Package 'Rpolyhedra'

July 10, 2018

Type Package

Title Polyhedra Database

Version 0.2.6

Maintainer Alejandro Baranek <abaranek@dc.uba.ar>

Description
A polyhedra database scraped from various sources as R6 objects and 'rgl' visualizing capabilities.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown, covr

VignetteBuilder knitr

Depends R (>= 3.0.0)

Imports futile.logger, rgl, stringr, R6, testthat, XML

Collate 'xml-lib.R' 'polyhedra-lib.R' 'ledger-lib.R' 'db-lib.R'
'zzz.R'

BugReports https://github.com/qbotics/Rpolyhedra/issues

NeedsCompilation no

Author Alejandro Baranek [aut, com, cre, cph],
Leonardo Belen [aut, com, cph]

Repository CRAN

Date/Publication 2018-07-10 16:50:03 UTC

R topics documented:

checkDatabaseVersion *checkDatabaseVersion*

Description

Determines if there is a need for a database update by checking the version file of both the package and the database installation.

Usage

```
checkDatabaseVersion()
```

Value

"UPDATE" if an update is required, "NO_ACTION_REQUIRED" otherwise.

checkVertices *checkVertices()*

Description

Check rendering vertices properties

Usage

```
checkVertices(vertices, positioned.vertices, triangulated.solid)
```

Arguments

vertices	vertices dataframe for checking
positioned.vertices	positioned vertices dataframe for checking
triangulated.solid	triangulated.solid for checking

Value

checked positioned vertices

`copyFilesToExtData` *copyFilesToExtData*

Description

Copies files from the home directory to the package one to build a preconfigured package.

Usage

```
copyFilesToExtData(force = FALSE)
```

Arguments

`force`	indicate if existings directories must be overwritten

Value

TRUE if sucessfull

`downloadRPolyhedraSupportingFiles` *downloadRPolyhedraSupportingFiles*

Description

Downloads the files from the remote location

Usage

```
downloadRPolyhedraSupportingFiles()
```

Value

TRUE if sucessfull, FALSE otherwise

getAvailablePolyhedra *getAvailablePolyhedra()*

Description

Gets the list of names of available polyhedra and its status in the polyhedra database, which can be later called with getPolyhedron

Usage

```
getAvailablePolyhedra(sources, search.string)
```

Arguments

sources	A source of polyhedra. Available sources are netlib, dmccooey
search.string	A search string

Value

polyhedra names vector

Examples

```
#gets all polyhedra in the database
available.polyhedra <- getAvailablePolyhedra()

#returns all polyhedra from a given source, in this case, netlib
available.netlib.polyhedra <- getAvailablePolyhedra(sources="netlib")

#search within the polyhedron names

cube <- getAvailablePolyhedra(sources="netlib",search.string="cube")
```

getAvailableSources *getAvailableSources()*

Description

Gets the list of names of available sources in database

Usage

```
getAvailableSources()
```

Value

sources string vector

Examples

```
#gets all sources in the database
available.sources <- getAvailableSources()

#returns all polyhedra from all sources
available.polyhedra <- getAvailablePolyhedra(sources=available.sources)

#search within the polyhedron names from all sources
cubes <- getAvailablePolyhedra(sources=available.sources,search.string="cube")
```

---

getDatabaseVersion *getDatabaseVersion*

Description

Obtains the generation code version from the database version file

Usage

```
getDatabaseVersion()
```

---

getDataDir *getDataDir*

Description

Gets the path of Rpolyhedra data dir.

Usage

```
getDataDir(data.env = getDataEnv())
```

Arguments

data.env	enviroment where data directory must be returned

Details

This function is used internally to determine whether the package is compiled in source or package directory.

Value

dir where the package access polyhedra database

`getDataEnv` *getDataEnv*

Description

Gets the current .data.env value

Usage

```
getDataEnv()
```

Value

.data.env

`getEnvironmentFilepath` *getEnvironmentFilepath*

Description

Gets the filename where package data environment is persisted

Usage

```
getEnvironmentFilepath()
```

Value

The environment filepath

`getGitCommit` *getGitCommit get the last git commit sha*

Description

getGitCommit get the last git commit sha

Usage

```
getGitCommit()
```

Value

String with git commit sha

getPackageDB *getPackageDB*

Description

Obtains the database version from environment

Usage

```
getPackageDB()
```

getPackageDir *getPackageDir*

Description

Gets the path of package data.

