

# Package ‘inferCSN’

March 15, 2024

**Type** Package

**Title** Inferring Cell-Specific Gene Regulatory Network

**Version** 1.0.1

**Date** 2024-3-14

**Maintainer** Meng Xu <mengxu98@qq.com>

**Description**

A method for inferring cell-specific gene regulatory network from single-cell sequencing data.

**License** MIT + file LICENSE

**URL** <https://mengxu98.github.io/inferCSN/>

**BugReports** <https://github.com/mengxu98/inferCSN/issues>

**Depends** R (>= 3.3.0)

**Imports** ComplexHeatmap, doParallel, dplyr, foreach, ggnetwork, ggplot2, ggraph, Matrix, methods, parallel, patchwork, progress, purrr, Rcpp, stats, utils

**Suggests** circlize, gtools, igraph, precrec, pROC, testthat (>= 3.0.0), tidygraph

**LinkingTo** Rcpp, RcppArmadillo

**Config/Needs/website** mengxu98/mxtemplate

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.1

**Language** en-US

**NeedsCompilation** yes

**Author** Meng Xu [aut, cre] (<<https://orcid.org/0000-0002-8300-1054>>)

**Repository** CRAN

**Date/Publication** 2024-03-15 07:00:02 UTC

## R topics documented:

inferCSN-package . . . . .	2
acc.calculate . . . . .	3
auc.calculate . . . . .	3
calculate.gene.rank . . . . .	4
check.parameters . . . . .	5
coef.SRM_fit . . . . .	6
contrast.networks . . . . .	7
dynamic.networks . . . . .	7
example_ground_truth . . . . .	8
example_matrix . . . . .	9
filter.sort.matrix . . . . .	9
inferCSN . . . . .	10
model.fit . . . . .	12
net.format . . . . .	14
network.heatmap . . . . .	15
predict.SRM_fit . . . . .	17
prepare.performance.data . . . . .	18
print.SRM_fit . . . . .	18
single.network . . . . .	19
sparse.regression . . . . .	20
table.to.matrix . . . . .	22

<b>Index</b>	<b>23</b>
--------------	-----------

---

inferCSN-package	<i>inferCSN: Inferring Cell-Specific Gene Regulatory Network</i>
------------------	--

---

### Description

A method for inferring cell-specific gene regulatory network from single-cell sequencing data.

### Author(s)

Meng xu (Maintainer), <mengxu98@qq.com>

### Source

<https://github.com/mengxu98/inferCSN>

### See Also

Useful links:

- <https://mengxu98.github.io/inferCSN/>
- Report bugs at <https://github.com/mengxu98/inferCSN/issues>

---

acc.calculate	<i>ACC calculate</i>
---------------	----------------------

---

**Description**

ACC calculate

**Usage**

```
acc.calculate(weight_table, ground_truth)
```

**Arguments**

weight_table	The weight data table of network
ground_truth	Ground truth for calculate AUC

**Value**

ACC value

**Examples**

```
library(inferCSN)
data("example_matrix")
data("example_ground_truth")
weight_table <- inferCSN(example_matrix)
acc <- acc.calculate(weight_table, example_ground_truth)
acc
```

---

auc.calculate	<i>AUC value calculate</i>
---------------	----------------------------

---

**Description**

AUC value calculate

**Usage**

```
auc.calculate(
  weight_table,
  ground_truth,
  plot = FALSE,
  line_color = "#1563cc",
  line_width = 1
)
```

**Arguments**

weight_table	The weight data table of network
ground_truth	Ground truth for calculate AUC
plot	If true, draw and print figure of AUC
line_color	The color of line in the figure
line_width	The width of line in the figure

**Value**

AUC values and figure

**Examples**

```
library(inferCSN)
data("example_matrix")
data("example_ground_truth")
weight_table <- inferCSN(example_matrix)
auc <- auc.calculate(weight_table, example_ground_truth, plot = TRUE)
auc
```

---

calculate.gene.rank    *Calculate and rank TFs in network*

---

**Description**

Calculate and rank TFs in network

**Usage**

```
calculate.gene.rank(weight_table, directed = FALSE)
```

**Arguments**

weight_table	The weight data table of network
directed	If GRN is directed or not

**Value**

A data.table with three columns

**Examples**

```
library(inferCSN)
data("example_matrix")
weight_table <- inferCSN(example_matrix)
ranks <- calculate.gene.rank(weight_table)
head(ranks)
```

