Package 'gemini'

May 1, 2024

Type Package

Title GEMINI: Variational inference approach to infer genetic interactions from pairwise CRISPR screens

Version 1.18.0

Description GEMINI uses log-fold changes to model sample-dependent and independent effects, and uses a variational Bayes approach to infer these effects. The inferred effects are used to score and identify genetic interactions, such as lethality and recovery. More details can be found in Zamanighomi et al. 2019 (in press).

Depends R (>= 4.1.0)

License BSD_3_clause + file LICENSE

Encoding UTF-8

LazyData false

RoxygenNote 7.1.1

biocViews Software, CRISPR, Bayesian, DataImport

BugReports https://github.com/sellerslab/gemini/issues

Imports dplyr, grDevices, ggplot2, magrittr, mixtools, scales, pbmcapply, parallel, stats, utils

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/gemini

git_branch RELEASE_3_19

git_last_commit ef78bcb

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-01

Author Mahdi Zamanighomi [aut], Sidharth Jain [aut, cre]

Maintainer Sidharth Jain <sidharthsjain@gmail.com>

Contents

`.median_normalize` *Median normalization*

Description

Median normalization

Usage

```
.median_normalize(counts, CONSTANT = 32)
```

Arguments

`counts`	a counts matrix derived from Input
`CONSTANT`	a pseudo-count to use (default = 32)

Value

median-normalized counts object

Examples

```
## Not run:
.median_normalize(rnorm(n = 1000))

## End(Not run)
```

`counts` *Big Papi counts matrix*

Description

Counts matrix from the Big Papi SynLet dataset. The Big Papi dataset was published in Najm et al. 2018 (doi: 10.1038/nbt.4048). The counts were acquired from Supplementary Table 3.

Usage

```
counts
```

Format

An object of class `matrix` (inherits from `array`) with 9216 rows and 16 columns.

See Also

https://www.nature.com/articles/nbt.4048

`gemini_boxplot` *gemini_boxplot*

Description

A function to visualize the results of GEMINI over the raw data.

Usage

```
gemini_boxplot(
  Model,
  g,
  h,
  nc_gene = NULL,
  sample,
  show_inference = TRUE,
  color_x = FALSE,
  identify_guides = FALSE
)
```

Arguments

Model	a gemini.model object
g	a character naming a gene to visualize
h	a character naming another gene to visualize
nc_gene	a character naming the gene to use as a negative control, to be paired with each individual g and h. Defaults to Model$nc_gene.
sample	a character naming the sample to visualize
show_inference	a logical indicating whether to show the inferred individual or combined values for each gene/gene pair (default TRUE)
color_x	a logical indicating whether to visualize the sample-independent effects for each individual guide or guide pair (default FALSE)
identify_guides	a logical indicating whether to identify guides with unique colors and shapes (default FALSE)

Details

Raw LFC data is plotted for each gene combination ('g'-'nc_gene', 'h'-'nc_gene', 'g'-'h') in a standard boxplot. Horizontal green line segments are plotted over the box plots indicating the individual gene effects or the inferred total effect of a particular gene combination. Each guide pair can be colored based on the inferred sample independent effects g_i, h_j, and g_i, h_j. Additionally, colors and shapes can be used to distinguish unique guides targeting gene g and h, respectively.

Value

a ggplot2 object

Examples

```
data("Model", package = "gemini")

gemini_boxplot(Model, g = "BRCA2", h = "PARP1", nc_gene = "CD81",
sample = "A549", show_inference = TRUE,
color_x = FALSE, identify_guides = FALSE)
```

---

gemini_calculate_lfc *Calculate log-fold change*

Description

Given a gemini.input object, calculates log-fold change from counts.

Usage

```
gemini_calculate_lfc(Input, counts = "counts", sample.column.name = "samplename",
normalize = TRUE, CONSTANT = 32)
```

Arguments

Input	a gemini.input object containing an object named counts.
counts	a character indicating the name of a matrix of counts within Input that can be used to calculate log-fold changes (defaults to "counts").
sample.column.name	a character or integer indicating which column of Input$replicate.map describes the samples.
normalize	a logical indicating if counts should be median-normalized, see Details (default = TRUE)
CONSTANT	a numeric indicating a constant value that shifts counts to reduce outliers, see Details (default = 32).

