Package 'CARNIVAL'

November 11, 2026

Title A CAusal Reasoning tool for Network Identification (from gene expression data) using Integer VALue programming

Version 2.22.0

Description An upgraded causal reasoning tool from Melas et al in R with updated assignments of TFs' weights from PROGENy scores.
Optimization parameters can be freely adjusted and multiple solutions can be obtained and aggregated.

URL https://github.com/saezlab/CARNIVAL

BugReports https://github.com/saezlab/CARNIVAL/issues

Depends R (>= 4.0)

Imports readr, stringr, lpSolve, igraph, dplyr, tibble, tidyr, rjson, rmarkdown

biocViews Transcriptomics, GeneExpression, Network

License GPL-3

LazyData true

Encoding UTF-8

Suggests RefManageR, BiocStyle, covr, knitr, testthat (>= 3.0.0), sessioninfo

VignetteBuilder knitr

RoxygenNote 7.1.2

Config/testthat/edition 3

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

git_branch RELEASE_3_23

git_last_commit 606bffa

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Enio Gjerga [aut] (ORCID: <https://orcid.org/0000-0002-3060-5786>),
Panuwat Trairatphisan [aut],
Anika Liu [ctb],
Alberto Valdeolivas [ctb],
Nikolas Peschke [ctb],
Aurelien Dugourd [ctb],
Attila Gabor [cre],
Olga Ivanova [aut]

Maintainer Attila Gabor <attila.gabor@uni-heidelberg.de>

Contents

addPerturbationNodes . . . 3
checkCarnivalOptions . . . 3
checkData . . . 4
checkOptionsValidity . . . 4
checkPriorKnowledgeNetwork . . . 5
createInternalDataRepresentation . . . 6
defaultCbcSolveCarnivalOptions . . . 6
defaultCplexCarnivalOptions . . . 7
defaultCplexSpecificOptions . . . 7
defaultLpSolveCarnivalOptions . . . 8
generateLpFileCarnival . . . 8
getOptionsList . . . 10
getSupportedSolvers . . . 10
getSupportedSolversFunctions . . . 11
isInputValidCarnival . . . 11
parseCplexLog . . . 12
prepareForCarnivalRun . . . 13
preprocessPriorKnowledgeNetwork . . . 13
processSolution . . . 14
readOptions . . . 15
runCARNIVAL . . . 15
runFromLpCarnival . . . 18
runInverseCarnival . . . 20
runVanillaCarnival . . . 22
sendTaskToSolver . . . 24
setCarnivalOptions . . . 25
solveCarnival . . . 25
solveCarnivalFromLp . . . 26
solveWithCbc . . . 27
solveWithGurobi . . . 27
suggestedCbcSpecificOptions . . . 28
suggestedCplexSpecificOptions . . . 28
writeCplexCommandFileFromJson . . . 29
writeParsedData . . . 29

Index **31**

addPerturbationNodes *Introduces a perturbation node connecting periphery nodes without a target in the prior knowledge network.*

Description

Introduces a perturbation node connecting periphery nodes without a target in the prior knowledge network.

Usage

```
addPerturbationNodes(priorKnowledgeNetwork)
```

Arguments

`priorKnowledgeNetwork`
: data.frame with priorKnowledgeNetwork with source, interaction, target columns.

Value

data.frame with prior knowledge network with added perturbations

Author(s)

Panuwat Trairatphisan, 2020

checkCarnivalOptions *Checks options provided for CARNIVAL*

Description

Checks options provided for CARNIVAL

Usage

```
checkCarnivalOptions(carnivalOptions)
```

Arguments

`carnivalOptions`
: all available carnival options

Value

returns TRUE if no error found.

checkData *Checks the input data for correctness.*

Description

Checks the input data for correctness.

Usage

```
checkData(
  perturbations = NULL,
  measurements,
  priorKnowledgeNetwork,
  weights = NULL
)
```

Arguments

perturbations

measurements
priorKnowledgeNetwork

weights

Value

returns list of checked data

Author(s)

Enio Gjerga, Olga Ivanova, Attila Gabor, 2020-2021

checkOptionsValidity *Checks if provided option names are valid.*

Description

Checks if provided option names are valid.

Usage

```
checkOptionsValidity(solver = getSupportedSolvers()$lpSolve, ...)
```

Arguments

solver	one of the solvers available from getSupportedSolvers().
...	any possible options from the solver's list

Value

TRUE/FALSE depending on the status of the checks

Examples

```
checkOptionsValidity(solver="lpSolve")
```

---

checkPriorKnowledgeNetwork

Checks prior knowledge network for correct format.

Description

Checks prior knowledge network for correct format.

