Package 'fbar'

September 2, 2020

Type Package

Title An Extensible Approach to Flux Balance Analysis

Version 0.6.0

Date 2020-09-02

Description A toolkit for Flux Balance Analysis and related metabolic modeling techniques. Functions are provided for: parsing models in tabular format, converting parsed metabolic models to input formats for common linear programming solvers, and evaluating and applying gene-protein-reaction mappings. In addition, there are wrappers to parse a model, select a solver, find the metabolic fluxes, and return the results applied to the original model. Compared to other packages in this field, this package puts a much heavier focus on providing reusable components that can be used in the design of new implementation of new techniques, in particular those that involve large parameter sweeps. For a background on the theory, see What is Flux Balance Analysis <doi:10.1038/nbt.1614>.

License GPL-3

URL http://maxconway.github.io/fbar/, https://github.com/maxconway/fbar

BugReports https://github.com/maxconway/fbar/issues

Depends R (>= 3.3.0)

Imports assertthat, dplyr, magrittr, Matrix, purrr, rlang, ROI, ROI.plugin.ecos, stringr, tibble, tidyr

Suggests spelling, jsonlite, curl, knitr, rmarkdown, testthat

VignetteBuilder knitr

RoxygenNote 7.1.1

LazyData true

Language en-GB

NeedsCompilation no

Author Max Conway [aut, cre]

Maintainer Max Conway <conway.max1@gmail.com>

Repository CRAN

Date/Publication 2020-09-02 13:50:03 UTC

R topics documented:

`decompose_metabolites` *Decompose a metabolite table into the metabolite stub itself and the compartment it is in*

Description

Decompose a metabolite table into the metabolite stub itself and the compartment it is in

Usage

```
decompose_metabolites(
  met_table,
  compartment_regex = "(\\[[a-zA-Z0-9]+]$)|(_[a-zA-Z]$)"
)
```

Arguments

`met_table`	A metabolite table, with one column, `met`
`compartment_regex`	Regular expression to identify compartments in model

Value

a metabolite table with the columns `chemical` and `compartment`

Examples

```
data(ecoli_core)

mod <- reactiontbl_to_expanded(ecoli_core)

decompose_metabolites(mod$mets)

recompose_metabolites(decompose_metabolites(mod$mets))
```

`ecoli_core` *A small E. coli model, created from a number of sources.*

Description

A small E. coli model, created from a number of sources.

Usage

```
ecoli_core
```

Format

A data frame with 95 rows and 7 columns:

abbreviation an abbreviated reaction name, acts as the reaction id

lowbnd lower bound on the reaction rate

uppbnd upper bound on the reaction rate

obj_coef identifies a reaction (or reactions) for which the maximum possible rate should be found

equation reaction equation

officialName full reaction name

geneAssociation A boolean combination of genes which control the reaction

subsystem an indicator of reaction function

Source

`http://bigg.ucsd.edu`, Reconstruction and Use of Microbial Metabolic Networks: the Core Escherichia coli Metabolic Model as an Educational Guide, A comprehensive genome-scale reconstruction of Escherichia coli metabolism–2011.

expanded_to_glpk *Parse a long format metabolic model to a glpk model*

Description

This function is deprecated. `ROI.plugin.glpk` **is recommended instead.**

Usage

```
expanded_to_glpk(reactions_expanded)
```

Arguments

`reactions_expanded`	A list of data frames as output by `reactiontbl_to_expanded`

Details

This parses the long format produced by `reactiontbl_to_expanded` to a glpk model.

To install the Rglpk package in Linux, run `sudo apt-get install libglpk-dev` in a terminal, and then run `install.packages('Rglpk')` in R.

The `reaction_table` must have columns:

- `abbreviation`,
- `equation`,
- `uppbnd`,
- `lowbnd`, and
- `obj_coef`.

Value

A list suitable for input to Rglpk

See Also

Other parsing_and_conversion: `expanded_to_ROI()`, `expanded_to_gurobi()`, `reactiontbl_to_expanded()`, `reactiontbl_to_gurobi()`

expanded_to_gurobi *Parse a long format metabolic model to a Gurobi model*

Description

This function is deprecated. `github.com/Fl0Sch/ROI.plugin.gurobi` **is recommended instead.**

Usage

```
expanded_to_gurobi(reactions_expanded)
```

Arguments

`reactions_expanded`	A list of data frames as output by `expand_reactions`

Details

Used as the second half of `reactiontbl_to_gurobi`, this parses the long format produced by `reactiontbl_to_expanded` to a Gurobi model

For installation instructions for Gurobi, refer to the Gurobi website: https://www.gurobi.com/.