---

check.parameters	<i>Check input parameters</i>
------------------	-------------------------------

---

## Description

Check input parameters

## Usage

```
check.parameters(
  matrix,
  penalty,
  algorithm,
  cross_validation,
  seed,
  n_folds,
  k_folds,
  r_threshold,
  regulators,
  targets,
  regulators_num,
  verbose,
  cores
)
```

## Arguments

matrix	An expression matrix, cells by genes
penalty	The type of regularization. This can take either one of the following choices: "L0" and "L0L2". For high-dimensional and sparse data, such as single-cell sequencing data, "L0L2" is more effective.
algorithm	The type of algorithm used to minimize the objective function. Currently "CD" and "CDPSI" are supported. The CDPSI algorithm may yield better results, but it also increases running time.
cross_validation	Check whether cross validation is used.
seed	The seed used in randomly shuffling the data for cross-validation.
n_folds	The number of folds for cross-validation.
k_folds	The number of folds for sample split.
r_threshold	Threshold of $R^2$ .
regulators	Regulator genes.
targets	Target genes.

regulators_num	The number of non-zero coef, this value will affect the final performance. The maximum support size at which to terminate the regularization path. Recommend setting this to a small fraction of min(n,p) (e.g. $0.05 * \min(n,p)$ ) as L0 regularization typically selects a small portion of non-zeros.
verbose	Print detailed information.
cores	CPU cores.

**Value**

No return value, called for check input parameters

---

coef.SRM_fit	<i>Extracts a specific solution in the regularization path</i>
--------------	--

---

**Description**

Extracts a specific solution in the regularization path

**Usage**

```
## S3 method for class 'SRM_fit'
coef(object, lambda = NULL, gamma = NULL, supportSize = NULL, ...)

## S3 method for class 'SRM_fit_CV'
coef(object, lambda = NULL, gamma = NULL, ...)
```

**Arguments**

object	The output of model.fit or inferCSN.cvfit
lambda	The value of lambda at which to extract the solution
gamma	The value of gamma at which to extract the solution
supportSize	The number of non-zeros each solution extracted will contain
...	Other parameters

**Value**

Return the specific solution

Return the specific solution

---

contrast.networks      *contrast.networks*

---

### Description

contrast.networks

### Usage

```
contrast.networks(  
  weight_table,  
  degree_value = 0,  
  weight_value = 0,  
  legend_position = "bottom"  
)
```

### Arguments

weight\_table      The weight data table of network.  
degree\_value      degree\_value  
weight\_value      weight\_value  
legend\_position      The position of legend.

### Value

Return a ggplot2 object

### Examples

```
library(inferCSN)  
data("example_matrix")  
weight_table <- inferCSN(example_matrix)  
contrast.networks(weight_table[1:50, ])
```

---

dynamic.networks      *Plot of dynamic networks*

---

### Description

Plot of dynamic networks

**Usage**

```
dynamic.networks(  
  weight_table,  
  regulators = NULL,  
  targets = NULL,  
  legend_position = "right"  
)
```

**Arguments**

weight_table	The weight data table of network.
regulators	Regulators list.
targets	Targets list.
legend_position	The position of legend.

**Value**

A list of ggplot2 objects

**Examples**

```
library(inferCSN)  
data("example_matrix")  
weight_table <- inferCSN(example_matrix)  
dynamic.networks(  
  weight_table,  
  regulators = weight_table[1, 1]  
)  
dynamic.networks(  
  weight_table,  
  targets = weight_table[1, 1]  
)  
dynamic.networks(  
  weight_table,  
  regulators = weight_table[1, 1],  
  targets = weight_table[1, 2]  
)
```

---

example\_ground\_truth *Example ground truth data*

---

**Description**

The data used for calculate the evaluating indicator.



---

example_matrix	<i>Example matrix data</i>
----------------	----------------------------

---

**Description**

The matrix used for reconstruct gene regulatory network.