Usage

```
checkPriorKnowledgeNetwork(priorKnowledgeNetwork)
```

Arguments

priorKnowledgeNetwork	a network with 3 columns: source node ('source'), interaction sign ('interaction') and target node('target').

Value

TRUE if everything is correct. Stops pipeline if not.

Author(s)

Enio Gjerga, Olga Ivanova 2020-2021

`createInternalDataRepresentation` *Creates internal data representation - variables for ILP solvers, on the basis of provided preprocessed data.*

Description

Creates internal data representation - variables for ILP solvers, on the basis of provided preprocessed data.

Usage

```
createInternalDataRepresentation(
  dataPreprocessed,
  newDataRepresentation = TRUE
)
```

Arguments

`dataPreprocessed`	list containing preprocessed priorKnowledgeNetwork, measurements, weights (if provided), perturbations (if provided).
`newDataRepresentation`	TRUE by default. For debugging with the old data representation, put to FALSE.

Value

variables for the new data representation or data vector (containing preprocessed information on measurement) and variables for the old data representation (CARNIVAL v.<2)

`defaultCbcSolveCarnivalOptions` *Sets default CARNIVAL options for cbc.*

Description

Sets default CARNIVAL options for cbc.

Usage

```
defaultCbcSolveCarnivalOptions(...)
```

Arguments

`...`	any possible options from the solver's list

Value

default CbB solver options as a list.

Examples

```
#defaultCbcSolveCarnivalOptions()
```

---

defaultCplexCarnivalOptions

Sets default CARNIVAL options for cplex.

Description

Sets default CARNIVAL options for cplex.

Usage

```
defaultCplexCarnivalOptions(...)
```

Arguments

...	any possible options from the solver's list

Value

default CPLEX solver options as a list.

Examples

```
defaultCplexCarnivalOptions()
```

---

defaultCplexSpecificOptions

Sets default options from cplex documentation.

Description

Sets default options from cplex documentation.

Usage

```
defaultCplexSpecificOptions(...)
```

Arguments

...	any possible options from the solver's list

Value

default CPLEX solver options as a list.

Examples

```
defaultCplexSpecificOptions()
```

`defaultLpSolveCarnivalOptions` *Sets default CARNIVAL options for lpSolve.*

Description

Sets default CARNIVAL options for lpSolve.

Usage

```
defaultLpSolveCarnivalOptions(...)
```

Arguments

`...`	any possible options from the solver's list

Value

default lpSolve solver options as a list.

Examples

```
defaultLpSolveCarnivalOptions()
```

`generateLpFileCarnival` `generateLpFileCarnival`

Description

`generateLpFileCarnival`

Usage

```
generateLpFileCarnival(
  perturbations = NULL,
  measurements,
  priorKnowledgeNetwork,
  weights = NULL,
  carnivalOptions = defaultLpSolveCarnivalOptions()
)
```

Arguments

perturbations	(optional, if inverse CARNIVAL flavour is used further) vector of targets of perturbations.
measurements	vector of the measurements (i.e. DoRothEA/VIPER normalised enrichment scores)
priorKnowledgeNetwork	data frame of the prior knowledge network
weights	(optional) vector of the additional weights: e.g. PROGENy pathway scores or measured protein activities.
carnivalOptions	the list of options for the run. See defaultLpSolveCarnivalOptions(), defaultCplexCarnivalOptions, defaultCbcCarnivalOptions.

Details

Prepares the input data for the run: tranforms data into lp file and .Rdata file. These files can be reused to run CARNIVAL without preprocessing step using runCarnivalFromLp(..)

Value

paths to .lp file and .RData file that can be used for runFromLpCarnival()

Examples

```
load(file = system.file("toy_perturbations_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_measurements_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_network_ex1.RData",
                        package="CARNIVAL"))

## lpSolve
#res1 = generateLpFileCarnival(perturbations = toy_perturbations_ex1,
#                              measurements = toy_measurements_ex1,
#                              priorKnowledgeNetwork = toy_network_ex1,
#                              carnivalOptions = defaultLpSolveCarnivalOptions())

#res1["lpFile"] ##path to generated lp file
#res1["parsedDataFile"] ##path to data file used during generation

## Examples for cbc and cplex are commented out because these solvers are not part of R environment
## and need to be installed separately
##
## cbc
## res2 = generateLpFileCarnival(perturbations = toy_perturbations_ex1,
##                               measurements = toy_measurements_ex1,
##                               priorKnowledgeNetwork = toy_network_ex1,
##                               carnivalOptions = defaultCbcCarnivalOptions())
##
## res2["lpFile"] ##path to generated lp file
```

```
## res2["parsedDataFile"] ##path to data file used during generation
##
## cplex
## res3 = generateLpFileCarnival(perturbations = toy_perturbations_ex1,
##                               measurements = toy_measurements_ex1,
##                               priorKnowledgeNetwork = toy_network_ex1,
##                               carnivalOptions = defaultCplexCarnivalOptions())
##
## res3["lpFile"] ##path to generated lp file
## res3["parsedDataFile"] ##path to data file used during generation
```

`getOptionsList` *Returns the list of options needed/supported for each solver.*

Description

Returns the list of options needed/supported for each solver.

