an integer with number of iterations.
write_res	a logical to indicate if the tables should be written.

**Value**

list with auroc and auprs of the DREAM4 multifactorial challenge.

**Examples**

```
res = d4_mfac()
```

---

`D4_multi_1`*Dream 4 multifactorial pertubation challenge dataset 1*

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**`D4_multi_1`**Format**`matrix`**Source**

<https://www.synapse.org/#!Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**`data(D4_multi_1)`

---

`D4_multi_2`*Dream 4 multifactorial pertubation challenge dataset 2*

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**`D4_multi_2`**Format**`matrix`**Source**

<https://www.synapse.org/#!Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**

```
data(D4_multi_2)
```

---

D4\_multi\_3

*Dream 4 multifactorial perturbation challenge dataset 3*

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**

```
D4_multi_3
```

**Format**

```
matrix
```

**Source**

<https://www.synapse.org/#!/Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**

```
data(D4_multi_3)
```

---

`D4_multi_4`*Dream 4 multifactorial pertubation challenge dataset 4*

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**

```
D4_multi_4
```

**Format**

```
matrix
```

**Source**

<https://www.synapse.org/#!Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**

```
data(D4_multi_4)
```

---

`D4_multi_5`*Dream 4 multifactorial pertubation challenge dataset 5*

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**

```
D4_multi_5
```

**Format**

```
matrix
```

**Source**

<https://www.synapse.org/#!Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**

```
data(D4_multi_5)
```

---

Gerstein\_Prior\_ENET\_2 *Gene Regulatory Network from human ChIP-Seq data in Encode*

---

**Description**

A gene regulatory network inferred from the ChIP-Seq Encode dataset Table with two columns  
The first column is a transcription factor and the second is a gene.

**Usage**

```
Gerstein_Prior_ENET_2
```

**Format**

```
matrix
```

**References**

Gerstein, M.B., et al. Architecture of the human regulatory network derived from ENCODE data. Nature 2012;489(7414):91-100.

**Examples**

```
data(Gerstein_Prior_ENET_2)
```

---

get\_prior\_Gerstein *Function to build a prior using a previously built Network on ChIP-Seq.*

---

**Description**

Function to build a prior using a previously built Network on ChIP-Seq.

**Usage**

```
get_prior_Gerstein(gen_names, TFs, pos_weight, neg_weight)
```



**Arguments**

gen_names	the gene names in Symbol nomenclature.
Tfs	the indexes of gene names which are TFs.
pos_weight	the prior weight for edges previously found in Gerstein 2011
neg_weight	the prior weight for edges not found in the Gerstein 2011/

**Value**

matrix with prior probabilities of the Tf target edges.

**Examples**

```
gen_names = c("TP53", "MDM2", "FOXO1", "ESR1", "CTCF", "YY1")
tfs = get_tfs_human(gen_names)
prior = get_prior_Gerstein(gen_names, tfs, 0.6, 0.4)
```

---

get\_tfs\_human      *Function to automatically assign Human TFs given a list of Symbols.*

---

**Description**

Function to automatically assign Human TFs given a list of Symbols.

**Usage**

```
get_tfs_human(gen_names)
```

**Arguments**

gen\_names      a vector or matrix with the Symbol Gene Names of the system.

**Value**

indexes of gen\_names who are TFs.

**Examples**

```
gen_names = c("TP53", "MDM2", "FOXO1", "ESR1", "CTCF")
tfs = get_tfs_human(gen_names)
```

---

`grid_search_kboost`      *Function to perform a grid search and find the hyperparameters.*

---

**Description**

Function to perform a grid search and find the hyperparameters.

**Usage**

```
grid_search_kboost(dataset, vs, gs, ite)
```

**Arguments**

<code>dataset</code>	1 for IRMA or 2 for DREAM4 multifactorial.
<code>vs</code>	The range of values of v. All values need to be between 0 and 1.
<code>gs</code>	The range of values of g. All values need to be larger than 0.
<code>ite</code>	An integer that is the number of iterations, fixed in this case.

**Value**

list with auprs and aurops of different values of vs and gs and ite.

