Package 'spectrino'

May 9, 2026

Type Package

Title Spectra Viewer, Organizer, Data Preparation and Property Blocks

Version 2.0.0

Date 2019-05-26

Author Teodor Krastev <spectrino@sicyon.com>

Maintainer Teodor Krastev <spectrino@sicyon.com>

Description Spectra viewer, organizer, data preparation and property blocks from within R or stand-alone. Binary (application) part is installed separately using spnInstallApp() from spectrino package.

License GPL (>= 2)

OS_type windows

Depends R (>= 3.0.0)

Imports httpuv, jsonlite, utils

LazyLoad yes

LazyData yes

URL http://www.spectrino.com

BuildKeepEmpty yes

NeedsCompilation no

Repository CRAN

Date/Publication 2019-06-03 23:50:07 UTC

Contents

`spectrino-package` *Spectra viewer, organizer, data preparation and property blocks*

Description

Spectra viewer, organizer, data preparation and property blocks from within R or stand-alone. Binary (Spectrino application) part is installed separately.

Details

Package:	spectrino
Type:	Package
Version:	2.0.0
Date:	2019-04-15
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 spectra/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.

General features:

Spec part: for the spec part of Spectrino common point are: - all indexes of items (spec-tree, spec-group and specs) are 1-based and 0 is resertved for the active one. For the active spec-tree is the top one, the active group is the selected one and the active - common notation for mask is "*" or "<ALL>" which mask all item in the designated list - when you refer to spec-tree, spec-group or spec, you can do that by number or by name. That's why it is not a good idea to name any item with a number (even you operation system would allow it).

Block part: - a core of block is list of properties, you can read/write the values of these properties from your code or from the user interface. - the four types of property are: boolean, integer, double and string. You create or modify the types and some other staff from configration dialog. - there are three other parts of a block: chart, log and source. Chart will follow the evolution of any numerical property in a graph. Log will do the same and more in text format. Source will keep multi-line pieces (up to 3) of text you can read from R as text or execute if it is an excutable R code.

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()
```

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

# Release of Spectrino object (optional)
#spnFree()
```

---

spnActGrp *Get/Set active group in the top tab*

Description

Get/Set active group to Grp in the top tab. if Grp=0 only get active group index; 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 index.

Value

spnActGrp returns the index (1-based index) of the active group.

Author(s)

Teodor Krastev

See Also

spnSetSpcChecked

Examples

```
# Initialization of Spectrino
spnNew()

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

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

spnActTree *Get/Set active tree (the top tab)*

Description

Get/Set active tree to Tree by name or index. if Tree=0 only get active tree index; else set one

Usage

```
spnActTree(Tree)
```

Arguments

Tree	- the name(character string) or the index(integer) of the tree; if 0 - just gets the active tree index.

Value

spnActTree returns the index (1-based index) of the active tree.

Author(s)

Teodor Krastev

See Also

spnActGrp

Examples

```
# Initialization of Spectrino
spnNew()

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

# set third group to be active
spnActTree(1)

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

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

spnAddTree *Open a spec-tree in a new tab*

Description

Open a spec-tree in a new tab from TFilename. If the active tab/spec-tree is empty (<no-name> title and no groups opened) that tab will be used instead of creating a new one. The spec-tree file (.str) always is saved with the preprocessing options. If the file does not exist the function will create an empty one, so you can add groups and specs and save it with spnSaveTree("")

Usage

```
spnAddTree(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 (simulation)
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

spnAddTree returns the number of tabs/spec-trees. spnGetGrpCount

Author(s)

Teodor Krastev

See Also

spnOpenTree , spnOpenSpc , spnOpenGrp , spnSaveTree

Examples

```
# Initialization of Spectrino
spnNew()

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

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

spnChartBlock *Initialization of block's chart*

Description

Set the properties to be charted with each new iteration. These properies could be set from r with this command or from Spectrino app block interface.

Usage

```
spnChartBlock(Block, listOfProps)
```

Arguments

Block	The name of the block whose chart is set
listOfProps	Vector with the property names to be charted

Value

spnChartBlock - Gets back TRUE if opration is successful or FALSE if any of properties does mach block's properies or the chart is disabled (not enabled).