Usage

```
getOptionsList(solver = "", onlyRequired = FALSE)
```

Arguments

`solver`	one of the solvers available from getSupportedSolvers()
`onlyRequired`	logic, set to TRUE if you want to obtain only required options for the run

Value

list of options, solver-dependent

`getSupportedSolvers` *Returns the list of supported solvers.*

Description

Returns the list of supported solvers.

Usage

```
getSupportedSolvers()
```

Value

list of currently supported solvers.

getSupportedSolversFunctions *Supported solvers functions to work with all solvers in a uniform way.*

Description

To add a new solver, one must write and add here the functions for 3 steps: solve, obtaining a solution matrix, exporting the solution matrix. More specific functions can be written and called (e.g. check saveDiagnostics in cplex).

Usage

```
getSupportedSolversFunctions()
```

Value

list of solvers and their corresponding functions.

isInputValidCarnival *Checks validity of all inputs of CARNIVAL*

Description

Checks validity of all inputs of CARNIVAL

Usage

```
isInputValidCarnival(
  perturbations = NULL,
  measurements,
  priorKnowledgeNetwork,
  weights = NULL,
  carnivalOptions = defaultLpSolveCarnivalOptions()
)
```

Arguments

perturbations	(optional, if inverse CARNIVAL flavour is used further) vector of targets of perturbations.
measurements	vector of the measurements (i.e. DoRothEA/VIPER normalised enrichment scores)
priorKnowledgeNetwork	data frame of the prior knowledge network
weights	(optional) vector of the additional weights: e.g. PROGENy pathway scores or measured protein activities.

carnivalOptions
: the list of options for the run. See defaultLpSolveCarnivalOptions(), defaultCplexCarnivalOptions, defaultCbcCarnivalOptions.

Value

TRUE if everything passed the checks.

Examples

```
load(file = system.file("toy_perturbations_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_measurements_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_network_ex1.RData",
                        package="CARNIVAL"))

## lpSolve
#isInputValidCarnival(perturbations = toy_perturbations_ex1,
#                     measurements = toy_measurements_ex1,
#                     priorKnowledgeNetwork = toy_network_ex1,
#                     carnivalOptions = defaultLpSolveCarnivalOptions())
```

---

parseCplexLog *Parses the cplex log file and reads some basic information.*

Description

Parses the cplex log file and reads some basic information.

Usage

```
parseCplexLog(log)
```

Arguments

log
: path of log file resulted from a carnival run OR the content of this file read by read_lines.

Value

list variable with following fields: - ‘convergence‘ a table that contains information on the convergence of CPLEX - ‘n_solutions‘ number of solutions found - ‘objective‘ objective function value - ‘termination_reason‘: reason of termination

Author(s)

Attila Gabor, 2021

`prepareForCarnivalRun` *Prepares ILP formulation and writes it to .lp file. Currently supports the old data representation (CARNIVAL v.<2) for debugging and testing if any problems arise with the new way to generate variables.*

Description

Prepares ILP formulation and writes it to .lp file. Currently supports the old data representation (CARNIVAL v.<2) for debugging and testing if any problems arise with the new way to generate variables.

Usage

```
prepareForCarnivalRun(
  dataPreprocessed,
  carnivalOptions,
  newDataRepresentation = TRUE
)
```

Arguments

`dataPreprocessed`
: list containing preprocessed priorKnowledgeNetwork, measurements, weights (if provided), perturbations (if provided).

`carnivalOptions`
: all options of CARNIVAL.

`newDataRepresentation`
: TRUE by default. For debugging with the old data representation, put to FALSE.

Value

list with all variables and ILP formulation written in .lp file.

`preprocessPriorKnowledgeNetwork`
Preprocesses prior knowledge network: correct nodes identifiers for symbols that might break solvers runs, assigns the types for each column: Node1 (character), Sign (numeric), Node2 (character). Stops if interaction/sign column has non-numeric value Detect and remove self-activation (would break loop constraints with CbC)

Description

Preprocesses prior knowledge network: correct nodes identifiers for symbols that might break solvers runs, assigns the types for each column: Node1 (character), Sign (numeric), Node2 (character). Stops if interaction/sign column has non-numeric value Detect and remove self-activation (would break loop constraints with CbC)

Usage

```
preprocessPriorKnowledgeNetwork(priorKnowledgeNetwork)
```

Arguments

priorKnowledgeNetwork	a network with 3 columns: source node ('source'), interaction sign ('interaction') and target node('target').

