

# Package: CARNIVAL (via r-universe)

May 29, 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

**Config/pak/sysreqs** cmake libglpk-dev make libicu-dev libuv1-dev libxml2-dev libx11-dev

**Repository** <https://bioc-release.r-universe.dev>

**Date/Publication** 2026-04-28 12:52:37 UTC

**RemoteUrl** <https://github.com/bioc/CARNIVAL>

**RemoteRef** RELEASE\_3\_23

**RemoteSha** 606bffa9b983b3ab86d5383295d8fb9d86f1d283

## Contents

checkOptionsValidity . . . . .	2
defaultCbcSolveCarnivalOptions . . . . .	3
defaultCplexCarnivalOptions . . . . .	3
defaultCplexSpecificOptions . . . . .	4
defaultLpSolveCarnivalOptions . . . . .	4
generateLpFileCarnival . . . . .	5
getOptionsList . . . . .	6
getSupportedSolvers . . . . .	7
isInputValidCarnival . . . . .	7
parseCplexLog . . . . .	8
runCARNIVAL . . . . .	9
runFromLpCarnival . . . . .	12
runInverseCarnival . . . . .	13
runVanillaCarnival . . . . .	15
setCarnivalOptions . . . . .	17
suggestedCbcSpecificOptions . . . . .	18
suggestedCplexSpecificOptions . . . . .	19
writeCplexCommandFileFromJson . . . . .	19
<b>Index</b>	<b>20</b>

---

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")
```

---

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: transforms 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 <code>getSupportedSolvers()</code>
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.

---

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

---

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 inferred 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 prepared 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 inferred 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 inferred 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 inferred 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

```

---

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")
```

---

`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

# Index

[checkOptionsValidity](#), 2

[defaultCbcSolveCarnivalOptions](#), 3

[defaultCplexCarnivalOptions](#), 3

[defaultCplexSpecificOptions](#), 4

[defaultLpSolveCarnivalOptions](#), 4

[generateLpFileCarnival](#), 5

[getOptionsList](#), 6

[getSupportedSolvers](#), 7

[isInputValidCarnival](#), 7

[parseCplexLog](#), 8

[read\\_lines](#), 8

[runCARNIVAL](#), 9

[runFromLpCarnival](#), 12

[runInverseCarnival](#), 13

[runVanillaCarnival](#), 15

[setCarnivalOptions](#), 17

[suggestedCbcSpecificOptions](#), 18

[suggestedCplexSpecificOptions](#), 19

[writeCplexCommandFileFromJson](#), 19