The `reaction_table` must have columns:

- `abbreviation`,
- `equation`,
- `uppbnd`,
- `lowbnd`, and
- `obj_coef`.

Value

A list suitable for input to Gurobi.

See Also

Other parsing_and_conversion: `expanded_to_ROI()`, `expanded_to_glpk()`, `reactiontbl_to_expanded()`, `reactiontbl_to_gurobi()`

expanded_to_reactiontbl *Convert intermediate expanded format back to a reaction table*

Description

Useful for saving a new or edited model

Usage

```
expanded_to_reactiontbl(expanded)
```

Arguments

expanded	A list of data frames: • rxns, which has one row per reaction, • mets, which has one row for each metabolite, and • stoich, which has one row for each time a metabolite appears in a reaction.

Value

A data frame describing the metabolic model.

expanded_to_ROI *Parse a long format metabolic model to an ROI model*

Description

This parses the long format produced by reactiontbl_to_expanded to an ROI model.

Usage

```
expanded_to_ROI(reactions_expanded)
```

Arguments

reactions_expanded	A list of data frames as output by reactiontbl_to_expanded

Details

To solve models using ROI, you will need a solver plugin for ROI. Probably the easiest one to install is ROI.plugin.glpk. To install this in Linux, run `sudo apt-get install libglpk-dev` in a terminal, and then run `install.packages('ROI.plugin.glpk')` in R.

The `reaction_table` must have columns:

- `abbreviation`,
- `equation`,
- `uppbnd`,
- `lowbnd`, and
- `obj_coef`.

Value

A list suitable for input to ROI.

See Also

Other parsing_and_conversion: `expanded_to_glpk()`, `expanded_to_gurobi()`, `reactiontbl_to_expanded()`, `reactiontbl_to_gurobi()`

Examples

```
## Not run:
data(ecoli_core)
library(dplyr)
try(library(ROI.plugin.ecos)) # make a solver available to ROI

roi_model <- ecoli_core %>%
  reactiontbl_to_expanded %>%
  expanded_to_ROI

if(length(ROI::ROI_applicable_solvers(roi_model))>=1){
  roi_result <- ROI::ROI_solve(roi_model)

  ecoli_core_with_flux <- ecoli_core %>%
    mutate(flux = roi_result[['solution']])
}

## End(Not run)
```

fbar *fbar: Flux Balance Analysis in R with a tidy data approach*

Description

fbar is a simple, easy to use Flux Balance Analysis package with a tidy data approach. Just `data_frames` and the occasional `list`, no new classes to learn. The focus is on simplicity and speed. Models are expected as a flat table, and results can be simply appended to the table. This makes this package very suitable for use in pipelines with pre- and post- processing of models and results, so that it works well as a backbone for customized methods. Loading, parsing and evaluating a model takes around 0.1s, which, together with the straightforward data structures used, makes this library very suitable for large parameter sweeps.

Details

For a list of functions in the package, see `vignette('Introduction','fbar')`

find_fluxes_df *Given a metabolic model as a data frame, return a new data frame with fluxes*

Description

Given a metabolic model as a data frame, return a new data frame with fluxes

Usage

```
find_fluxes_df(reaction_table, do_minimization = FALSE)
```

Arguments

`reaction_table`	a data frame representing the metabolic model
`do_minimization`	toggle to uniformly minimize all non-objective fluxes after finding the objective

Details

This function uses ROI, so to solve models, you will need a solver plugin for ROI. Probably the easiest one to install is ROI.plugin.glpk. To install this in Linux, run `sudo apt-get install libglpk-dev` in a terminal, and then run `install.packages('ROI.plugin.glpk')` in R.

Value

The input data frame with a new numeric column, "`flux`".

See Also

find_fluxes_vector

Examples

```
## Not run:
data(ecoli_core)
ecoli_core_with_flux <- find_fluxes_df(ecoli_core)

## End(Not run)
```

`find_flux_variability_df` *Given a metabolic model as a data frame, return a new data frame with fluxes and variability*

Description

This function calculates fluxes `folds` times with shuffled versions of the metabolic model. This is designed to detect and quantify underdetermined fluxes.