Details

See Methods from Zamanighomi et al. 2019 for a comprehensive description of the calculation of log-fold change, normalization, and count processing.

If multiple early time-points are provided for a given sample, they are treated as replicates and averaged, and used to compute log-fold change against any specified late time points.

If a sample has a specific early-time point, these are matched as long as the sample names are identical between the early and late timepoints in sample.column.name

Value

a gemini object identical to Input that also contains new objects called LFC and sample.annot.

Examples

```
data("Input", package = "gemini")
Input <- gemini_calculate_lfc(Input)

head(Input$LFC)
```

---

gemini_create_input *gemini_create_input*

Description

Creates a gemini.input object from a counts matrix with given annotations.

Usage

```
gemini_create_input(
  counts.matrix,
  sample.replicate.annotation = NULL,
  guide.annotation = NULL,
  samplesAreColumns = TRUE,
  sample.column.name = "samplename",
  gene.column.names = NULL,
  ETP.column = 1,
  LTP.column = NULL,
  verbose = FALSE
)
```

Arguments

counts.matrix	a matrix of counts with rownames corresponding to features (e.g. guides) and colnames corresponding to samples.
sample.replicate.annotation	a data.frame of annotations for each sample/replicate pair. Note that at least one column in sample.replicate.annotation must correspond to the colnames of counts.matrix (see Details) (default = NULL)
guide.annotation	a data.frame of annotations for each guide. Note that at least one column in guide.annotation must correspond to the rownames of counts.matrix (default = NULL)
samplesAreColumns	a logical indicating if samples are on the columns or rows of counts.matrix. (default = TRUE)
sample.column.name	a character or integer indicating which column of sample.replicate.annotation describes the samples.
gene.column.names	a character or integer vector of length(2) indicating which columns of guide.annotation describe the genes being targeted.
ETP.column	a character or integer vector indicating which column(s) of counts.matrix contain the early time-point(s) of the screen (i.e. pDNA, early sequencing, etc.). Defaults to the first column.
LTP.column	a character or integer vector indicating which column(s) is the later time-point of the screen (i.e. day21, post-treatment, etc.). Defaults to (1:ncol(counts.matrix))[-ETP.column], or all other columns except for those specified by ETP.column.
verbose	Verbosity (default FALSE)

Details

This function initializes a gemini.input object from a counts matrix. There are a few key assumptions made in the input format.

- The counts matrix is regular.
- The counts matrix structure is in accordance with the `samplesAreColumns` parameter.
- The first column of `sample.replicate.annotation` matches with the existing dimension names of the counts matrix.
- The first column of `guide.annotations` matches with the existing dimension names of the counts matrix.
- `sample.column.name` must specify a column in `sample.replicate.annotation` (either by name or index) that describes unique samples.
- `gene.column.names` must specify two columns in `sample.replicate.annotation` (either by name or index) that describe genes.

Value

a gemini.input object

Examples

```
data("counts", package = "gemini")
data("sample.replicate.annotation", package = "gemini")
data("guide.annotation", package = "gemini")
Input <- gemini_create_input(
    counts.matrix = counts,
    sample.replicate.annotation = sample.replicate.annotation,
    guide.annotation = guide.annotation,
    sample.column.name = "samplename",
    gene.column.names = c("U6.gene", "H1.gene")
)
```

`gemini_inference` *gemini_inference*

Description

Estimate the posterior using a variational inference technique. Inference is performed through an iterative process until convergence.

Usage

```
gemini_inference(
  Model,
  n_iterations = 20,
  mean_x = 1,
  sd_x = 1,
  mean_xx = 1,
  sd_xx = 1,
  mean_y = 0,
```

```
  sd_y = 10,
  mean_s = 0,
  sd_s = 10,
  threshold = 0.001,
  cores = 1,
  force_results = FALSE,
  verbose = FALSE,
  save_iterations = FALSE
)
```