Value

preprocessed prior knowledge network with corrected nodes identifiers add 3 columns: Node1, Sign, Node2

Author(s)

Enio Gjerga, Olga Ivanova 2020-2021

---

processSolution *Exports the solution matrix to the final solution.*

Description

Exports the solution matrix to the final solution.

Usage

```
processSolution(
  solutionMatrix,
  variables,
  dataPreprocessed,
  carnivalOptions,
  newDataRepresentation = TRUE
)
```

Arguments

solutionMatrix	the output matrix from ILP solver containing variables list (rows) and their values in different solutions (columns).
variables	list of nodes, edges and measurements variables generated by createLpFormulation_v2.
dataPreprocessed	list containing preprocessed priorKnowledgeNetwork, measurements, weights (if provided), perturbations (if provided).
carnivalOptions	all options of CARNIVAL.
newDataRepresentation	TRUE by default. For debugging with the old data representation, put to FALSE.

Value

Carnival results exported from the solution matrix. see runCARNIVAL for details.

`readOptions` *Reads options from json file.*

Description

Reads options from json file.

Usage

```
readOptions(jsonFileName = "inst/carnival_cplex_parameters.json")
```

Arguments

`jsonFileName`	path to json files with setups for the solver

Value

full list of options

`runCARNIVAL` `runCARNIVAL`

Description

`runCARNIVAL`

Usage

```
runCARNIVAL(
  inputObj = NULL,
  measObj = measObj,
  netObj = netObj,
  weightObj = NULL,
  solverPath = NULL,
  solver = c("lpSolve", "cplex", "cbc", "gurobi"),
  timelimit = 3600,
  mipGAP = 0.05,
  poolrelGAP = 1e-04,
  limitPop = 500,
  poolCap = 100,
  poolIntensity = 4,
  poolReplace = 2,
```

```
  alphaWeight = 1,
  betaWeight = 0.2,
  threads = 0,
  cleanTmpFiles = TRUE,
  keepLPFiles = TRUE,
  clonelog = -1,
  dir_name = getwd()
)
```

Arguments

inputObj	Data frame of the list for target of perturbation - optional or default set to NULL to run invCARNIVAL when inputs are not known.
measObj	Data frame of the measurement file (i.e. DoRothEA normalised enrichment scores) - always required.
netObj	Data frame of the prior knowledge network - always required.
weightObj	Data frame of the additional weight (i.e. PROGENy pathway score or measured protein activities) - optional or default set as NULL to run CARNIVAL without weights.
solverPath	Path to executable cbc/cplex file - default set to NULL, in which case the solver from lpSolve package is used.
solver	Solver to use: lpSolve/cplex/cbc (Default set to lpSolve).
timelimit	CPLEX/Cbc parameter: Time limit of CPLEX optimisation in seconds (default set to 3600).
mipGAP	CPLEX parameter: the absolute tolerance on the gap between the best integer objective and the objective of the best node remaining. When this difference falls below the value of this parameter, the linear integer optimization is stopped (default set to 0.05)
poolrelGAP	CPLEX/Cbc parameter: Allowed relative gap of accepted solution comparing within the pool of accepted solution (default: 0.0001)
limitPop	CPLEX parameter: Allowed number of solutions to be generated (default: 500)
poolCap	CPLEX parameter: Allowed number of solution to be kept in the pool of solution (default: 100)
poolIntensity	CPLEX parameter: Intensity of solution searching (0,1,2,3,4 - default: 4)
poolReplace	CPLEX parameter: Replacement strategy of solutions in the pool (0,1,2 - default: 2 = most diversified solutions)
alphaWeight	Objective function: weight for mismatch penalty (default: 1 - will only be applied once measurement file only contains discrete values)
betaWeight	Objective function: weight for node penalty (default: 0.2)
threads	CPLEX/CBC parameter: Number of threads to use default: 0 for maximum number possible threads on system
cleanTmpFiles	logic (default-TRUE), specifying if the tmp files made by solvers should be cleaned after run.
keepLPFiles	logic (default=TRUE), specifying if the LP file should be kept.

clonelog	determines if CPLEX clones the log files in case of multi-threaded optimization, default: -1, (no cloning)
dir_name	Specify directory name to store results. by default set to NULL

Details

Run CARNIVAL pipeline using to the user-provided list of inputs or run CARNIVAL built-in examples. The function is from v1.2 of CARNIVAL and is left for backward compatibility.

Value

The function will return a list of results containing:

1. weightedSIF: A table with 4 columns containing the combined network solutions from CARNIVAL. It contains the Source of the interaction (Node1), Sign of the interaction (Sign), the Target of the interaction (Node2) and the weight of the interaction (Weight) which shows how often an interaction appears across all solutions.