Usage

```
find_flux_variability_df(reaction_table, folds = 10, do_minimization = TRUE)
```

Arguments

`reaction_table`	a data frame representing the metabolic model
`folds`	number of times to calculate fluxes
`do_minimization`	toggle to uniformly minimize all non-objective fluxes after finding the objective

Details

This function uses ROI, so to solve models, you will need a solver plugin for ROI. Probably the easiest one to install is ROI.plugin.glpk. To install this in Linux, run `sudo apt-get install libglpk-dev` in a terminal, and then run `install.packages('ROI.plugin.glpk')` in R.

Value

reaction_table with two added columns: sd (the standard deviation of fluxes found) and flux (a typical flux) from this distribution

gene_associate *Apply gene expressions to reaction table*

Description

A convenience function that uses gene_eval and a custom function to apply new upper and lower bounds.

Usage

```
gene_associate(
  reaction_table,
  gene_table,
 expression_flux_function = function(x) {    (1 + log(x)/stats::sd(x)^2)^sign(x - 1)
    }
)
```

Arguments

reaction_table	A data frame describing the metabolic model.
gene_table	A data frame showing gene presence
expression_flux_function	a function to convert from gene set expression to flux

Value

the reaction_table, with a new column, present, and altered upper and lower bounds

Warning

This function relies on gene_eval, which uses eval to evaluate gene expression sets. This gives flexibility, but means that malicious code in the gene_sets argument could get evaluated. gene_sets is evaluated in a restricted environment, but there might be a way around this, so you might want to check for anything suspicious in this argument manually. For more information, read the code.

See Also

gene_eval

Examples

```
data(iJO1366)
library(dplyr)

gene_table = tibble(name = iJO1366$geneAssociation %>%
stringr::str_split('and|or|\\s|\\(|\\)') %>%
  purrr::flatten_chr() %>%
  unique,
```

```
presence = 1) %>%
  filter(name != '', !is.na(name))

gene_associate(reaction_table = iJO1366 %>%
                 mutate(geneAssociation = geneAssociation %>%
                          stringr::str_replace_all('and', '&') %>%
                          stringr::str_replace_all('or', '|')
                 ),
               gene_table = gene_table
)
```

---

gene_eval *Function to estimate the expression levels of gene sets*

Description

Function to estimate the expression levels of gene sets

Usage

```
gene_eval(gene_sets, genes, presences)
```

Arguments

gene_sets	A list of gene set strings: names of genes punctuated with &, | and brackets.
genes	A list of gene names
presences	A list of gene presences, the same length as genes

Value

a vector the same length as gene_sets, with the the calculated combined gene expression levels.

This function evaluates the gene sets in the context of the gene presences. It can take booleans, or numbers, in which case it associates & with finding the minimum, and | with finding the maximum.

Warning

This function uses eval to evaluate gene expression sets. This gives flexibility, but means that malicious code in the gene_sets argument could get evaluated. gene_sets is evaluated in a restricted environment, but there might be a way around this, so you might want to check for anything suspicious in this argument manually. For more information, read the code.

See Also

gene_associate

get_BiGG *Download a model from a BiGG json file*

Description

Download a model from a BiGG json file

Usage

```
get_BiGG(address)
```

Arguments

address	An address to download from

Value

A model in expanded format

iJO1366 *A full size E. coli model.*

Description

A full size E. coli model.

Usage

```
iJO1366
```

Format

A data frame with 2,583 rows and 10 columns:

abbreviation an abbreviated reaction name, acts as the reaction id
lowbnd lower bound on the reaction rate
uppbnd upper bound on the reaction rate
obj_coef identifies a reaction (or reactions) for which the maximum possible rate should be found
equation reaction equation
officialName full reaction name
geneAssociation A boolean combination of genes which control the reaction
subsystem an indicator of reaction function

Source

http://bigg.ucsd.edu, A comprehensive genome-scale reconstruction of Escherichia coli metabolism–2011.

nutrient_types — *A subset of exchange reactions annotated to indicate typical availability*

Description

A subset of exchange reactions annotated to indicate typical availability

Usage

```
nutrient_types
```

Format

A data frame with 25 rows and 2 columns:

abbreviation an exchange reaction id

nutrient_type the nutrient availability, one of 'micro', 'macro' or 'substrate'

parse_met_list — *Internal function: Expand half reaction equations into a long form*

Description

Internal function: Expand half reaction equations into a long form

Usage

```
parse_met_list(mets)
```

Arguments

mets	Character vector of halves of reaction equations.