Arguments

Model	an object of class gemini.model
n_iterations	a numeric indicating the maximum number of iterations (default=20).
mean_x	a numeric indicating prior mean of x (default=1).
sd_x	a numeric indicating prior sd of x (default=1).
mean_xx	a numeric indicating prior mean of xx (default=1).
sd_xx	a numeric indicating prior sd of xx (default=1)
mean_y	a numeric indicating prior mean of y (default=0)
sd_y	a numeric indicating prior sd of y (default=10).
mean_s	a numeric indicating prior mean of s(default=0)
sd_s	a numeric indicating prior sd of s (default=10)
threshold	a numeric indicating the threshold of change in MAE at which to stop the iterative process (default=0.001).
cores	a numeric indicating the number of cores to use. See details in gemini_parallel. (default=1)
force_results	a logical indicating if the CAVI algorithm should be halted if non-convergence is detected. (default=FALSE)
verbose	default FALSE
save_iterations	for especially large libraries that require long computations, saves the latest iteration of each update. default FALSE

Details

GEMINI uses the following parameters, which are described in Zamanighomi et al. and translated here for clarity:

- **y:** individual gene effect
- **s:** combination effect
- **x:** screen variation corresponding to individual guides
- **xx:** screen variation corresponding to paired guides

Default parameters may need to be changed if convergence is not achieved. See README for more details.

Value

a gemini.model object with estimated posteriors

Examples

```
data("Model", package = "gemini")
Model %<>% gemini_inference(verbose = FALSE, n_iterations = 1) # iterations set to 1 for testing
```

`gemini_initialize` *gemini_initialize*

Description

Creates a gemini.model object given Input data and initialization parameters.

Usage

```
gemini_initialize(
  Input,
  guide.pair.annot = "guide.pair.annot",
  replicate.map = "replicate.map",
  sample.column.name = "samplename",
  LFC.name = "LFC",
  nc_gene,
  CONSTANT = 32,
  concordance = 1,
  prior_shape = 0.5,
  pattern_join = ";",
  pattern_split = ";",
  window = NULL,
  monotonize = FALSE,
  verbose = FALSE,
  cores = 1,
  save = NULL
)
```

Arguments

`Input`	an object of class gemini.input
`guide.pair.annot`	the name of an object within `Input` containing guide to gene annotations (Default = "guide.pair.annot")
`replicate.map`	the name of an object within `Input` containing replicate and sample mappings (Default = "replicate.map")

`sample.column.name`
: a character or integer indicating which column of `Input$replicate.map` describes the samples.

`LFC.name`
: a character indicating an object within Input to treat as LFC. By default, "LFC" is used, which is the output of `gemini_calculate_lfc`.

`nc_gene`
: a character naming the gene to use as a negative control. See details for more.

`CONSTANT`
: a numeric indicating a constant value that shifts counts to reduce outliers, see Details (default = 32).

`concordance`
: a numeric value to initialize x (default = 1)

`prior_shape`
: shape parameter of Gamma distribution used to model the variation in the data in `Input` (default = 0.5)

`pattern_join`
: a character to join the gene combinations found in `guide.pair.annot`. Default ':'

`pattern_split`
: a character to split the guide combinations found in `guide.pair.annot`. Default ':'

`window`
: a numeric if window smoothing should be done on initialized tau values, otherwise NULL (default) for no window smoothing

`monotonize`
: a logical specifying whether the variance should be monotonically increasing. (default FALSE)

`verbose`
: default FALSE

`cores`
: numeric indicating the number of cores to use. See details in `gemini_parallelization`.

`save`
: a character (file path) or logical indicating whether the initialized model should be saved.

Details

guide.pair.annot is created with the following format:

- 1st column contains guide pairs, joined by `pattern_split`.
- 2nd column contains the name of gene mapping to the first guide in the 1st column.
- 3rd column contains the name of gene mapping to the second guide in the 1st column.

Additional columns may be appended to guide.pair.annot, but the first three columns must be defined as above.

Use of negative control As described in Zamanighomi et al., it is highly recommended to specify a negative control. In the event that no negative control is specified, GEMINI will use the median LFC of all guide pairs targeting a gene to initialize that gene's effect. While this may be reasonable in large all-by-all screens, it is **not recommended** in smaller screens or some-by-some screens. As a result, when possible, be sure to specify a negative control.