---

filter.sort.matrix	<i>Filter and sort matrix</i>
--------------------	-------------------------------

---

**Description**

Filter and sort matrix

**Usage**

```
filter.sort.matrix(weight_matrix, regulators = NULL, targets = NULL)
```

**Arguments**

weight_matrix	The matrix of network weight.
regulators	Regulators list.
targets	Targets list.

**Value**

Filtered and sorted matrix

**Examples**

```
library(inferCSN)
data("example_matrix")
weight_table <- inferCSN(example_matrix)
weight_matrix <- table.to.matrix(weight_table)
filter.sort.matrix(weight_matrix)[1:6, 1:6]

filter.sort.matrix(
  weight_matrix ,
  regulators = c("g1", "g2"),
  targets = c("g3", "g4")
)
```

**Description**

Inferring Cell-Specific Gene Regulatory Network

**Usage**

```
inferCSN(object, ...)  
  
## S4 method for signature 'matrix'  
inferCSN(  
  object,  
  penalty = "L0",  
  algorithm = "CD",  
  cross_validation = FALSE,  
  seed = 1,  
  n_folds = 10,  
  k_folds = NULL,  
  r_threshold = 0,  
  regulators = NULL,  
  targets = NULL,  
  regulators_num = NULL,  
  verbose = FALSE,  
  cores = 1,  
  ...  
)  
  
## S4 method for signature 'data.frame'  
inferCSN(  
  object,  
  penalty = "L0",  
  algorithm = "CD",  
  cross_validation = FALSE,  
  seed = 1,  
  n_folds = 10,  
  k_folds = NULL,  
  r_threshold = 0,  
  regulators = NULL,  
  targets = NULL,  
  regulators_num = NULL,  
  verbose = FALSE,  
  cores = 1,  
  ...  
)
```

**Arguments**

object	Input object
...	Arguments for other methods
penalty	The type of regularization. This can take either one of the following choices: "L0" and "L0L2". For high-dimensional and sparse data, such as single-cell sequencing data, "L0L2" is more effective.
algorithm	The type of algorithm used to minimize the objective function. Currently "CD" and "CDPSI" are supported. The CDPSI algorithm may yield better results, but it also increases running time.
cross_validation	Check whether cross validation is used.
seed	The seed used in randomly shuffling the data for cross-validation.
n_folds	The number of folds for cross-validation.
k_folds	The number of folds for sample split.
r_threshold	Threshold of $R^2$ .
regulators	Regulator genes.
targets	Target genes.
regulators_num	The number of non-zero coef, this value will affect the final performance. The maximum support size at which to terminate the regularization path. Recommend setting this to a small fraction of $\min(n,p)$ (e.g. $0.05 * \min(n,p)$ ) as L0 regularization typically selects a small portion of non-zeros.
verbose	Print detailed information.
cores	CPU cores.

**Value**

A data table of gene-gene regulatory relationship

**Examples**

```
library(inferCSN)
data("example_matrix")
weight_table <- inferCSN(example_matrix, verbose = TRUE)
head(weight_table)

weight_table <- inferCSN(example_matrix, verbose = TRUE, cores = 2)
head(weight_table)
```

---

`model.fit`*Fit a sparse regression model*

---

### Description

Computes the regularization path for the specified loss function and penalty function

### Usage

```
model.fit(  
  x,  
  y,  
  penalty = "L0",  
  algorithm = "CD",  
  regulators_num = NULL,  
  cross_validation = FALSE,  
  n_folds = 10,  
  seed = 1,  
  loss = "SquaredError",  
  nLambda = 100,  
  nGamma = 5,  
  gammaMax = 10,  
  gammaMin = 1e-04,  
  partialSort = TRUE,  
  maxIters = 200,  
  rtol = 1e-06,  
  atol = 1e-09,  
  activeSet = TRUE,  
  activeSetNum = 3,  
  maxSwaps = 100,  
  scaleDownFactor = 0.8,  
  screenSize = 1000,  
  autoLambda = NULL,  
  lambdaGrid = list(),  
  excludeFirstK = 0,  
  intercept = TRUE,  
  lows = -Inf,  
  highs = Inf  
)
```

### Arguments

<code>x</code>	The data matrix
<code>y</code>	The response vector
<code>penalty</code>	The type of regularization. This can take either one of the following choices: "L0" and "LOL2". For high-dimensional and sparse data, such as single-cell sequencing data, "LOL2" is more effective.