Usage

```
getPackageDir()
```

getPackageEnvir *getPackageEnvir*

Description

getPackageEnvir

Usage

```
getPackageEnvir(variable.name)
```

Arguments

variable.name	name of variable to be retrieved Obtains a variable from package environment

getPackageVersion *getPackageVersion*

Description

Obtains code version from the Description

Usage

```
getPackageVersion()
```

getPercentilPolyhedraQuant *getPercentilPolyhedraQuant returns polyhedra quantity of parameter percentil*

Description

getPercentilPolyhedraQuant returns polyhedra quantity of parameter percentil

Usage

```
getPercentilPolyhedraQuant(percentil, quant.min = 100)
```

Arguments

percentil	is the percentil which must be applied to estimate the figure
quant.min	minimum quantity of files to return

getPolyhedraObject *getPolyhedraObject*

Description

Gets the polyhedra object

Usage

```
getPolyhedraObject()
```

Value

.polyhedra

getPolyhedraRDSPath *getPolyhedraRDSPath*

Description

Gets the path of Polyhedra RDS database file

Usage

```
getPolyhedraRDSPath(polyhedra_rds_filename = "polyhedra.RDS")
```

Arguments

polyhedra_rds_filename	filename of polyhedra database

Value

the path to the Polyhedra database file

getPolyhedron *getPolyhedron()*

Description

Gets a polyhedron from the database. It returns an R6 Class with all its characteristics and functions. The object returned, of type Polyhedron.class, allows to the user to get access to all the functionality provided.

Usage

```
getPolyhedron(source = "netlib", polyhedron.name)
```

Arguments

source	source name
polyhedron.name	a valid name of a polyhedron in the database. Current names can be found with getAvailablePolyhedra()

Value

polyhedron R6 object

Examples

```
tetrahedron <- getPolyhedron(source="netlib", polyhedron.name = 'tetrahedron')

# returns name of polyhedra
tetrahedron$getName()

# polyhedron state
tetrahedron.state <- tetrahedron$getState()

# Johnson symbol and Schlafli symbol
tetrahedron.state$getSymbol()

# vertex data.frame
tetrahedron.state$getVertices()

# List of faces of solid representation (3D)
tetrahedron.state$getSolid()

# List of faces of net representation (2D)
tetrahedron.state$getNet()
```

`getPreloadedDataFilename` *getPreloadedDataFilename*

Description

Gets the path of Polyhedra preloaded data CSV file

Usage

```
getPreloadedDataFilename(polyhedra_preloaded_data = "polyhedra.preloaded.data.csv")
```

Arguments

`polyhedra_preloaded_data`	filename of polyhedra preloaded data csv

Value

the path to the Polyhedra database file

`getUserEnvir` *getUserEnvir*

Description

Gets a writable environment for the package

Usage

```
getUserEnvir(variable.name)
```

Arguments

`variable.name`	name of variable to be retrieved

`getUserSpace` *getUserSpace*

Description

This function is used internally for accesing the local database path

Usage

```
getUserSpace()
```

Value

path of user space

`initDataDirEnvironment` *initDataDirEnvironment*

Description

initialize data enviornment

Usage

```
initDataDirEnvironment()
```

Value

the data dir environment

`isCompatiblePolyhedraRDS` *isCompatiblePolyhedraRDS()*

Description

Tests if the polyhedra RDS is compatible with the current format

Usage

```
isCompatiblePolyhedraRDS(.polyhedra.candidate = getPolyhedraObject(),
  halts = FALSE)
```

Arguments

`.polyhedra.candidate`	polyhedra db to test
`halts`	indicates whether it has to halt execution when it is not compatible

`maxWithoutNA` *maxWithoutNA Function that returns NA if all elements are NA, and the max value not NA, if not.*

Description

maxWithoutNA Function that returns NA if all elements are NA, and the max value not NA, if not.

Usage

```
maxWithoutNA(x)
```

Arguments

`x`	vector parameter

norm *norm calculates norm of a vector*

Description

norm calculates norm of a vector

Usage

```
norm(vector)
```

Arguments

vector	numeric vector

PolyhedraDatabase.class *PolyhedraDatabase*

Description

Scrapes all polyhedra in data folder to save a representation which is accesible by the final users upon call to `getPolyhedron()`.

Usage

```
PolyhedraDatabase.class
```

Format

R6Class object.

