Package 'spectrino'

July 27, 2015

Type Package

Title Spectra Visualization, Organizer and Data Preparation

Version 1.6.0

Date 2015-07-27

Author Teodor Krastev <spectrino@sicyon.com>

Maintainer Teodor Krastev <spectrino@sicyon.com>

Description Spectra visualization, organizer and data preparation from within R or stand-alone. Binary (application) part is installed separately by running spnInstallApp() immediately after installing the package.

License GPL (>= 2)

OS_type windows

Depends R (>= 3.0.0)

Imports utils, httpuv, jsonlite

LazyLoad yes

LazyData yes

URL http://www.spectrino.com

BuildKeepEmpty yes

NeedsCompilation no

Repository CRAN

Date/Publication 2015-07-27 23:02:14

R topics documented:

`spectrino-package` *Spectra visualization, organizer and data preparation*

Description

Spectra visualization, organizer and data preparation from within R or stand-alone

Details

Package:	spectrino
Type:	Package
Version:	1.6.0
Date:	2015-03-15
License:	MIT license

Installation: This is a binary package, Spectrino application (Windows) is integral part of the package. If you install it as a binary package (spectrino***.zip) all the components will be installed. If you install it as a source package from CRAN or some CRAN mirror (spectrino***.gz) only the R part will be installed, you have to run spnInstallApp() in order to install the application part from http://spectrino.com website.

Running: After creating spectrino object in R and opening spectrino application by using spnNew function, you can manipulate, select/deselect, open/close/save - individual spectra/group of spec-

tra/tree of groups. The manipulation features are complete enough. After finishing with the package you should free the Spectrino object and depending on "inclApp" parameter Spectrino application will be closed or not as well.

Author(s)

Teodor Krastev <spectrino@sicyon.com>

Maintainer: Teodor Krastev <spectrino@sicyon.com>

References

Teodor Krastev, Journal of Statistical Software" (v18, 2007)

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# set third group to be active
spnActGrp(3)

# get the number of active group
i <- spnActGrp(0)

# Release of Spectrino object
spnFree()
```

`spnActGrp` *Get/Set active group*

Description

Get/Set active group to Grp. if Grp=0 only get active group; else set one

Usage

```
spnActGrp(Grp)
```

Arguments

`Grp`	- the name(character string) or the index(integer) of the group; if 0 - just gets the active group number.

Value

`spnActGrp` returns the number of the active group.

Author(s)

Teodor Krastev

See Also

spnSetSpcChecked

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# set third group to be active
spnActGrp(3)

# get the number of active group
i <- spnActGrp(0)

# Release of Spectrino
spnFree(TRUE)
```

---

spnCheck *Check spectrino object existance*

Description

Check spectrino object existance. If the object is not there, you probably missed to create it with spnNew(...) or for some reason it has been destroyed.

Usage

```
spnCheck(inclApp = FALSE)
```

Arguments

inclApp	- logical (default is FALSE) option to include (or not) the connection to the application in the verification

Value

spnCheck returns logical for spectrino object/app existance.

Author(s)

Teodor Krastev

Examples

```
# Initialization of Spectrino
spnNew()

spnCheck(TRUE)

# Release of Spectrino
spnFree(TRUE)
```

---

spnDelGrp *Delete a group(s)*

Description

Delete Grp group. If Grp="*" then delete all of the groups from the spec-tree.

Usage

```
spnDelGrp(Grp)
```

Arguments

Grp	- the name(character string) or the index(integer) of the group; 0 - active group; "*" - all groups.

Value

spnDelGrp returns the number of groups after the deleting. (spnGetGrpCount)

Author(s)

Teodor Krastev

See Also

spnDelSpc

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# delete third group from the list
spnDelGrp(3)

# empty the whole list of groups
```

```
spnDelGrp("*")

# Release of Spectrino
spnFree(TRUE)
```

---

spnDelSpc *Delete a spec from a group*

Description

Delete Spc spectrum from Grp group. If Spc="*" then delete all of the spectra in that group them.

Usage

```
spnDelSpc(Grp,Spc)
```

Arguments

Grp	- the name(character string) or the index(integer) of the group; 0 - active group.
Spc	- the name(character string) or the index(integer) of spec; 0 - selected spec; "*" - all specs.

Value

spnDelSpc The function returns number of the spectra in that group after the deleting `spnGetSpcCount(false,Grp)`

Author(s)

Teodor Krastev

See Also

spnDelGrp

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# delete third spec from the first group
# i1 - the number of specs after deleting
i1 <- spnDelSpc(1,3)

# delete all the specs from the active group;
spnDelSpc(0,"*")

# Release of Spectrino
spnFree(TRUE)
```

spnFree *Release of Spectrino*

Description

Release R-object and closes application of Spectrino (optionally) after you have finished working with it. That is the proper way to close Spectrino, closing only the application will leave R-object of Spectrino.