algorithm	The type of algorithm used to minimize the objective function. Currently "CD" and "CDPSI" are supported. The CDPSI algorithm may yield better results, but it also increases running time.
regulators_num	The number of non-zero coef, this value will affect the final performance. The maximum support size at which to terminate the regularization path. Recommend setting this to a small fraction of $\min(n,p)$ (e.g. $0.05 * \min(n,p)$ ) as L0 regularization typically selects a small portion of non-zeros.
cross_validation	Check whether cross validation is used.
n_folds	The number of folds for cross-validation.
seed	The seed used in randomly shuffling the data for cross-validation.
loss	The loss function
nLambda	The number of Lambda values to select
nGamma	The number of Gamma values to select
gammaMax	The maximum value of Gamma when using the L0L2 penalty
gammaMin	The minimum value of Gamma when using the L0L2 penalty
partialSort	If TRUE, partial sorting will be used for sorting the coordinates to do greedy cycling. Otherwise, full sorting is used
maxIters	The maximum number of iterations (full cycles) for CD per grid point
rtol	The relative tolerance which decides when to terminate optimization (based on the relative change in the objective between iterations)
atol	The absolute tolerance which decides when to terminate optimization (based on the absolute L2 norm of the residuals)
activeSet	If TRUE, performs active set updates
activeSetNum	The number of consecutive times a support should appear before declaring support stabilization
maxSwaps	The maximum number of swaps used by CDPSI for each grid point
scaleDownFactor	This parameter decides how close the selected Lambda values are
screenSize	The number of coordinates to cycle over when performing initial correlation screening
autoLambda	Ignored parameter. Kept for backwards compatibility
lambdaGrid	A grid of Lambda values to use in computing the regularization path
excludeFirstK	This parameter takes non-negative integers
intercept	If FALSE, no intercept term is included in the model
lows	Lower bounds for coefficients
highs	Upper bounds for coefficients

**Value**

An S3 object describing the regularization path

## Examples

```
library(inferCSN)
data("example_matrix")
fit <- model.fit(
  example_matrix[, -1],
  example_matrix[, 1]
)
head(coef(fit))
```

---

net.format

*Format weight table*

---

## Description

Format weight table

## Usage

```
net.format(weight_table, regulators = NULL, targets = NULL, abs_weight = TRUE)
```

## Arguments

weight_table	The weight data table of network.
regulators	Regulators list.
targets	Targets list.
abs_weight	Logical value, whether to perform absolute value on weights, default set to 'TRUE', and when set 'abs_weight' to 'TRUE', the output of weight table will create a new column named 'Interaction'.

## Value

Format weight table

## Examples

```
library(inferCSN)
data("example_matrix")
weight_table <- inferCSN(example_matrix)

net.format(
  weight_table,
  regulators = c("g1")
)
net.format(
  weight_table,
  regulators = c("g1"),
  abs_weight = FALSE
)
```

```
net.format(  
  weight_table,  
  targets = c("g3")  
)  
net.format(  
  weight_table,  
  regulators = c("g1", "g3"),  
  targets = c("g3", "g5")  
)
```

---

network.heatmap	<i>The heatmap of network</i>
-----------------	-------------------------------

---

### Description

The heatmap of network

### Usage

```
network.heatmap(  
  weight_table,  
  regulators = NULL,  
  targets = NULL,  
  switch_matrix = TRUE,  
  show_names = FALSE,  
  heatmap_size = 5,  
  heatmap_height = NULL,  
  heatmap_width = NULL,  
  heatmap_title = NULL,  
  heatmap_color = c("#1966ad", "white", "#bb141a"),  
  legend_name = "Weight",  
  row_title = "Regulators"  
)
```

### Arguments

weight_table	The weight data table of network.
regulators	Regulators list.
targets	Targets list.
switch_matrix	Logical value, whether to weight data table to matrix.
show_names	Logical value, whether to show names of row and column.
heatmap_size	The size of heatmap, default set to 5.
heatmap_height	The height of heatmap.
heatmap_width	The width of heatmap.

heatmap\_title The title of heatmap.  
 heatmap\_color Colors of heatmap.  
 legend\_name The name of legend.  
 row\_title The title of row.