**Examples**

```
res = grid_search_kboost(1,c(0.1,0.5,1),c(1,10,60,100),3)
```

---

`G_D4_multi_1`      *Gold Standard Dream 4 multifactorial pertubation challenge dataset 1*

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**

```
G_D4_multi_1
```

**Format**

matrix

**Source**

<https://www.synapse.org/#!Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**

```
data(G_D4_multi_1)
```

---

G_D4_multi_2	<i>Gold Standard Dream 4 multifactorial pertubation challenge dataset</i>
	2

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**

```
G_D4_multi_2
```

**Format**

```
matrix
```

**Source**

<https://www.synapse.org/#!Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**

```
data(G_D4_multi_2)
```

---

G_D4_multi_3	<i>Gold Standard Dream 4 multifactorial pertubation challenge dataset</i> 3
--------------	--

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**

```
G_D4_multi_3
```

**Format**

```
matrix
```

**Source**

<https://www.synapse.org/#!Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**

```
data(G_D4_multi_3)
```

---

G_D4_multi_4	<i>Gold Standard Dream 4 multifactorial pertubation challenge dataset</i> 4
--------------	--

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**

```
G_D4_multi_4
```

**Format**

```
matrix
```

**Source**

<https://www.synapse.org/#!Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**

```
data(G_D4_multi_4)
```

---

G_D4_multi_5	<i>Gold Standard Dream 4 multifactorial pertubation challenge dataset</i>
	5

---

**Description**

Each column is a gene and each row is a simulated experiment.

**Usage**

```
G_D4_multi_5
```

**Format**

```
matrix
```

**Source**

<https://www.synapse.org/#!Synapse:syn3049712/wiki/74628>

**References**

Marbach D, Prill RJ, Schaffter T, Mattiussi C, Floreano D, and Stolovitzky G. Revealing strengths and weaknesses of methods for gene network inference. PNAS, 107(14):6286-6291, 2010. Pubmed

**Examples**

```
data(G_D4_multi_5)
```

---

Human_TFs	<i>Index of human genes' Symbols, Entrez and Ensembl for K-Boost Pacakge that correspond to transcription factors.</i>
-----------	--

---

**Description**

Table with three columns corresponding to Symbol

**Usage**

Human\_TFs

**Format**

matrix

**References**

Lambert, S.A., et al. The Human Transcription Factors. Cell 2018;172(4):650-665.

**Examples**

```
data(Human_TFs)
```

---

irma_check	<i>Function to produce the AUPR and AUROC Results on the IRMA datasets.</i>
------------	---

---

**Description**

Function to produce the AUPR and AUROC Results on the IRMA datasets.

**Usage**

```
irma_check(v, g, ite)
```

**Arguments**

v	a number between 0 and 1 that is the shrinkage parameter
g	a number larger than 0 that is the width parameter for the RBF Kernel
ite	an integer with number of iterations.

**Value**

list with aurocs and auprs for IRMA datasets

**Examples**

```
res = irma_check()
```

---

IRMA\_Gold

*IRMA Gold Standard Network*

---

**Description**

Matrix where the rows are genes and columns are transcription factor.

**Usage**

```
IRMA_Gold
```

**Format**

```
matrix
```

**Source**

<https://www.sciencedirect.com/science/article/pii/S0092867409001561>

**References**

Cantone, I., et al. A Yeast Synthetic Network for In Vivo Assessment of Reverse-Engineering and Modeling Approaches. *Cell* 2009;137(1):172-181.

**Examples**

```
data(IRMA_Gold)
```

---

irma\_off

*IRMA Off Dataset*

---

**Description**

Matrix where the rows are experiments and columns are genes for the IRMA Off dataset.

**Usage**

```
irma_off
```

**Format**

```
matrix
```

**Source**

<https://www.sciencedirect.com/science/article/pii/S0092867409001561>

**References**

Cantone, I., et al. A Yeast Synthetic Network for In Vivo Assessment of Reverse-Engineering and Modeling Approaches. *Cell* 2009;137(1):172-181.

**Examples**

```
data(irma_off)
```

---

irma\_on

*IRMA On Dataset*

---

**Description**

Matrix where the rows are experiments and columns are genes for the IRMA Off dataset.

**Usage**

```
irma_on
```

**Format**

```
matrix
```

**Source**

<https://www.sciencedirect.com/science/article/pii/S0092867409001561>

**References**

Cantone, I., et al. A Yeast Synthetic Network for In Vivo Assessment of Reverse-Engineering and Modeling Approaches. *Cell* 2009;137(1):172-181.

**Examples**

```
data(irma_on)
```



---

kboost *A function to run KBoost.*

---

### Description

A function to run KBoost.

### Usage

```
kboost(X, TFs, g, v, prior_weights, ite)
```

### Arguments

X	an NxG matrix with the expression values of G genes and N obsv..
TFs	a Kx1 numeric matrix with integers of columns of X that are TFs.
g	a positive no., width parameter for RBF kernel. (default g = 40)
v	a no. between 0 and 1 with the shrinkage parameter. (default v =0.1)
prior_weights	it can be a scalar or GxK. (default 0.5)
ite	an integer for the maximum number of iterations (default 3)

### Value

a list with the results for kboost, with fields: GRN a matrix with the posterior edge probability after network refinement. GRN\_UP a matrix with the posterior edges before refinement. model a matrix with logical values for the TFs selected for each gene. g the width parameter for the RBF kernel. v the shrinkage parameter. prior the prior of each model. TFs a matrix with integers of each gene that is a TF. prior\_weights the prior\_weights with which KBoost was run. run\_time a scalar with the running time.