Value

a Model object of class gemini.model

Examples

```
data("Input", package = "gemini")
Model <- gemini_initialize(Input, nc_gene = "CD81")
```

`gemini_parallelization`

Parallelization in GEMINI

Description

Notes about parallelization in combinatorial CRISPR analysis

Implementation

To improve efficiency and scalability, GEMINI employs parallelization, enabling a rapid initialization and update routine. Parallelization was implemented using the R `pbmclapply` package, and specifically through the `pbmclapply` function. As pbmclapply (and it's parent function, `mclapply`) relies on forking, parallelization is currently limited to Unix-like (Linux flavors and macOS) machines, but may be extended to other OS in later versions through socket parallelization. In the event that GEMINI is used on a Windows machine, all computations are performed in serial.

Parallelized processes

GEMINI enables parallelization of both initialization and inference. For initialization, parallelization is used to quickly hash the data. For the inference procedure, parallelization is used to speed up the CAVI approach by performing updates independently across cores.

Caveats

To note, there is usually a trade-off in terms of number of cores and shared resources/memory transfer. See inst/figs/gemini-benchmarking.png for a visual depiction.

Also, while most functions in GEMINI have been parallelized, initialization of tau, update of x (group 3), and updates of y are performed in serial. As such, especially in large libraries, tau initialization and x (group 3) updates may take some time. Updating y is usually fast, as the number of genes tends to be the smallest in parameter space. However, these functions may be parallelized in the future as well.

Active Development

Noticed that in some cases, after using multicore processing through pbmclapply, warnings have been produced: "In selectChildren(pids[!fin], -1) : cannot wait for child ... as it does not exist" "In parallel::mccollect(...) : 1 parallel job did not deliver a result" R.version=3.6.0 These warnings, although they appear menacing, are in fact harmless. See: http://r.789695.n4.nabble.com/Strange-error-messages-from-parallel-mcparallel-family-under-3-6-0-td4756875.html#a4756939

gemini_plot_mae *MAE plot*

Description

Plots the average MAE of gemini model at each iteration step.

Usage

```
gemini_plot_mae(Model)
```

Arguments

Model	a gemini.model object

Value

a ggplot2 object

Examples

```
data("Model", package = "gemini")
gemini_plot_mae(Model)
```

gemini_prepare_input *Prepare input before Model creation*

Description

This is an internal function to GEMINI, allowing for data cleanup and preprocessing before a Model object is created. This removes any gene pairs targeting the same gene twice, and removes any empty samples/replicates.

Usage

```
gemini_prepare_input(Input, gene.columns, sample.col.name = "samplename")
```

Arguments

Input	An object of class gemini.input
gene.columns	a character vector of length(2)
sample.col.name	a character indicating the name of the sample column (default = "samplename")

Value

a (prepared) gemini.input object

Examples

```
data("Input", package = "gemini")
Input %<>% gemini_prepare_input(gene.columns = c("U6.gene", "H1.gene"))
```

---

gemini_score *Scoring Combination Effect*

Description

Score genetic interactions from a gemini.model and produce a gemini.score object, and generate p-values and FDRs if negative control pairs (nc_pairs) are available.

Usage

```
gemini_score(
  Model,
  pc_gene = NA,
  pc_threshold = NULL,
  pc_weight = 0.5,
  nc_pairs = NA
)
```

Arguments

Model	an object of class gemini.model
pc_gene	a character vector of any length naming genes to use as positive controls
pc_threshold	a numeric value to indicate the LFC corresponding to a positive control, if no pc_gene is specified.
pc_weight	a weighting applied to the positive control (pc_gene) to filter genes whose individual phenotype is more lethal than $pc_weight * y(pc_gene)$.
nc_pairs	a set of non-interacting gene pairs to define statistical significance.

Value

An object of class gemini.score containing score values for strong interactions and sensitive lethality and recovery, and if nc_pairs is specified, statistical significance for each scoring type.

Examples

```
data("Model", package = "gemini")
Score <- gemini_score(Model, pc_gene = "EEF2")
```

guide.annotation *Big Papi guide annotations*

Description

Guide and gene annotations for the Big Papi SynLet dataset. The Big Papi dataset was published in Najm et al. 2018 (doi: 10.1038/nbt.4048). The guide and gene annotations were acquired from Supplementary Table 2.

Usage

```
guide.annotation
```

Format

An object of class `data.frame` with 9216 rows and 7 columns.