Value

a date_frame with columns:

stoich the stoichiometric coefficient

met the metabolite

`reactiontbl_to_expanded` *Parse a reaction table to an intermediate, long format*

Description

The long format can also be suitable for manipulating equations.

Usage

```
reactiontbl_to_expanded(reaction_table, regex_arrow = "<?[-=]+>")
```

Arguments

`reaction_table`	A data frame describing the metabolic model.
`regex_arrow`	Regular expression for the arrow splitting sides of the reaction equation.

Details

The `reaction_table` must have columns:

- `abbreviation`,
- `equation`,
- `uppbnd`,
- `lowbnd`, and
- `obj_coef`.

Value

A list of data frames:

- `rxns`, which has one row per reaction,
- `mets`, which has one row for each metabolite, and
- `stoich`, which has one row for each time a metabolite appears in a reaction.

See Also

Other parsing_and_conversion: `expanded_to_ROI()`, `expanded_to_glpk()`, `expanded_to_gurobi()`, `reactiontbl_to_gurobi()`

Examples

```
## Not run:
data(ecoli_core)
library(dplyr)
try(library(ROI.plugin.ecos)) # make a solver available to ROI

roi_model <- ecoli_core %>%
  reactiontbl_to_expanded %>%
  expanded_to_ROI

if(length(ROI::ROI_applicable_solvers(roi_model))>=1){
  roi_result <- ROI::ROI_solve(roi_model)

  ecoli_core_with_flux <- ecoli_core %>%
    mutate(flux = roi_result[['solution']])
}

## End(Not run)
```

---

reactiontbl_to_gurobi *Parse reaction table to Gurobi format*

Description

This function is deprecated. `github.com/Fl0Sch/ROI.plugin.gurobi` **is recommended instead.**

Usage

```
reactiontbl_to_gurobi(reaction_table, regex_arrow = "<?[-=]+>")
```

Arguments

`reaction_table`	A data frame describing the metabolic model.
`regex_arrow`	Regular expression for the arrow splitting sides of the reaction equation.

Details

Parses a reaction table to give a list in Gurobi's input format. This function is a shorthand for `reactiontbl_to_expanded` followed by `expanded_to_gurobi`.

The `reaction_table` must have columns:

- `abbreviation`,
- `equation`,
- `uppbnd`,
- `lowbnd`, and
- `obj_coef`.

Value

A list suitable for input to Gurobi.

See Also

Other parsing_and_conversion: `expanded_to_ROI()`, `expanded_to_glpk()`, `expanded_to_gurobi()`, `reactiontbl_to_expanded()`

`recompose_metabolites` *Merge metabolite stub and compartment to form an id*

Description

Merge metabolite stub and compartment to form an id

Usage

```
recompose_metabolites(
  expanded_metabolites,
  before_signifier = "_",
  after_signifier = ""
)
```

Arguments

`expanded_metabolites`	a metabolite table as created by `decompose_metabolites`
`before_signifier`	a string that is inserted before the compartment identifier
`after_signifier`	a string that is inserted after the compartment identifier

Value

A merged metabolite table with one column, `met`

Examples

```
data(ecoli_core)

mod <- reactiontbl_to_expanded(ecoli_core)

decompose_metabolites(mod$mets)

recompose_metabolites(decompose_metabolites(mod$mets))
```

split_on_arrow *Internal function: Splitting reaction equation into substrate and product*

Description

Internal function: Splitting reaction equation into substrate and product

Usage

```
split_on_arrow(equations, regex_arrow = "<?[-=]+>")
```

Arguments

equations	Character vector of reaction equations.
regex_arrow	Regular expression for the arrow splitting sides of the reaction equation.

Value

a data_frame, with columns:

reversible boolean, is reaction reversible

before the left hand side of the reaction string

after the right hand side of the reaction string

validate_expanded *Validate an expanded model*

Description

Validate an expanded model

Usage

```
validate_expanded(reactions_expanded)
```

Arguments

reactions_expanded	the expanded model to check

Value

TRUE

Index