Usage

```
spnFree(inclApp)
```

Arguments

inclApp	- logical (default is FALSE) option to include (or not) the release of tspectrino application as well

Value

spnFree - Returns nothing.

Author(s)

Teodor Krastev

Examples

```
# Initialization of Spectrino
spnNew()

# Release of Spectrino
spnFree(TRUE)
```

spnGetGrp *Get a group data*

Description

Get spectra from one spec-group (matrix). All the spectra in a group are assumed to have common X set of values, so if there is loaded spectrum in different X values, the spectrum is recalculated to fit that reference set.

Usage

```
spnGetGrp(OnlyChecked,Grp)
```

Arguments

OnlyChecked	- logical; if true gets only the checked specs.
Grp	- the name(character string) or the index(integer) of the group; 0 - active group.

Value

spnGetGrp returns a preprocessed group data in matrix. Spectra are always in rows (one spectrum is one row). The variables are columns, one variable (e.g. mass) is one column.

Author(s)

Teodor Krastev

See Also

spnGetSpc , spnGetTree , spnGetRefer

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# all the checked specs from the first group
m1 <- spnGetGrp(TRUE,1)

# all the specs from "Test2" group
m2 <- spnGetGrp(FALSE,"Test2")

# Release of Spectrino
spnFree(TRUE)
```

---

spnGetGrpCount *Number of groups loaded*

Description

Counts the number of spec-groups loaded in the current spec-tree.

Usage

```
spnGetGrpCount()
```

Value

spnGetGrpCount returns number of spec-groups loaded.

Author(s)

Teodor Krastev

See Also

spnGetSpcCount

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

n <- spnGetGrpCount()

# Release of Spectrino
spnFree(TRUE)
```

---

spnGetGrpName *Get the group name by index*

Description

Get the group name with an index GrpIdx

Usage

```
spnGetGrpName(GrpIdx)
```

Arguments

GrpIdx	- the index(integer) of the spec-group. Use GrpIdx=0 for active group. If GrpIdx="*" gets back a comma-separated list of all groups names.

Value

spnGetSpcCount returns name(character string) of spec-group with GrpIdx index.

Author(s)

Teodor Krastev

See Also

spnGetSpcName

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# the name of second spec-group
s1 <- spnGetGrpName(2)

# the name of the active group
s2 <- spnGetGrpName(0)

# Release of Spectrino
spnFree(TRUE)
```

---

spnGetRefer *Get common X values in a vector*

Description

Get reference X set of values (vector). All the spectra in a group list are assumed to have common X set of values, so if there is loaded spectrum in different X values, the spectrum is recalculated to fit the set given by the options: Boundaries from Low to High by 1. (in future versions the step could vary)

Usage

```
spnGetRefer()
```

Value

spnGetRefer returns the reference X set of values (vector)

Author(s)

Teodor Krastev

See Also

spnGetSpc ,spnGetGrp , spnGetTree

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")
```

```
# the reference X values
v1 <- spnGetRefer()

# Release of Spectrino
spnFree(TRUE)
```

---

spnGetSpc *Get the group name by index*

Description

Get one spectrum (vector) - only the Y-values of raw (unprocessed) data. All the spectra in a group are assumed to have common X set of values, so if there is loaded spectrum in different X values, the spectrum is recalculated to fit that reference set. If Spc is * the command is equivalent to getGetGrp(False,Grp) and gives back preporocessed data.

Usage

```
spnGetSpc(Grp,Spc)
```

Arguments

Grp	- the name(character string) or the index(integer) of the spec-group; 0 - active group.
Spc	- the name(character string) or the index(integer) of spec; 0 - selected spec; "*" - all specs

Value

spnGetSpc returns one spectrum (vector) - only the Y-values of raw (unprocessed) data.

Author(s)

Teodor Krastev

See Also

spnGetGrp , spnGetTree , spnGetRefer

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# if "Test2" is the second group, and "test23" - the third spec in it
```

```
v1 <- spnGetSpc(2,3)
# is equivalent to
v1 <- spnGetSpc("Test2","test23")

spnGetSpc(2,"*")
# is equivalent to
spnGetGrp(FALSE,2)

# Release of Spectrino
spnFree(TRUE)
```

---

spnGetSpcChecked *Gets the vector of the state of spec checking boxes of a group*

Description

Gets the logical vector of the state of checking boxes of Grp group.

Usage

```
spnGetSpcChecked(Grp)
```

Arguments

Grp	- the name(character string) or the index(integer) of the group; 0 - active group.

Value

spnGetSpcChecked returns the logical vector of checked spec state.

Author(s)

Teodor Krastev

See Also

spnSetSpcChecked

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# the logical vector of checked spec of the first group
bv1 <- spnGetSpcChecked(1)

# Release of Spectrino
spnFree(TRUE)
```

spnGetSpcCount *Number of spectra in Grp spec-group*

Description

Counts the number of specs in Grp group.