2. nodesAttributes: A table with 6 columns containing information about infered protein activity states and attributes. It contains the Protein IDs (Node); how often this node has taken an activity of 0, 1 and -1 across the solutions (ZeroAct, UpAct, DownAct); the average activities across solutions (AvgAct); and the node attribute (measured, target, inferred).

3. sifAll: A list of separate network solutions.

4. attributesAll: A list of separate inferred node activities in each solution.

5. diagnostics: reports the convergence of optimization and reason of the termination. Only for CPLEX solver.

Author(s)

Enio Gjerga, 2020 <carnival.developers@gmail.com>

Examples

```
load(file = system.file("toy_perturbations_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_measurements_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_network_ex1.RData",
                        package="CARNIVAL"))

## lpSolve
#res1 = runCARNIVAL(inputObj = toy_perturbations_ex1,
#                   measObj = toy_measurements_ex1,
#                   netObj = toy_network_ex1,
#                   solver = 'lpSolve')

#res1$weightedSIF ##see @return
#res1$nodesAttributes ## see @return
#res1$sifAll ## see @return
#res1$attributesAll ## see @return
```

```
## Examples for cbc and cplex are commented out because these solvers are not part of R environment
## and need to be installed separately
##
## cbc
## res2 = runCARNIVAL(inputObj = toy_perturbations_ex1,
##                    measObj = toy_measurements_ex1,
##                    netObj = toy_network_ex1,
##                    solver = 'cbc')
##
## res2$weightedSIF ##see @return
## res2$nodesAttributes ## see @return
## res2$sifAll ## see @return
## res2$attributesAll ## see @return
##
## cplex
## res3 = runCARNIVAL(inputObj = toy_perturbations_ex1,
##                    measObj = toy_measurements_ex1,
##                    netObj = toy_network_ex1,
##                    solver = 'cplex')
##
## res3$weightedSIF ##see @return
## res3$nodesAttributes ## see @return
## res3$sifAll ## see @return
## res3$attributesAll ## see @return
```

`runFromLpCarnival` `runCarnivalFromLp`

Description

`runCarnivalFromLp`

Usage

```
runFromLpCarnival(
  lpFile = "",
  parsedDataFile = "",
  carnivalOptions = defaultLpSolveCarnivalOptions()
)
```

Arguments

`lpFile`	full path to .lp file
`parsedDataFile`	full path to preprocessed .RData file
`carnivalOptions`	the list of options for the run. See defaultLpSolveCarnivalOptions(), defaultLpSolveCarnivalOptions, defaultCbcCarnivalOptions.

Details

Runs CARNIVAL pipeline with preparsed data - lp file and Rdata file containing variables for ILP formulation.

Value

The function will return a list of results containing: 1. weightedSIF: A table with 4 columns containing the combined network solutions from CARNIVAL. It contains the Source of the interaction (Node1), Sign of the interaction (Sign), the Target of the interaction (Node2) and the weight of the interaction (Weight) which shows how often an interaction appears across all solutions.

2. nodesAttributes: A table with 6 columns containing information about infered protein activity states and attributes. It contains the Protein IDs (Node); how often this node has taken an activity of 0, 1 and -1 across the solutions (ZeroAct, UpAct, DownAct); the average activities across solutions (AvgAct); and the node attribute (measured, target, inferred).

3. sifAll: A list of separate network solutions.

4. attributesAll: A list of separate inferred node activities in each solution.

5. diagnostics: reports the convergence of optimization and reason of the termination. Only for CPLEX solver.

Author(s)

Enio Gjerga, Olga Ivanova 2020-2021 <carnival.developers@gmail.com>

Examples

```
lpFilePath = system.file("toy_lp_file_ex1.lp",
                         package="CARNIVAL")

parsedDataFilePath = system.file("toy_parsed_data_ex1.RData",
                                 package="CARNIVAL")

## lpSolve
#res1 = runFromLpCarnival(lpFile = lpFilePath,
#                         parsedDataFile = parsedDataFilePath,
#                         carnivalOptions = defaultLpSolveCarnivalOptions())

#res1$weightedSIF ##see @return
#res1$nodesAttributes ## see @return
#res1$sifAll ## see @return
#res1$attributesAll ## see @return

## Examples for cbc and cplex are commented out because these solvers are not part of R environment
## and need to be installed separately
##
## cbc
## res2 = runFromLpCarnival(lpFile = lpFilePath,
##                          parsedDataFile = parsedDataFilePath,
##                          carnivalOptions = defaultLpCbcCarnivalOptions())
##
```

```
## res2$weightedSIF ##see @return
## res2$nodesAttributes ## see @return
## res2$sifAll ## see @return
## res2$attributesAll ## see @return
##
## cplex
## res3 = runFromLpCarnival(lpFile = lpFilePath,
##                          parsedDataFile = parsedDataFilePath,
##                          carnivalOptions = defaultLpCplexCarnivalOptions())
##
## res3$weightedSIF ##see @return
## res3$nodesAttributes ## see @return
## res3$sifAll ## see @return
## res3$attributesAll ## see @return
```