### Examples

```
data(D4_multi_1)
Net <- kboost(D4_multi_1)
```

---

KBoost\_human\_symbol *Function for KBoost on data from a human sample annotated with Symbol names.*

---

### Description

Function for KBoost on data from a human sample annotated with Symbol names.

**Usage**

```
KBoost_human_symbol(X, gen_names, g, v, ite, pos_weight, neg_weight)
```

**Arguments**

`X` an NxG matrix with the expression values of G genes and N samples.  
`gen_names` SYMBOL gene names corresponding to the columns of X.  
`g` a positive no., width parameter for the RBF kernel. (default `g = 40`)  
`v` a double between 0 and 1, the shrinkage parameter. (default `v = 0.1`)  
`ite` an integer with the number of iterations (default `ite = 3`)  
`pos_weight` no. between 0 and 1. Prior that a TF regulate a gene.  
`neg_weight` no. between 0 and 1, for TF gene pairs not seen before.

**Value**

list with results of KBoost on a dataset with Symbol gene names.

**Examples**

```
X = rnorm(50,0,1)
X = matrix(X,10,5)
gen_names = c("TP53", "YY1", "CTCF", "MDM2", "ESR1")
grn = KBoost_human_symbol(X,gen_names,pos_weight = 0.6, neg_weight =0.4)
```

---

<code>kernel_normal</code>	<i>A function to perform feature normalization in kernel space.</i>
----------------------------	---

---

**Description**

A function to perform feature normalization in kernel space.

**Usage**

```
kernel_normal(K)
```

**Arguments**

`K` an NxN numeric matrix with the kernel function with N observations.

**Value**

feature centred kernel.

**Examples**

```
x = rnorm(100,0,1)
k = RBF_K(x,40)
k_ = kernel_normal(k)
```

---

kernel\_pc\_boosting      *Function to perform Kernel Principal Component Boosting*

---

**Description**

Function to perform Kernel Principal Component Boosting

**Usage**

```
kernel_pc_boosting(X, Y, g, v, ite, thr)
```

**Arguments**

X	A matrix with the explanatory variables.
Y	a matrix with the variable to predict.
g	a positive number with the width parameter for the RBF Kernel.
v	a number between 0 and 1 that corresponds to the shrinkage parameter.
ite	an integer with the number of iterations.
thr	a threshold to discard Kernel principal components whose eigenvalue

**Value**

function an sum of squared errors.

**Examples**

```
data(D4_multi_1)
Y = scale(matrix(D4_multi_1[,91],100,1))
X = scale(D4_multi_1[,-91])
res = kernel_pc_boosting(X,Y, g= 40, v = 0.5, ite = 3, thr = 1e-10)
```

---

KPC      *Function to calculate the principal components of a kernel.*

---

**Description**

Function to calculate the principal components of a kernel.

**Usage**

```
KPC(K, thr)
```

**Arguments**

**K** an NxN numeric matrix with the Kernel matrix.

**thr** a positive scalar which is a threshold to discard eigen-vectors based on eigen-values.

**Value**

the kernel principal components

**Examples**

```
x = rnorm(100,0,1)
k = RBF_K(x,1)
k_ = kernel_normal(k)
kpca = KPC(k,1e-8)
```

---

net\_dist\_bin                      *Function to calculate the distance between nodes.*

---

**Description**

Function to calculate the distance between nodes.

**Usage**

```
net_dist_bin(GRN, TFs, thr)
```

**Arguments**

**GRN** An inferred networks with the predictive probabilities that a transcription factor regulates a gene.

**TFs** A vector with indexes of the rows of GRN which correspond to TFs.

**thr** A scalar between 0 and 1 that is used select the edges with large posterior probabilities.

**Value**

a matrix with the distances between edges.

**Examples**

```
data(D4_multi_1)
Net = kboost(D4_multi_1)
dist = net_dist_bin(Net$GRN,Net$TFs,0.1)
```

---

net_refine	<i>Function to do a heuristic post-processing that improves accuracy. Each column is multiplied by its variance.</i>
------------	--

---

**Description**

Function to do a heuristic post-processing that improves accuracy. Each column is multiplied by its variance.

**Usage**

```
net_refine(Net)
```

**Arguments**

Net                    a GRN with TFs in the columns.