Fields

polyhedra.rds.file	path of rds database file
sources.config	Sources configuration for scraping different sources
ledger	rr ledger of scraping process
data	Polyhedra data from different sources

Methods

`initialize()` Initializes the object

`existsSource(source)` Determines if the source exists on the database

`getSource(source, strict=False)` Retrieves a source by name

`addSource(source)` Adds a new source to the database

`configPolyhedraRDSPath()` config path for rds database file

`existsPolyhedron(source,polyhedron.name)` Determines if the polyhedron exists on the database

`getPolyhedron(source, polyhedron.name, strict)` Retrieves a polyhedron by source and name

`addPolyhedron(source,polyhedron,overwrite)` Adds a polyhedron by source and name, if orverwrite is TRUE, it will update any existing one by that source and name

`configPolyhedraSource(source.config, max.quant)` Scrapes all polyhedra in the given directory for adding to db or testing

`schedulePolyhedraSources(sources.config,max.quant, test)` Scrapes files applying parameter sources.config

`getAvailablePolyhedra(sources,search.string)` Retrieves all polyhedron within the source those names match with search.string

---

`Polyhedron.class` *Polyhedron*

Description

Polyhedron container class, which is accesible by the final users upon call to `getPolyhedron()`

Usage

```
Polyhedron.class
```

Format

`R6Class` object.

Fields

`number` Polyhedron number

`state` Polyhedron state

Methods

`initialize(number, state = NULL)` Initializes the object

`scrapeNetlib(polyhedron.lines)` Scrapes polyhedra from the definition

`getName()` Gets the name from polyhedron definition

`getState()` Gets the state from polyhedron definition

`getSolid()` Gets the solid definition of polyhedron definition

`isChecked()` Returns TRUE is polyhedron is checked

`getErrors()` Returns errors collected in checking process

`getRGLModel(size = 1, origin = c(0, 0, 0)` Builds the RGL model

`exportToXML()` Gets an XML representation out of the polyhedron object

`checkProperties(expected.vertices, expected.faces)` check polyhedron basic properties

PolyhedronScraperConfiguration.class

PolyhedronScraperConfiguration

Description

Abstract class for configuring specific scrapers for Diferent sources

Usage

```
PolyhedronScraperConfiguration.class
```

Format

An object of class `R6ClassGenerator` of length 24.

Methods

`initialize()` Initializes the object

PolyhedronScraperConfigurationDmccoey.class
PolyhedronScraperConfigurationDmccoey

Description

Scraper configuration for Dmccoey source

Usage

```
PolyhedronScraperConfigurationDmccoey.class
```

Format

R6Class object.

Methods

`initialize()` initializes the object

`getPolyhedraFiles(home.dir.data)` returns the file names on the netlib database

`scrape(polyhedron.number, polyhedron.filename)` scrapes the object

PolyhedronScraperConfigurationNetlib.class
PolyhedronScraperConfigurationNetlib

Description

Scraper configuration for Netlib source

Usage

```
PolyhedronScraperConfigurationNetlib.class
```

Format

R6Class object.

Methods

`initialize()` initializes the object

`getPolyhedraFiles(home.dir.data)` returns the file names on the netlib database

`scrape(polyhedron.number, polyhedron.filename)` scrapes the object

PolyhedronState.class *Polyhedron State*

Description

This abstract class provide the basis from which polyhedron state class derivate.

Usage

```
PolyhedronState.class
```

Format

R6Class object.

Fields

`errors` Errors string

`source` polyhedron definition source

`number` polyhedron number

Methods

`addError(current.error)` Adds an error to the error string and log it as info

`scrape()` Scrapes the polyhedra folder files

`geSolid()` returns the object corresponding to the solid

`buildRGL(size = 1, origin = c(0, 0, 0), normalize.size = TRUE)` creates a RGL representation of the object

`exportToXML()` Gets an XML representation out of the polyhedron object

PolyhedronStateDefined.class *Polyhedron State Defined*

Description

Polyhedron state inside database.

Usage

```
PolyhedronStateDefined.class
```

Format

R6Class object.

Fields

`source` polyhedron definition source

`number` polyhedron arbitrary numeration (netlib|dmccooey)

`name` polyhedron name (netlib|dmccooey)

`symbol` the eqn(1) input for two symbols separated by a tab; the Johnson symbol, and the Schlafli symbol (netlib)

`dual` the name of the dual polyhedron optionally followed by a horizontal tab and the number of the dual (netlib)

`sfaces` polyhedron solid face list (netlib)

`svertices` polyhedron solid vertice list (netlib)

`net` polyhedron 2D net model with vertices defined for a planar representation (netlib)

`hinges` Polyhedron hinge list (netlib)

`solid` polyhedron list of edges which generate a solid (netlib|dmccooey)

`dih` Dih attribute (netlib)

`vertices` Polyhedron vertices list (netlib|dmccooey)

`vertices.rgl` Polyhedron triangulated vertices list for RGL

`solid.triangulated` Polyhedron solid (triangulated) for RGL visualization

`mass.center` polyhedron mass center

Methods

`initialize(source, number,name, symbol, dual, sfaces, svertices, net, solid, hinges, dih, vertices)` Initializes the object, taking defaults.