`runInverseCarnival` `runInverseCarnival`

Description

`runInverseCarnival`

Usage

```
runInverseCarnival(
  measurements,
  priorKnowledgeNetwork,
  weights = NULL,
  carnivalOptions = defaultLpSolveCarnivalOptions()
)
```

Arguments

`measurements`	vector of the measurements (i.e. DoRothEA/VIPER normalised enrichment scores)
`priorKnowledgeNetwork`	data frame of the prior knowledge network
`weights`	(optional) vector of the additional weights: e.g. PROGENy pathway score or measured protein activities.
`carnivalOptions`	the list of options for the run. See defaultLpSolveCarnivalOptions(), defaultLpSolveCarnivalOptions, defaultCbcCarnivalOptions.

Details

TODO Replace with correct description

Value

The function will return a list of results containing:

1. weightedSIF: A table with 4 columns containing the combined network solutions from CARNIVAL. It contains the Source of the interaction (Node1), Sign of the interaction (Sign), the Target of the interaction (Node2) and the weight of the interaction (Weight) which shows how often an interaction appears across all solutions.

2. nodesAttributes: A table with 6 columns containing information about infered protein activity states and attributes. It contains the Protein IDs (Node); how often this node has taken an activity of 0, 1 and -1 across the solutions (ZeroAct, UpAct, DownAct); the average activities across solutions (AvgAct); and the node attribute (measured, target, inferred).

3. sifAll: A list of separate network solutions.

4. attributesAll: A list of separate inferred node activities in each solution.

5. diagnostics: reports the convergence of optimization and reason of the termination. Only for CPLEX solver.

Author(s)

Enio Gjerga, Olga Ivanova 2020-2021 <carnival.developers@gmail.com>

Examples

```
load(file = system.file("toy_measurements_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_network_ex1.RData",
                        package="CARNIVAL"))

## lpSolve
#res1 = runInverseCarnival(measurements = toy_measurements_ex1,
#                          priorKnowledgeNetwork = toy_network_ex1,
#                          carnivalOptions = defaultLpSolveCarnivalOptions())

#res1$weightedSIF ##see @return
#res1$nodesAttributes ## see @return
#res1$sifAll ## see @return
#res1$attributesAll ## see @return

## Examples for cbc and cplex are commented out because these solvers are not part of R environment
## and need to be installed separately
##
## cbc
## res2 = runInverseCarnival(measurements = toy_measurements_ex1,
##                           priorKnowledgeNetwork = toy_network_ex1,
##                           carnivalOptions = defaultCbcCarnivalOptions())
##
## res2$weightedSIF ##see @return
## res2$nodesAttributes ## see @return
## res2$sifAll ## see @return
## res2$attributesAll ## see @return
##
```

```
## cplex
## res3 = runVanillaCarnival(measurements = toy_measurements_ex1,
##                           priorKnowledgeNetwork = toy_network_ex1,
##                           carnivalOptions = defaultCplexCarnivalOptions())
##
## res3$weightedSIF ##see @return
## res3$nodesAttributes ## see @return
## res3$sifAll ## see @return
## res3$attributesAll ## see @return
```

---

runVanillaCarnival runVanillaCarnival

Description

runVanillaCarnival

Usage

```
runVanillaCarnival(
  perturbations,
  measurements,
  priorKnowledgeNetwork,
  weights = NULL,
  carnivalOptions = defaultLpSolveCarnivalOptions()
)
```

Arguments

perturbations	vector of targets of perturbations.
measurements	vector of the measurements (i.e. DoRothEA/VIPER normalised enrichment scores)
priorKnowledgeNetwork	data frame of the prior knowledge network
weights	(optional) vector of the additional weights: e.g. PROGENy pathway score or measured protein activities.
carnivalOptions	the list of options for the run. See defaultLpSolveCarnivalOptions(), defaultLpSolveCarnivalOptions, defaultCbcCarnivalOptions.

Details

Runs full CARNIVAL pipeline, vanilla(classic) flavour.

Value

The function will return a list of results containing: 1. weightedSIF: A table with 4 columns containing the combined network solutions from CARNIVAL. It contains the Source of the interaction (Node1), Sign of the interaction (Sign), the Target of the interaction (Node2) and the weight of the interaction (Weight) which shows how often an interaction appears across all solutions.