Author(s)

Teodor Krastev

Examples

```
# Initialization of Spectrino
spnNew()

# generate test group of blocks
spnOpenGroupOfBlocks("<test>")

spnChartBlock("test1",c("objective.0","sug.epsilon"))

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

spnCheck *Check spectrino object existance*

Description

Check for 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. If you are checking for the application too, the function will check only for existing connection to the application.

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)
```

spnDelBlock *Delete a block from the current group of blocks*

Description

Delete a block from the current group of blocks. You can delete all blocks <ALL> or the entire group <GROUP>. If you want to open a block, a group of blocks (even empty) must be present.

Usage

```
spnDelBlock(Block)
```

Arguments

Block	- character string of the name of the block to be deleted. – <ALL> will delete all the block leaving empty group of blocks. – <GROUP> will close current group of blocks leaving only the console behind.

Value

spnDelBlock returns TRUE if successful, FALSE - otherwise.

Author(s)

Teodor Krastev

See Also

spnOpenBlock , spnOpenGroupOfBlocks , spnSaveBlock

Examples

```
# Initialization of Spectrino
spnNew()

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

# delete a block
spnDelBlock("test2")

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

---

spnDelGrp *Delete a group(s) in the top tab/spec-tree*

Description

Delete Grp group in the top tab/spec-tree. If Grp="*" or "<ALL>" 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; "*" or "<ALL>" - 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 in the top tab/spec-tree*

Description

Delete Spc spectrum from Grp group in the top tab/spec-tree. If Spc="*" or "<ALL>" 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; "*" or "<ALL>" - 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)
```

---

spnDelTree *Delete a tree / tab*

Description

Delete a tree(s) from spec side. If Tree="<ALL>" then delete all of the trees.

Usage

```
spnDelTree(Tree)
```

Arguments

Tree	- the name(character string) or the index(integer) of the tree; 0 - active tree; "<ALL>" - all trees.

Value

spnDelTree returns the number of trees after the deleting.

Author(s)

Teodor Krastev

See Also

spnDelGrp

Examples

```
# Initialization of Spectrino
spnNew()

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

# delete active tree / tab
spnDelTree(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 Spectrino application as well

Value

spnFree - Returns nothing.

Author(s)

Teodor Krastev

Examples

```
# Initialization of Spectrino
spnNew()

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

spnGetBlockNames *Names of all properies of a block or all the blocks names*

Description

Get the names of all properies of a block or all the blocks names in the current group of blocks

Usage

```
spnGetBlockNames(Block = "")
```

Arguments

Block	- the name(character string) of a block; if empty - gets back the vector of block names.

Value

spnGetBlockNames returns a vector with the names of all properies of a block or all the blocks names.

Author(s)

Teodor Krastev

See Also

spnGetGrpName

Examples

```
# Initialization of Spectrino
spnNew()

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

# all the properties of block test1
spnGetBlockNames("test1")

# all the blocks if current group of blocks
spnGetBlockNames()

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

spnGetGrp *Get a group data in the top tab/spec-tree*

Description

Get spectra from one spec-group (matrix) in the top tab/spec-tree. 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 a group in the top tab/spec-tree; 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 in the top tab/spec-tree*

Description

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

Usage

```
spnGetGrpCount()
```

Value

`spnGetGrpCount` returns number of spec-groups loaded in the top tab/spec-tree.

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 in the top tab/spec-tree*

Description

Get the group name with an index GrpIdx in the top tab/spec-tree

Usage

```
spnGetGrpName(GrpIdx)
```

Arguments

GrpIdx	- the index(integer) of the spec-group in the top tab/spec-tree. Use GrpIdx=0 for active group. If GrpIdx="*" or "<ALL>" 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)
```

---

spnGetProperty *Get a property value from a block*

Description

Get a property value from a block. The properties of a block can be configured from the block menu (configure).

Usage

```
spnGetProperty(Block, Prop)
```

Arguments

Block	- the name(character string) of a block.
Prop	- the name(character string) of a property. A special integrated property <ITERS> will give the current number of the iteration counter

Value

spnGetProperty returns the value of the property, the type depends of the type of propery in the block.