`scrape()` Do nothing as the object is defined

`getNet()` Gets the 2d net model

`getSolid()` Gets the solid representation

`triangulate(force = FALSE)` Generates the triangular faces model for generating tmesh

`getBoundingBox(vertices.3d)` Gets the bounding box of the object

`calculateMassCenter(size = 1, vertices.3d)` Calculates the object's Mass Center for parameter vertices

`getNormalizedSize()` Normalizes the volume of the object to a tetrahedron bounding box#'

`getPositionedVertices(size,origin)` Returns the vertices adjusted to size and origin parameters#'

`buildRGL(size = 1, origin = c(0, 0, 0), normalize.size = TRUE)` Builds the RGL model, taking the object's size, the origin

`exportToXML()` Gets an XML representation out of the polyhedron object

`PolyhedronStateDmccoeyScraper.class`
Polyhedron State Dmccoey Scraper

Description

Scrapes polyhedra from a dmccooey file format

Usage

```
PolyhedronStateDmccoeyScraper.class
```

Format

`R6Class` object.

Methods

`initialize(number, netlib.p3.lines)` Initializes the object, taking the number and PDH file as parameters

`scrape()` Scrapes data from dmccooey file format

`scrapeValues(values.lines)` Scrapes values

`scrapeVertices(vertices.lines)` Scrapes vertices

`scrapeFaces(face.lines)` Scrapes faces

`buildRGL(size = 1, origin = c(0, 0, 0), normalize.size = TRUE)` Builds the `RGL` implementation

`PolyhedronStateNetlibScraper.class`
Polyhedron State Netlib Scraper

Description

Scrapes polyhedra from a PHD file format.

Usage

```
PolyhedronStateNetlibScraper.class
```

Format

`R6Class` object.

Fields

`netlib.p3.lines` The path to the PHD files

`labels.rows` Labels - row of appearance

`labels.map` Labels - Map of content

Methods

`initialize(number, netlib.p3.lines)` Initializes the object, taking the number and PDH file as parameters

`extract_fows_from_label(label.number, expected.label)` Extracts data from the label, taking the label number and the expected label as parameters

`getLabels()` Gets the label from the polyhedron

`scrapeNet(net.txt, offset = 0)` Scrape the net model

`extractCFOutBrackets()` Gets the CF Out Brackets

`scrapeVertices(vertices.txt)` Scrapes the vertices

`setupLabelsOrder()` Sets up the order of labels included in PHD file

`getDataFromLabel(label)` Gets data from the Label

`scrape()` Scrapes the data from the PHD file

`buildRGL(size = 1, origin = c(0, 0, 0), normalize.size = TRUE)` Builds the `RGL` mmodel

`PolyhedronTestTask.class` *PolyhedronTestTask*

Description

Is an abstract class for specifying TestTask to make R6 iteration methods like cover complaint with testhat infrastructure

Usage

```
PolyhedronTestTask.class
```

Format

`R6Class` object.

Methods

`initialize()` initializes the object

`run()` Run the test task

PolyhedronTestTaskEdgesConsistency.class *PolyhedronTestTaskEdgesConsistency*

Description

A Test task for running edges consistency test for current polyhedron

Usage

```
PolyhedronTestTaskEdgesConsistency.class
```

Format

An object of class `R6ClassGenerator` of length 24.

PolyhedronTestTaskScrape.class *PolyhedronTestTaskScrape*

Description

A Test task for comparing a new scrape with an already scraped polyhedron in database

Usage

```
PolyhedronTestTaskScrape.class
```

Format

`R6Class` object.

Methods

`initialize()` initializes the object

`run()` Run the test task

`polyhedronToXML` *polyhedronToXML()*

Description

Gets an XML representation out of the polyhedron object

Usage

```
polyhedronToXML(polyhedron.state.defined)
```

Arguments

`polyhedron.state.defined`	the polyhedron to get a representation from

Value

an XML document, ready to be converted to String with XML::saveXML()

`Rpolyhedra` *Rpolyhedra*

Description

polyhedra database

Usage

```
.onLoad(libname, pkgname)
```

Arguments

`libname`	The library name
`pkgname`	The package name

Details

A polyhedra database scraped from: * http://paulbourke.net/dataformats/phd/: PHD files as R6 objects and 'rgl' visualizing capabilities. The PHD format was created to describe the geometric polyhedra definitions derived mathematically <http://www.netlib.org/polyhedra/> by Andrew Hume and by the Kaleido program of Zvi Har'El. * http://dmccooey.com/Polyhedra: Polyhedra text datafiles.