2. nodesAttributes: A table with 6 columns containing information about infered protein activity states and attributes. It contains the Protein IDs (Node); how often this node has taken an activity of 0, 1 and -1 across the solutions (ZeroAct, UpAct, DownAct); the average activities across solutions (AvgAct); and the node attribute (measured, target, inferred).

3. sifAll: A list of separate network solutions.

4. attributesAll: A list of separate inferred node activities in each solution.

5. diagnostics: reports the convergence of optimization and reason of the termination. Only for CPLEX solver.

Author(s)

Enio Gjerga, Olga Ivanova 2020-2021 <carnival.developers@gmail.com>

Examples

```
load(file = system.file("toy_perturbations_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_measurements_ex1.RData",
                        package="CARNIVAL"))
load(file = system.file("toy_network_ex1.RData",
                        package="CARNIVAL"))

## lpSolve
#res1 = runVanillaCarnival(perturbations = toy_perturbations_ex1,
#                          measurements = toy_measurements_ex1,
#                          priorKnowledgeNetwork = toy_network_ex1,
#                          carnivalOptions = defaultLpSolveCarnivalOptions())

#res1$weightedSIF ##see @return
#res1$nodesAttributes ## see @return
#res1$sifAll ## see @return
#res1$attributesAll ## see @return

## Examples for cbc and cplex are commented out because these solvers are not part of R environment
## and need to be installed separately
##
## cbc
## res2 = runVanillaCarnival(perturbations = toy_perturbations_ex1,
##                                measurements = toy_measurements_ex1,
##                                priorKnowledgeNetwork = toy_network_ex1,
##                                carnivalOptions = defaultCbcCarnivalOptions())
##
## res2$weightedSIF ##see @return
## res2$nodesAttributes ## see @return
## res2$sifAll ## see @return
```

```
## res2$attributesAll ## see @return
##
## cplex
## res3 = runVanillaCarnival(perturbations = toy_perturbations_ex1,
##                           measurements = toy_measurements_ex1,
##                           priorKnowledgeNetwork = toy_network_ex1,
##                           carnivalOptions = defaultCplexCarnivalOptions())
##
## res3$weightedSIF ##see @return
## res3$nodesAttributes ## see @return
## res3$sifAll ## see @return
## res3$attributesAll ## see @return
```

`sendTaskToSolver` *Executes the solve on the provided ILP formulation (in .lp file).*

Description

Executes the solve on the provided ILP formulation (in .lp file).

Usage

```
sendTaskToSolver(
  variables,
  dataPreprocessed,
  carnivalOptions,
  newDataRepresentation = TRUE
)
```

Arguments

`variables`	list of nodes, edges and measurements variables generated by createLpFormulation_v2.
`dataPreprocessed`	list containing preprocessed priorKnowledgeNetwork, measurements, weights (if provided), perturbations (if provided).
`carnivalOptions`	all options of CARNIVAL.
`newDataRepresentation`	TRUE by default. For debugging with the old data representation, put to FALSE.

Value

solution matrix from ILP solver containing variables list (rows) and their values in different solutions (columns).

`setCarnivalOptions` *Sets CARNIVAL options for the solver.*

Description

Sets CARNIVAL options for the solver.

Usage

```
setCarnivalOptions(solver = getSupportedSolvers()$lpSolve, ...)
```

Arguments

`solver`	one of the solvers available from getSupportedSolvers().
`...`	any possible options from the solver's list

Value

carnival options as list.

Examples

```
setCarnivalOptions(solver="lpSolve")
```

`solveCarnival` *Main CARNIVAL function to execute the full pipeline: 1) preprocess the data 2) prepare ILP formulation 3) executes the solver on ILP formulation 4) parse the output of the solver and map it to the original data.*

Description

Main CARNIVAL function to execute the full pipeline: 1) preprocess the data 2) prepare ILP formulation 3) executes the solver on ILP formulation 4) parse the output of the solver and map it to the original data.

Usage

```
solveCarnival(dataPreprocessed, carnivalOptions, newDataRepresentation = TRUE)
```

Arguments

`dataPreprocessed`
: list containing preprocessed priorKnowledgeNetwork, measurements, weights (if provided), perturbations (if provided).

`carnivalOptions`
: all options of CARNIVAL.

`newDataRepresentation`
: TRUE by default. For debugging with the old data representation, put to FALSE.

Value

solution of the ILP problem.

`solveCarnivalFromLp` *Sends the ILP formulation defined in .lp file to solver. Uses parsedDataFile to process the final solution and map the ILP variables back to inital data.*

Description

Sends the ILP formulation defined in .lp file to solver. Uses parsedDataFile to process the final solution and map the ILP variables back to inital data.

Usage

```
solveCarnivalFromLp(
  lpFile = "",
  parsedDataFile = "",
  carnivalOptions,
  newDataRepresentation = TRUE
)
```

Arguments

`lpFile`
: path to .lp file that will be used to run the solver.