### Value

Return a heatmap of ggplot2 object

### Examples

```
library(inferCSN)
data("example_matrix")
data("example_ground_truth")
weight_table <- inferCSN(example_matrix)

p1 <- network.heatmap(
  example_ground_truth[, 1:3],
  heatmap_title = "Ground truth",
  legend_name = "Ground truth"
)
p2 <- network.heatmap(
  weight_table,
  heatmap_title = "inferCSN",
  legend_name = "inferCSN"
)
ComplexHeatmap::draw(p1 + p2)

p3 <- network.heatmap(
  weight_table,
  heatmap_title = "inferCSN",
  legend_name = "Weight1",
  heatmap_color = c("#20a485", "#410054", "#fee81f")
)
p4 <- network.heatmap(
  weight_table,
  heatmap_title = "inferCSN",
  legend_name = "Weight2",
  heatmap_color = c("#20a485", "white", "#fee81f")
)
ComplexHeatmap::draw(p3 + p4)

network.heatmap(
  weight_table,
  heatmap_title = "inferCSN",
  show_names = TRUE
)

network.heatmap(
  weight_table,
  regulators = c("g1", "g2"),
```



```

heatmap_title = "inferCSN",
show_names = TRUE
)

network.heatmap(
weight_table,
targets = c("g1", "g2"),
heatmap_title = "inferCSN",
show_names = TRUE
)

network.heatmap(
weight_table,
regulators = c("g1", "g3", "g5"),
targets = c("g3", "g6", "g9"),
heatmap_title = "inferCSN",
show_names = TRUE
)

```

---

predict.SRM_fit	<i>Predict Response</i>
-----------------	-------------------------

---

## Description

Predicts response for a given sample

## Usage

```

## S3 method for class 'SRM_fit'
predict(object, newx, lambda = NULL, gamma = NULL, ...)

## S3 method for class 'SRM_fit_CV'
predict(object, newx, lambda = NULL, gamma = NULL, ...)

```

## Arguments

object	The output of model.fit
newx	A matrix on which predictions are made. The matrix should have p columns
lambda	The value of lambda to use for prediction. A summary of the lambdas in the regularization path can be obtained using <code>print(fit)</code>
gamma	The value of gamma to use for prediction. A summary of the gammas in the regularization path can be obtained using <code>print(fit)</code>
...	Other parameters

## Details

If both lambda and gamma are not supplied, then a matrix of predictions for all the solutions in the regularization path is returned. If lambda is supplied but gamma is not, the smallest value of gamma is used. In case of logistic regression, probability values are returned

**Value**

Return predict value  
 Return the predict value

---

```
prepare.performance.data
      prepare.performance.data
```

---

**Description**

prepare.performance.data

**Usage**

```
prepare.performance.data(weight_table, ground_truth)
```

**Arguments**

weight\_table    The weight data table of network  
 ground\_truth    Ground truth for calculate AUC

**Value**

Formatted data

---

```
print.SRM_fit            Prints a summary of model.fit
```

---

**Description**

Prints a summary of model.fit

**Usage**

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

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

**Arguments**

x                    The output of model.fit or inferCSN.cvfit  
 ...                  Other parameters

**Value**

Return information of model.fit

Return information of model.fit

---

single.network	<i>Construct network for single gene</i>
----------------	--

---

**Description**

Construct network for single gene

**Usage**

```
single.network(  
  matrix,  
  regulators,  
  target,  
  cross_validation = FALSE,  
  seed = 1,  
  penalty = "L0",  
  algorithm = "CD",  
  regulators_num = NULL,  
  n_folds = 10,  
  k_folds = NULL,  
  r_threshold = 0,  
  verbose = FALSE  
)
```

**Arguments**

matrix	An expression matrix, cells by genes.
regulators	Regulator genes.
target	Target genes.
cross_validation	Check whether cross validation is used.
seed	The seed used in randomly shuffling the data for cross-validation.
penalty	The type of regularization. This can take either one of the following choices: "L0" and "L0L2". For high-dimensional and sparse data, such as single-cell sequencing data, "L0L2" is more effective.
algorithm	The type of algorithm used to minimize the objective function. Currently "CD" and "CDPSI" are supported. The CDPSI algorithm may yield better results, but it also increases running time.

regulators_num	The number of non-zero coef, this value will affect the final performance. The maximum support size at which to terminate the regularization path. Recommend setting this to a small fraction of $\min(n,p)$ (e.g. $0.05 * \min(n,p)$ ) as L0 regularization typically selects a small portion of non-zeros.
n_folds	The number of folds for cross-validation.
k_folds	The number of folds for sample split.
r_threshold	Threshold of $R^2$ .
verbose	Print detailed information.