Usage

```
spnGetSpcCount(OnlyChecked,Grp)
```

Arguments

OnlyChecked	- logical; if true gets only the checked specs
Grp	- the name(character string) or the index(integer) of the spec-group; 0 - active group.

Value

spnGetSpcCount returns number of specs in Grp group.

Author(s)

Teodor Krastev

See Also

spnGetGrpCount

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# the number of the checked specs in second group
i1 <- spnGetSpcCount(TRUE,2)

# the number of specs in the active group
i2 <- spnGetSpcCount(FALSE,0)

# the number of specs in "Test3" group
i3 <- spnGetSpcCount(FALSE,"Test3")

# Release of Spectrino
spnFree(TRUE)
```

spnGetSpcName *Get the group name by index*

Description

Gets the spec name with an index SpcIdx from Grp group

Usage

```
spnGetSpcName(Grp,SpcIdx)
```

Arguments

Grp	- the name(character string) or the index(integer) of the spec-group; 0 - active group.
SpcIdx	- the index(integer) of the spec; 0 - selected spec. If SpcIdx="*" gets back a comma-separated list of all specs names in the group.

Value

spnGetSpcCount returns the name(character string) of spec with SpcIdx index from Grp group.

Author(s)

Teodor Krastev

See Also

spnGetGrpName

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# the name of second spec from the first group
s1 <- spnGetSpcName(2,1)

# the names-list of the active group
s2 <- spnGetSpcName(0,"*")

# the name of the third spec from "Test2" group
s3 <- spnGetSpcName("Test2",3)

# Release of Spectrino
spnFree(TRUE)
```

spnGetTree *Gets specs from all the groups*

Description

Get spectra from all the groups. Only the checked ones or all of them. Variables are by columns; measurements are by rows. The reason is "prcomp" principal component analysis accepts that order of data. Excluding the spectra from a group called "unknowns". That protected name is supposed to be for testing purposes only, so the data from that group are not included in all-data-get command.

Usage

```
spnGetTree(OnlyChecked)
```

Arguments

OnlyChecked	- logical; if true gets only the checked specs

Value

spnGetTree returns the matrix of specs from all the groups.

Author(s)

Teodor Krastev

See Also

spnGetSpc , spnGetGrp, spnGetRefer

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# all the specs from all the groups (excluding "unknowns" group, if any)
m1 <- spnGetTree(FALSE)

# Release of Spectrino
spnFree(TRUE)
```

spnInstallApp *Install Spectrino application from local zip file or spectrino website*

Description

Install spectrino application, which is Windows application required by the spectrino package to be functional. If zipFile argument is supplied spnInstallApp will install it from there, if left empty (default) it will download it from spectrino website and install it. This function is supposed to be used only once. The function spnNew() checks of spectrino.exe presence and if absent it will offer you to run spnInstallApp for you. If you run spnInstallApp with spectrino application already installed, the function will offer you to overwrite it.

Usage

```
spnInstallApp(zipFile="")
```

Arguments

zipFile	- character string to local zip file contains exec directory of spectrino application.

Value

spnInstallApp returns TRUE if succesful and FALSE otherwise.

Author(s)

Teodor Krastev

See Also

spnNew , spnFree

Examples

```
# Download and install the last version of Spectrino application
spnInstallApp(NULL)

# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# Release of Spectrino
spnFree(TRUE)
```

spnIsError *Check for spectrino error in a function result*

Description

Check if the result of spectrino function is error. It's a good practice to check the result from any spectrino function (except for spnNew, spnFree and spnCheck).

Usage

```
spnIsError(rslt)
```

Arguments

rslt	check if rslt is character and if it is, if it is an error message (first char is E

Value

spnIsError returns logical for spectrino error.

Author(s)

Teodor Krastev

Examples

```
# some data
mx <- c(10.4, 5.6, 3.1, 6.4, 21.7)

# is there an error
spnIsError(mx)

er <- "Error: test mistake"