See Also

https://www.nature.com/articles/nbt.4048

initialize_s *initialize_s*

Description

Initialize s values using initialized y values and data from `Input`

Usage

```
initialize_s(Model, cores = 1, verbose = FALSE)
```

Arguments

`Model`	an object of class gemini.model
`cores`	a numeric indicating the number of cores to use. See `gemini_parallelization` default 1.
`verbose`	default FALSE

Value

a Model object of class gemini.model including new slots for s values

Examples

```
data("Model", package = "gemini")
Model <- initialize_s(Model)
```

initialize_tau *initialize_tau*

Description

Initialize all tau values based on the observed replicate variance.

Usage

```
initialize_tau(
  Model,
  CONSTANT = 32,
  prior_shape = 0.5,
  window = NULL,
  monotonize = FALSE,
  verbose = FALSE
)
```

Arguments

Model	an object of class gemini.model
CONSTANT	a numeric indicating a constant value that shifts counts to reduce outliers (default = 32).
prior_shape	shape parameter of Gamma distribution used to model the variation in the data in Input. If single numeric value, then shape parameters for all samples are assumed equal. Otherwise, a named numeric vector of shape parameters the same length as the number of samples (excluding early time point).
window	numeric if window smoothing should be done on initialized tau values, otherwise NULL (default) for no window smoothing
monotonize	logical specifying whether the variance should be monotonically increasing (default FALSE)
verbose	default FALSE

Value

a Model object of class gemini.model including new slots for alpha and beta values

Examples

```
data("Model", package = "gemini")
Model <- initialize_tau(Model, CONSTANT = 32, prior_shape = 0.5)
```

initialize_x *intialize_x*

Description

Initialize all x values to the value of concordance

Usage

```
initialize_x(Model, concordance = 1, cores = 1, verbose = FALSE)
```

Arguments

Model	a Model object of class gemini.model
concordance	a numeric value to initialize x
cores	a numeric indicating the number of cores to use. See gemini_parallelization default 1.
verbose	default FALSE

Value

a Model object of class gemini.model including new slots for x values and internal-use hashes

Note

As there is much hashing involved in this function, this tends to be computationally intensive. As such, we have enabled parallelization of most hash steps, but this may still be rate-limited by the amount of memory consumed.

Examples

```
data("Model", package = "gemini")
Model %<>% initialize_x()
```

initialize_y *initialize_y*

Description

Initialize all y values from guide pairs including a negative control.

Usage

```
initialize_y(Model, verbose = FALSE, cores = 1)
```

Arguments

`Model`	an object of class gemini.model
`verbose`	default FALSE
`cores`	a numeric indicating the number of cores to use. See details in `gemini_parallelization`. (default=1)

Value

a Model object of class gemini.model including new slots for y values

Examples

```
data("Model", package = "gemini")
Model %<>% initialize_y()
```

`Input` *Input object from Big Papi*

Description

A gemini.input object created from the Big Papi SynLet dataset. The Big Papi dataset was published in Najm et al. 2018 (doi: 10.1038/nbt.4048). The Input object here is created from the data in counts, guide.annotation, and sample.replicate.annotation using the `gemini_create_input` function.

Usage

```
Input
```

Format

An object of class `list` (inherits from `gemini.input`, `gemini.input`) of length 6.

See Also

https://www.nature.com/articles/nbt.4048

Model *Model object from Big Papi*

Description

A gemini.model object created from the Big Papi SynLet dataset after `gemini_inference` was run. The Big Papi dataset was published in Najm et al. 2018 (doi: 10.1038/nbt.4048). The Model object here is created from the data in Input using the `gemini_initialize` function.

Usage

```
Model
```

Format

An object of class `list` (inherits from `gemini.model`) of length 24.

See Also

https://www.nature.com/articles/nbt.4048

sample.replicate.annotation *Big Papi sample and replicate annotations*

Description

Sample and replicate annotations for the Big Papi SynLet dataset. The Big Papi dataset was published in Najm et al. 2018 (doi: 10.1038/nbt.4048). The sample and replicate annotations were acquired from Supplementary Table 3

Usage

```
sample.replicate.annotation
```

Format

An object of class `data.frame` with 16 rows and 3 columns.