`parsedDataFile`
: path to parsed data file that was created after running `prepareForCarnivalRun` or in previous CARNIVAL runs.

`carnivalOptions`
: all options of CARNIVAL.

`newDataRepresentation`
: TRUE by default. For debugging with the old data representation, put to FALSE.

Value

solution of ILP problem

`solveWithCbc` *Executes cbc solver on provided .lp file.*

Description

Executes cbc solver on provided .lp file.

Usage

```
solveWithCbc(carnivalOptions)
```

Arguments

`carnivalOptions`

Value

returns optimized variables in a solution matrix from CBC

`solveWithGurobi` *Executes gurobi solver on provided .lp file.*

Description

Executes gurobi solver on provided .lp file.

Usage

```
solveWithGurobi(carnivalOptions)
```

Arguments

`carnivalOptions`

Value

Returns the name of the result files without ".sol" extension.

suggestedCbcSpecificOptions

Suggests cbc specific options.

Description

Suggests cbc specific options.

Usage

```
suggestedCbcSpecificOptions(...)
```

Arguments

...	any possible options from the solver's list

Value

additional CbC solver options as a list.

Examples

```
suggestedCbcSpecificOptions()
```

suggestedCplexSpecificOptions

Suggests cplex specific options.s

Description

Suggests cplex specific options.s

Usage

```
suggestedCplexSpecificOptions(...)
```

Arguments

...	any possible options from the solver's list

Value

additional CPLEX solver options as a list.

Examples

```
suggestedCplexSpecificOptions()
```

writeCplexCommandFileFromJson *writeCplexCommandFileFromJson*

Description

writeCplexCommandFileFromJson

Usage

```
writeCplexCommandFileFromJson(
  carnivalOptions,
  jsonFileName = "parameters/cplex_parameters_cmd_file.json"
)
```

Arguments

carnivalOptions	list of options for the CPLEX solver
jsonFileName	name to JSONfile containing the solver parameters

Value

list of params

writeParsedData *Saves all provided data together with generated variables for ILP problem in .RData file.*

Description

Saves all provided data together with generated variables for ILP problem in .RData file.

Usage

```
writeParsedData(
  variables = variables,
  dataPreprocessed = dataPreprocessed,
  filename = "parsedData.RData"
)
```

Arguments

`variables`	list of nodes, edges and measurements variables generated by createLpFormulation_v2
`dataPreprocessed`	list containing preprocessed priorKnowledgeNetwork, measurements, weights (if provided), perturbations (if provided).
`filename`	filename of the parsed data file.

Value

filename of the parsed data file.

Index

* **=**
    readOptions, 15
* **#TODO**
    readOptions, 15
* **#examples**
    readOptions, 15
* **#readOptions(jsonFileName**
    readOptions, 15
* **inst/carnival_cplex_parameters.json)**
    readOptions, 15
* **internal**
    addPerturbationNodes, 3
    checkCarnivalOptions, 3
    checkData, 4
    checkPriorKnowledgeNetwork, 5
    createInternalDataRepresentation, 6
    getSupportedSolversFunctions, 11
    prepareForCarnivalRun, 13
    preprocessPriorKnowledgeNetwork, 13
    processSolution, 14
    readOptions, 15
    sendTaskToSolver, 24
    solveCarnival, 25
    solveCarnivalFromLp, 26
    solveWithCbc, 27
    solveWithGurobi, 27
    writeParsedData, 29

addPerturbationNodes, 3

checkCarnivalOptions, 3
checkData, 4
checkOptionsValidity, 4
checkPriorKnowledgeNetwork, 5
createInternalDataRepresentation, 6

defaultCbcSolveCarnivalOptions, 6
defaultCplexCarnivalOptions, 7
defaultCplexSpecificOptions, 7
defaultLpSolveCarnivalOptions, 8

generateLpFileCarnival, 8
getOptionsList, 10
getSupportedSolvers, 10
getSupportedSolversFunctions, 11

isInputValidCarnival, 11

parseCplexLog, 12
prepareForCarnivalRun, 13, *26*
preprocessPriorKnowledgeNetwork, 13
processSolution, 14

read_lines, *12*
readOptions, 15
runCARNIVAL, 15
runFromLpCarnival, 18
runInverseCarnival, 20
runVanillaCarnival, 22

sendTaskToSolver, 24
setCarnivalOptions, 25
solveCarnival, 25
solveCarnivalFromLp, 26
solveWithCbc, 27
solveWithGurobi, 27
suggestedCbcSpecificOptions, 28
suggestedCplexSpecificOptions, 28

writeCplexCommandFileFromJson, 29
writeParsedData, 29