# is there an error
spnIsError(er)
```

spnNew *Initialization of Spectrino*

Description

Check if R-object of Spectrino exists, and if not, creates/initializes Spectrino object/application. The command recommendable, but optional - it will be called, when any command is executed, if the R-object of Spectrino does not exists.

Usage

```
spnNew(TimeOut = 2, Host = "127.0.0.1", Port = 9876)
```

Arguments

TimeOut	time to wait [s] for spectrino application to reply, default is 2. In case of time-out error you may try to increase it
Host	Host IP address for the websocket server, the default is 127.0.0.1 which is local-host
Port	Port of web-socket communication, default is 9876

Value

spnNew - Gets back TRUE if the Spectrino object exists or has been created; otherwise - FALSE.

Author(s)

Teodor Krastev

Examples

```
# Initialization of Spectrino
spnNew()

# Release of Spectrino
spnFree(TRUE)
```

spnOpenGrp *Opens/Creates a spec-group*

Description

Open/Create a spec-group. All the spec files from one spec-group must be in the direcory of the group file. The names of spec-groups or specs are the filenames of the respected ones without path and extension (no space or special characters allowed). To avoid misinterpretation the names of group or spec cannot be numbers (even the operation system let you do it).

Usage

```
spnOpenGrp(GFilename,NewGrp)
```

Arguments

GFilename	- character string; If the path is missing, the path of current spec-tree is assumed. The extension (*.SGR) is not a must. All the filenames or directories must be with forward slashes e.g. D:/Prime/Data/Test.sgr
NewGrp	- logical. If NewGrp is true then GFilename shouldn't exist to be created. If NewGrp is false then GFilename must exist to be opened.

Value

spnOpenGrp returns the number of groups after the adding, which is the index of added group. spnGetGrpCount

Author(s)

Teodor Krastev

See Also

spnOpenSpc , spnOpenTree , spnSaveGrp

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# empty the whole list of groups
spnDelGrp("*")

# creates a new group;
# the directory must exists and the spec files must be in it
```

```
# i1 <- spnOpenGrp("D:/Prime/Data/Oils.sgr", TRUE)

# opens "Test2.sgr" file from current spec-tree directory
i2 <- spnOpenGrp("Test2", FALSE)

# Release of Spectrino
spnFree(TRUE)
```

---

spnOpenSpc *Open spec in spec-group*

Description

Open spec in Grp spec-group. All the spec files from one spec-group must be in the direcory of the group file, so avoid using a path for SFilename. The names of groups or specs are the filenames of the respected ones without path and extension (no space or special characters allowed). To avoid misinterpretation the names of group or spec cannot be numbers (even the operation system let you do it).

Usage

```
spnOpenSpc(Grp,SFilename)
```

Arguments

Grp	- the name(character string) or the index(integer) of the group; 0 - active group.
SFilename	- character string. The path is ignored and the path of current spec-group is assumed, so it would be better if you don't use path

Value

spnOpenSpc returns returns the number of specs after the adding, which is the index of added spec.

Author(s)

Teodor Krastev

See Also

spnOpenGrp , spnOpenTree

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# delete all the specs from second group;
spnDelSpc(2,"*")

# that will open existing spec Test23.txt from the directory of "Test2"
i1 <- spnOpenSpc("Test2","test23.txt")

# Release of Spectrino
spnFree(TRUE)
```

`spnOpenTree` *Open a spec-tree*

Description

Open a spec-tree from TFilename. The spec-tree file (.str) always is saved with the preprocessing options. The function returns To create new spec-tree you have to empty current one by `spnDelGrp("*")` and save the empty one under new name.

Usage

```
spnOpenTree(TFilename,InclOpt)
```

Arguments

`TFilename`	- character string. The extension (*.STR) is not a must. All the filenames or directories must be with forward slashes e.g. D:/Spectrino/Data/Test.str. "<test>" string generates test example
`InclOpt`	- integer (0,1,2); InclOpt rules where the preprocessing options will be taken from. If InclOpt = 0 then factory setting (no preprocessing) is used; 1 is for last used one; if 2 - the options are taken from TFilename

Value

`spnOpenTree` returns the number of groups in the new spec-tree. `spnGetGrpCount`

Author(s)

Teodor Krastev

See Also

`spnOpenSpc` , `spnOpenGrp` , `spnSaveTree`

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# Release of Spectrino
spnFree(TRUE)
```

---

spnSaveGrp *Save aspec- group*

Description

Save Grp group as GFilename file. The most common use is with GFilename="", to save the group under its proper name.