**Value**

the network with Slavek and Arodz heuristic

**Examples**

```
Net = rbeta(10000,1,2)
Net = matrix(Net,100,100)
net_ref = net_refine(Net)
```

---

net_summary_bin	<i>Function to summarize the GRN filtered with a threshold,</i>
-----------------	---

---

**Description**

Function to summarize the GRN filtered with a threshold,

**Usage**

```
net_summary_bin(GRN, TFs, thr, a, b)
```

**Arguments**

GRN                    An inferred network

TFs                    A vector with indexes of the rows of GRN which correspond to TFs.

thr                    a scalar between 0 and 1, a threshold for posterior probabilities.

a                      parameter for Katz and PageRank centrality (default the inverse of the largest eigenvalue of GRN).

b                      parameter for Katz and PageRank centrality (default b = 1).

**Value**

list with table version of the GRN, outdegree and indegree, and closeness centrality.

**Examples**

```
data(D4_multi_1)
Net = kboost(D4_multi_1)
Net_Summary = net_summary_bin(Net$GRN)
```

---

RBF\_K

*Function to calculate the RBF Kernel of a matrix X with width g.*

---

**Description**

Function to calculate the RBF Kernel of a matrix X with width g.

**Usage**

```
RBF_K(x, g)
```

**Arguments**

x                    an Nx1 numeric matrix with N observations.  
g                    a positive scalar with the width parameter.

**Value**

the matrix with the RBF kernel

**Examples**

```
x = rnorm(100,0,1)
k = RBF_K(x,40)
```

---

tab_2_matrix_D4	<i>Function to produce the gold standard of the DREAM4 Multifactorial Challenge in matrix format.</i>
-----------------	---

---

**Description**

Function to produce the gold standard of the DREAM4 Multifactorial Challenge in matrix format.

**Usage**

```
tab_2_matrix_D4(g_table, G)
```

**Arguments**

g_table	the network in table format. The first column is the Tf, the second column the gene, and the third indicates if there is an interaction.
G	the number of genes.

**Value**

a network in table format transformed into a matrix.

**Examples**

```
g_table = KBoost::G_D4_multi_1
g_mat = tab_2_matrix_D4(g_table,100)
```

---

write_GRN_D4	<i>Function to write output in DREAM4 Challenge Format.</i>
--------------	---

---

**Description**

Function to write output in DREAM4 Challenge Format.

**Usage**

```
write_GRN_D4(GRN, TFs, filename)
```

**Arguments**

GRN	a GxK gene regulatory network.
TFs	a K set of indexes of G that are TFs.
filename	a string with the filename.

**Value**

a file with the network written as a file.

**Examples**

```
data(D4_multi_1)
Net = kboost(D4_multi_1)
write_GRN_D4(Net$GRN, seq_len(100), "D4_multi_1_network.txt")
```



# Index

## \* datasets

D4\_multi\_1, [5](#)  
D4\_multi\_2, [5](#)  
D4\_multi\_3, [6](#)  
D4\_multi\_4, [7](#)  
D4\_multi\_5, [7](#)  
G\_D4\_multi\_1, [10](#)  
G\_D4\_multi\_2, [11](#)  
G\_D4\_multi\_3, [12](#)  
G\_D4\_multi\_4, [12](#)  
G\_D4\_multi\_5, [13](#)  
Gerstein\_Prior\_ENET\_2, [8](#)  
Human\_TFs, [14](#)  
IRMA\_Gold, [15](#)  
irma\_off, [15](#)  
irma\_on, [16](#)

add\_names, [3](#)  
AUPR\_AUROC\_matrix, [3](#)

d4\_mfac, [4](#)  
D4\_multi\_1, [5](#)  
D4\_multi\_2, [5](#)  
D4\_multi\_3, [6](#)  
D4\_multi\_4, [7](#)  
D4\_multi\_5, [7](#)

G\_D4\_multi\_1, [10](#)  
G\_D4\_multi\_2, [11](#)  
G\_D4\_multi\_3, [12](#)  
G\_D4\_multi\_4, [12](#)  
G\_D4\_multi\_5, [13](#)  
Gerstein\_Prior\_ENET\_2, [8](#)  
get\_prior\_Gerstein, [8](#)  
get\_tfs\_human, [9](#)  
grid\_search\_kboost, [10](#)

Human\_TFs, [14](#)

irma\_check, [14](#)  
IRMA\_Gold, [15](#)  
irma\_off, [15](#)  
irma\_on, [16](#)  
kboost, [17](#)  
KBoost\_human\_symbol, [17](#)  
kernel\_normal, [18](#)  
kernel\_pc\_boosting, [19](#)  
KPC, [19](#)  
net\_dist\_bin, [20](#)  
net\_refine, [21](#)  
net\_summary\_bin, [21](#)  
RBF\_K, [22](#)  
tab\_2\_matrix\_D4, [23](#)  
write\_GRN\_D4, [23](#)