Author(s)

Alejandro Baranek <abaranek@dc.uba.ar>, Leonardo Javier Belen <leobelen@gmail.com> Executes code while loading the package.

`scrapePolyhedra` *scrapePolyhedra()*

Description

Gets polyhedra objects from text files of different sources, scheduling and scraping using predefined configurations

Usage

```
scrapePolyhedra(scrape.config,
  sources.config = getPackageEnvir(".available.sources"))
```

Arguments

`scrape.config`	predefined configuration for scraping
`sources.config`	the sources that will be used by the function

Value

polyhedra db object

`scrapePolyhedraSources` *scrapePolyhedraSources()*

Description

Method for obtaining polyhedra objects from text files of different sources, scheduling and scraping

Usage

```
scrapePolyhedraSources(sources.config = getPackageEnvir(".available.sources"),
    max.quant.config.schedule = 0,
    max.quant.scrape = 0, time2scrape.source = 30, retry.scrape = FALSE)
```

Arguments

`sources.config`	the sources that will be used by the function
`max.quant.config.schedule`	number of files to schedule
`max.quant.scrape`	number of files scrape
`time2scrape.source`	time applied to scrape source
`retry.scrape`	should it retry scrape?

Value

polyhedra db object

`ScraperLedger.class` *ScraperLedger*

Description

Ledger of scraping status of each objects. Allows different type of states: queued, scraping, scraped, failed, exception, skipped

Usage

```
ScraperLedger.class
```

Format

`R6Class` object.

Details

For flexible and reproducible configuration for package development

Methods

`initialize()` initializes the object

`addFilename(source, filename)` add filename to the ledger

`getIdFilename(source, filename)` Returns id/row of source and filenames parameters in the ledger

`updateStatus(source, filename, status, status.field = 'status', scraped.polyhedron = NA, obs ='')` Updates status of source and filenames parameters in Ledger

`savePreloadedData()` Internal method which saves a file with an estimation of time required time to scrape each filename

`loadPreloadedData()` Load a file with an estimation of time required time to scrape each filename

`getSizeToTimeScrape(sources, time2scrape = 60)` Estimates how much filenames could be scraped in a time frame, considering data retrieved with loadPreloadedData

`resetStatesMetrics()` Reset metrics of application of different status values

`countStatusUse(status.field,status)` Add an use to the metrics of status.field and status parameters

`getFilenamesStatusMode(mode,sources = sort(unique(self$df$source)),max.quant = 0,order.by.vertices.` Get a list of the filenames in the ledger with a defined mode (status agrupation)

`getFilenamesStatus(status,sources = sort(unique(self$df$source)),max.quant = 0,order.by.vertices.fa` Get a list of the filenames in the ledger with specified status

`selectDataEnv` *selectDataEnv*

Description

Asks the user where to set the system variable .data.env

Usage

```
selectDataEnv(env=NA)
```

Arguments

`env`	The environment to run on, can be PACKAGE, HOME or NULL. If null, it asks the user for a an Environment.

Value

.data.env

`setDataDirEnvironment` *setDataDirEnvironment*

Description

Sets the path of Rpolyhedra data dir.

Usage

```
setDataDirEnvironment(env = "PACKAGE")
```

Arguments

`env`	The type of environment to work with. Values are "PACKAGE" or "HOME" and it defaults to package

Details

This function is used to set the data directories either to the package or the user home directory.

Value

the curruent .data.env

setPackageEnvir *setPackageEnvir*

Description

Set a variable from package environment

Usage

```
setPackageEnvir(variable.name, value)
```

Arguments

variable.name	name of variable to be set
value	variable value

setUserEnvir *setUserEnvir*

Description

Sets variable in the writable environment for the package

Usage

```
setUserEnvir(variable.name, value)
```

Arguments

variable.name	name of variable to be set
value	variable value

`switchToFullDatabase` *switchToFullDatabase()*

Description

Prompts user for changing database to fulldb in user filespace

Usage

```
switchToFullDatabase(env=NA)
```

Arguments

`env`	The environment to run on, can be PACKAGE, HOME or NA. If NA, it asks the user for a an Environment.

Value

.data.env

`updatePolyhedraDatabase` *updatePolyhedraDatabase*

Description

Function for initializing database

Usage

```
updatePolyhedraDatabase()
```

Value

polyhedra db object

Index