Usage

```
spnSaveGrp(Grp,GFilename)
```

Arguments

Grp	- the name(character string) or the index(integer) of the group; 0 - active group; "*" - all groups. If Grp="*" then all groups are saved under their proper names (in that case GFilename is ignored, but some string be present) and nothing gets back to R.
GFilename	- character string. The path of GFilename of is ignored, because any group file must be in the same directory as the specs in it. If GFilename is empty (the most common use), then Spectrino uses the proper name of the group.

Value

spnSaveGrp returns the full name of saved spec-group, except for Grp="*".

Author(s)

Teodor Krastev

See Also

spnOpenGrp , spnSaveTree

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# save second group under its name;
s1 <- spnSaveGrp(2,"")

# rename "Test2" group to "gassew" and then save it;
s2 <- spnSaveGrp("Test2","gassew")

# save all the groups under their proper names;
spnSaveGrp("*","")

# Release of Spectrino
spnFree(TRUE)
```

---

spnSaveTree *Save current spec-tree with the preprocessing options.*

Description

Save the current spec-tree along with the preprocessing options.

Usage

```
spnSaveTree(TFilename)
```

Arguments

TFilename	- character string. If TFilename is empty then Spectrino uses the proper name of the spec-tree.

Value

spnSaveTree returns the full name of saved spec-tree

Author(s)

Teodor Krastev

See Also

spnOpenTree , spnSaveGrp

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# save the spec-tree under its name
spnSaveTree("")

# rename  the spec-tree and save it
spnSaveTree("savenow")

# Release of Spectrino
spnFree(TRUE)
```

---

spnSetPPOpt *Set pre-processing options of Spectrino*

Description

Set Spectrino pre-processing options as string of semicolumn delimited list of following options (see example) Baseline=<integer> BaselineOn=0/1 MassBins=0/1 Normalize=0/1 MeanExtract=0/1 BaseGrp=<GroupName> LowLimit=<integer> HighLimit=<integer> Precision=<integer 1..10>

Usage

```
spnSetPPOpt(OptionList)
```

Arguments

OptionList	- string; comma separated list of options as they are in Preprocess section of a spec-tree.

Value

spnSetPPOpt Gets back the full options list.

Author(s)

Teodor Krastev

Examples

```
# Initialization of Spectrino
spnNew()

# Set Spectrino pre-processing options
spnSetPPOpt("Normalize=0;MeanExtract=0")
```

```
# Release of Spectrino
spnFree(TRUE)
```

---

spnSetSpcChecked *Set checked state of spec(s)*

Description

Set Spc spectrum of Grp spec-group checkbox to checked/unchecked state. If Spc="*" then all of specs in Grp group are set. If Grp="*" then all of spectra in all groups are set to Checked (in that case Spc is ignored).

Usage

```
spnSetSpcChecked(Grp,Spc,Checked)
```

Arguments

Grp	- the name(character string) or the index(integer) of the group; 0 - active group; "*" - all groups
Spc	- the name(character string) or the index(integer) of spec; 0 - selected spec; "*" - all specs.
Checked	- logical, the state which will be set

Value

spnSetSpcChecked none

Author(s)

Teodor Krastev

See Also

spnGetSpcChecked, spnActGrp

Examples

```
# Initialization of Spectrino
spnNew()

# generate test set
spnOpenTree("<test>")

# check just one
spnSetSpcChecked(2,3,TRUE)
```

```
# all the spec from second group to OFF
spnSetSpcChecked(2,"*",FALSE)

# all the specs in all groups to ON
spnSetSpcChecked("*","*",TRUE)

# Release of Spectrino
spnFree(TRUE)
```

---

spnSetVis *Set Spectrino to be visible or hidden*

Description

Set Spectrino to be visible or hidden, if Visible = TRUE shows Spectrino else hides Spectrino. Gets back the current visibility. If you want to use Spectrino only as a data-tank, that is the way to hide it.

Usage

```
spnSetVis(Visible)
```

Arguments

Visible	- logical; set a visibility state

Value

spnSetVis Gets back the current visibility.

Author(s)

Teodor Krastev

Examples

```
# Initialization of Spectrino
spnNew()

# shows Spectrino
b1 <- spnSetVis(TRUE)

# hides Spectrino
b1 <- spnSetVis(FALSE)

# Release of Spectrino
spnFree(TRUE)
```

spnValidation *Validation of Spectrino*

Description

Spectrino validation - not conclusive, tests only the most common functions and modes.

Usage

```
spnValidation()
```

Value

spnValidation - If it gets back "Validation confirmed" you have very good chances that Spectrino might work, otherwise you will have the error with a number (see the code).

Author(s)

Teodor Krastev

Examples

```
# Initialization of Spectrino
spnNew()

spnValidation()

# Release of Spectrino
spnFree(TRUE)
```

Index