Author(s)

Teodor Krastev

See Also

spnSetProperty

Examples

```
# Initialization of Spectrino
spnNew()

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

# set a property
spnSetProperty("test1","objective.0",3.58)

# get a property
spnGetProperty("test1","objective.0")

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

---

spnGetRefer — *Get common X values in a vector of the active group in the top tab/spec-tree*

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.

Usage

```
spnGetRefer()
```

Value

spnGetRefer returns the reference X set of values (vector) of the active group in the top tab/spec-tree

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 vector of specific spec in the top tab/spec-tree*

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; "*" or "<ALL>" - 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")

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

---

spnGetSpcChecked *Gets the vector of the state of spec checking boxes of Grp group in the top tab/spec-tree*

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 in the top tab/spec-tree*

Description

Counts the number of specs in Grp group in the top tab/spec-tree.

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 in the top tab/spec-tree.

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 *Gets the spec name with an index SpcIdx from Grp group*

Description

Gets the spec name with an index SpcIdx from Grp group in the top tab/spec-tree

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 of the top tab/spec-tree*

Description

Get spectra from all the groups of the top tab/spec-tree. 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)
```

---

spnGetTreeNames *Get a vector of tree names or filenames*

Description

Get a vector of tree names or filenames depending on Filenames FALSE/TRUE respectively

Usage

```
spnGetTreeNames(Filenames = FALSE)
```

Arguments

Filenames	- the boolean indicates if all Filenames will be returned or just the names.

Value

spnGetTreeNames a vector of tree names or filenames.

Author(s)

Teodor Krastev

See Also

spnGetGrpName

Examples

```
# Initialization of Spectrino
spnNew()

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

# the filesname of all trees
```

```
s1 <- spnGetTreeNames(TRUE)

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

---

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

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. In case of update (when you run R-package version lower than your spectrino app version) that would be the easiest way to update your app. N.B. Version 2.0.* of spectrino application is fully backwards compatible to 1.5 and 1.6 versions of spectrino R-package.

Usage

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

Arguments

zipFile	- character string to local zip file contains exec directory of spectrino application. If NULL, the instalation is void. (omitted)

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()
```

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

---

spnIsError *Check for spectrino error in a function result*

Description

Checking if the result of spectrino function is error. In your code, it's a good practice to check the result from any spectrino function (except for spnNew, spnFree and spnCheck). From command line there is no sense to call this function because you would see the error message on the terminal.

Usage

```
spnIsError(rslt)
```

Arguments

rslt	check if rslt is character and if it is, if it is an error message (starts with Error:)

Value

spnIsError returns logical for spectrino error.

Author(s)

Teodor Krastev

Examples

```
spnNew()

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

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

# is there an error
spnIsError(m1)

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

`spnIteration` *Iteration control in blocks*

Description

Control iteration count: - if the expected number of iterations is unknown Initial should be set to -2 or -1 (without or with progres bar) - if the expected number of iterations is known Initial should be set to that number

after initializing the iteration counter with any Initial<>0, your code should call Iteration() on each iteration (usually at the end)

Apart of the progress bar, the block will react to an iteration depending of how have been set. E.g. Chart will draw next value(s) and/or log will write next status according to the log template.

Usage

```
spnIteration(Initial = 0)
```

Arguments

`Initial`	if Initial <> 0 the iteration counter is est 0 -2: only internal count (no progress bar) -1: normal count including progress bar, when the final count is unknown 0: one iteration (moves the iteration count 1 up) n: (n>0) the expected number of iterations, with progress bar

Value

`spnIteration` - Gets back the iteration counter value.

Author(s)

Teodor Krastev

Examples

```
# Initialization of Spectrino
spnNew()

# initializing with 5 iteration expected iteration count
spnIteration(5)

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

spnLogBlock *Log to a blocks log or in the console*

Description

Log some text and/or values in the blocks log or in the console. The user can send some text, or use some special syntax for listing one or all properties. Alernative to that command you can use a log template (set from block's menu) which contains text and $property.name$ the last one will be replaced by that property value. The template will ber used at each iteration.