See Also

https://www.nature.com/articles/nbt.4048

Sgene2Pguides_hash *Gene hashing*

Description

Gene hashing

Usage

```
Sgene2Pguides_hash(guide2gene, cores = 1)
```

Arguments

guide2gene	derived from Input object
cores	number of cores to use (default 1)

Value

Gene hash/list

Examples

```
## Not run:
#' data("Input", package = "gemini")
Sgene2Pguides_hash(Input$guide.pair.annot[1:10,])

## End(Not run)
```

Sguide2Pguides_hash *Guide hashing*

Description

Guide hashing

Usage

```
Sguide2Pguides_hash(guide2gene, split = ";", cores = 1)
```

Arguments

guide2gene	derived from Input object
split	character to split guides
cores	number of cores to use (default 1)

Value

Guide hash/list

Examples

```
## Not run:
data("Input", package = "gemini")
Sguide2Pguides_hash(gemini::Input$guide.pair.annot[1:10,])

## End(Not run)
```

---

update_mae *update_mae*

Description

Calculate mean absolute error across all guide combinations.

Usage

```
update_mae(Model, verbose = FALSE)
```

Arguments

Model	a Model object of class gemini.model
verbose	default FALSE

Value

An object of class gemini.model

Examples

```
data("Model", package = "gemini")
Model %<>% update_mae()
```

update_s_pb *update_s*

Description

Update values of s using data from Input and current values of other parameters.

Usage

```
update_s_pb(Model, mean_s = 0, sd_s = 10, cores = 1, verbose = FALSE)
```

Arguments

Model	a Model object of class gemini.model
mean_s	numeric indicating prior mean of s (default 0)
sd_s	numeric indicating prior sd of s (default 10)
cores	a numeric indicating the number of cores to use, see gemini_parallelization. default=1.
verbose	default FALSE

Value

An object of class gemini.model

Examples

```
data("Model", package = "gemini")
Model %<>% update_s_pb()
```

update_tau_pb *update_tau_pb*

Description

Update parameters of tau using data from Input and current values of other parameters.

Usage

```
update_tau_pb(Model, cores = 1, verbose = FALSE)
```

Arguments

Model	a Model object of class gemini.model
cores	a numeric indicating the number of cores to use. See gemini_parallelization for details. (default=1).
verbose	default FALSE

Value

An object of class gemini.model

Examples

```
data("Model", package = "gemini")
Model %<>% update_tau_pb()
```

---

update_x_pb *update_x_p*

Description

Update values of x using data from Input and current values of other parameters.

Usage

```
update_x_pb(
  Model,
  mean_x = 1,
  sd_x = 1,
  mean_xx = 1,
  sd_xx = 1,
  cores = 1,
  verbose = FALSE
)
```

Arguments

Model	a Model object of class gemini.model
mean_x	a numeric indicating prior mean of x
sd_x	a numeric indicating prior sd of x
mean_xx	a numeric indicating prior mean of xx
sd_xx	a numeric indicating prior sd of xx
cores	a numeric indicating the number of cores to use. See gemini_parallelization for details. (default=1).
verbose	default FALSE

Value

An object of class gemini.model

Note

The structure of the screen may impede parallelization. Our ability to parallelize the updates is contingent upon the independence between guides in position 1 and 2. To account for potential dependence, we define three groups of guides: group 1 (guides only in position 1), group 2 (guides only in position 2), and group 3 (guides in both position 1 and position 2). Parallelization is possible for groups 1 and 2, but only serial updates are possible for group 3. As such, updates for this group will take longer.

Examples

```
data("Model", package = "gemini")
Model %<>% update_x_pb()
```

`update_y_pb` *update_y_pb*

Description

Update values of y using data from `Input` and current values of other parameters.

Usage

```
update_y_pb(Model, mean_y = 0, sd_y = 10, verbose = FALSE)
```

Arguments

`Model`	a Model object of class gemini.model
`mean_y`	numeric indicating prior mean of y
`sd_y`	numeric indicating prior sd of y
`verbose`	default FALSE

Value

An object of class gemini.model

Examples

```
data("Model", package = "gemini")
Model %<>% update_y_pb()
```

%<>% *Compound pipe*

Description

Compound pipe

Arguments

lhs, rhs	An object modified in place, and a function to apply to it

Examples

```
a <- 1:5
a %<>% sum() # a is modified in place
```

%>% *Pipe*

Description

Pipe

Arguments

lhs, rhs	An object, and a function to apply to it

Examples

```
a <- 1:5
b <- a %>% sum()
```

Index