**Value**

The weight data table of sub-network

**Examples**

```
library(inferCSN)
data("example_matrix")
single_network <- single.network(
  example_matrix,
  regulators = colnames(example_matrix),
  target = "g1"
)
head(single_network)

single.network(
  example_matrix,
  regulators = "g1",
  target = "g2"
)
```

---

sparse.regression      *Sparse regression model*

---

**Description**

Sparse regression model

**Usage**

```
sparse.regression(
  x,
  y,
  cross_validation = FALSE,
  seed = 1,
  penalty = "L0",
  algorithm = "CD",
  regulators_num = NULL,
```

```

n_folds = 10,
k_folds = NULL,
r_threshold = 0,
verbose = FALSE
)

```

### Arguments

x	The data matrix
y	The response vector
cross_validation	Check whether cross validation is used.
seed	The seed used in randomly shuffling the data for cross-validation.
penalty	The type of regularization. This can take either one of the following choices: "L0" and "L0L2". For high-dimensional and sparse data, such as single-cell sequencing data, "L0L2" is more effective.
algorithm	The type of algorithm used to minimize the objective function. Currently "CD" and "CDPSI" are supported. The CDPSI algorithm may yield better results, but it also increases running time.
regulators_num	The number of non-zero coef, this value will affect the final performance. The maximum support size at which to terminate the regularization path. Recommend setting this to a small fraction of $\min(n,p)$ (e.g. $0.05 * \min(n,p)$ ) as L0 regularization typically selects a small portion of non-zeros.
n_folds	The number of folds for cross-validation.
k_folds	The number of folds for sample split.
r_threshold	Threshold of $R^2$ .
verbose	Print detailed information.

### Value

Coefficients

### Examples

```

library(inferCSN)
data("example_matrix")
coefficients <- sparse.regression(
  example_matrix[, -1],
  example_matrix[, 1]
)
coefficients

```

---

table.to.matrix	<i>Switch weight table to matrix</i>
-----------------	--------------------------------------

---

**Description**

Switch weight table to matrix

**Usage**

```
table.to.matrix(weight_table, regulators = NULL, targets = NULL)
```

**Arguments**

weight_table	The weight data table of network.
regulators	Regulators list.
targets	Targets list.

**Value**

Weight matrix

**Examples**

```
library(inferCSN)
data("example_matrix")
weight_table <- inferCSN(example_matrix)
head(weight_table)

table.to.matrix(weight_table)[1:6, 1:6]

table.to.matrix(
  weight_table,
  regulators = c("g1", "g2"),
  targets = c("g3", "g4")
)
```

# Index

`acc.calculate`, 3  
`auc.calculate`, 3

`calculate.gene.rank`, 4  
`check.parameters`, 5  
`coef.SRM_fit`, 6  
`coef.SRM_fit_CV (coef.SRM_fit)`, 6  
`contrast.networks`, 7

`dynamic.networks`, 7

`example_ground_truth`, 8  
`example_matrix`, 9

`filter.sort.matrix`, 9

`inferCSN`, 10  
`inferCSN, data.frame-method (inferCSN)`, 10  
`inferCSN, matrix-method (inferCSN)`, 10  
`inferCSN-package`, 2

`model.fit`, 12

`net.format`, 14  
`network.heatmap`, 15

`predict.SRM_fit`, 17  
`predict.SRM_fit_CV (predict.SRM_fit)`, 17  
`prepare.performance.data`, 18  
`print.SRM_fit`, 18  
`print.SRM_fit_CV (print.SRM_fit)`, 18

`single.network`, 19  
`sparse.regression`, 20

`table.to.matrix`, 22