Usage

```
spnLogBlock(Block, text)
```

Arguments

Block	- the name(character string) of the target block; if cannot be found (e.g. "") the main log (console) is used.
text	- message character string to be logged. Usually it's a some text message, but there are some special cases: – <property.name> wil log the name and the value of that property – <ALL> will log all the properties (name=value) of that block – <CLEAR> will clear the log

Value

spnLogBlock returns TRUE if successful, FALSE - otherwise.

Author(s)

Teodor Krastev

See Also

spnChartBlock , spnIteration

Examples

```
# Initialization of Spectrino
spnNew()

# write into the console
spnLogBlock("","some text")

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

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 load and reply, default is 2. In case of time-out error maybe your hardware is not that fast and you may need to increase the TimeOut.
Host	Host IP address for the websocket server, the default is 127.0.0.1 which is localhost. Do not change it unless you really know what you are doing.
Port	Port of web-socket communication, default is 9876. Do not change it unless you really know what you are doing.

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)
```

spnOpenBlock *Open a block of properties*

Description

Open a block of properties at a position. You can specify the whole file name including *.blk extention, but it's a bit pointless as all the blocks must be in the same directory as their group of block file.

Usage

```
spnOpenBlock(Block, atPos = -1)
```

Arguments

Block	- character string, name of the block to open. All the blocks must be in the same directory as their group of block file.
atPos	- integer (0..8) position of the block 1..8 are positions from top to bottom left to right (see Spectrino option for the map). 0 if you wish the block to be hidden. -1 is for automatic placement.

Value

spnOpenBlock returns TRUE if successful, FALSE - otherwise.

Author(s)

Teodor Krastev

See Also

spnOpenGroupOfBlocks , spnSaveBlock

Examples

```
# Initialization of Spectrino
spnNew()

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

# delete a block
spnDelBlock("test2")

# open a block
spnOpenBlock("test2")

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

`spnOpenGroupOfBlocks` *Open a group of blocks*

Description

Open a group of blocks from file "Filename" as list of names of blocks and then it opens all the blocks from the list. "<test>" as filename will generate test example group.

Usage

```
spnOpenGroupOfBlocks(Filename)
```

Arguments

`Filename`	- character string. The extension (*.GBK) is not a must. If the path is missing the default (blocks) folder is assumed. All the filenames or directories must be with forward slashes e.g. D:/Spectrino/Blocks/Test.gbk. "<test>" string generates test example group of blocks.

Value

`spnOpenGroupOfBlocks` returns TRUE if successful, FALSE - otherwise.

Author(s)

Teodor Krastev

See Also

`spnOpenBlock` , `spnSaveBlock` , `spnSaveGroupOfBlocks`

Examples

```
# Initialization of Spectrino
spnNew()

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

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

spnOpenGrp *Opens/Creates a spec-group in the top tab/spec-tree*

Description

Open/Create a spec-group in the top tab/spec-tree. 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 in the top tab/spec-tree*

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 in the top tab/spec-tree*

Description

Open a spec-tree in the active tab from TFilename. The spec-tree file (.str) always is saved with the preprocessing options. If there is no active tab opened Spectrino will create one for you (as in spnAddTree()). To create new spec-tree you have to either open spec-tree with non-existant filename or 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)
```

---

spnSaveBlock *Save a block of properties*

Description

Save a block of properties in the folder of the group of blocks it belongs to.

Usage

```
spnSaveBlock(Block)
```

Arguments

Block	- the name(character string) of block to be saved. The destination directory is that of the group of block the block belongs to. – <ALL> will save all the blocks from current group of blocks

Value

spnSaveBlock returns TRUE if successful, FALSE - otherwise.

Author(s)

Teodor Krastev

See Also

spnOpenBlock , spnSaveGroupOfBlocks

Examples

```
# Initialization of Spectrino
spnNew()

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

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

```
# save a block
spnSaveBlock("test3")

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

---

spnSaveGroupOfBlocks *Save the current Group of Blocks*

Description

Save the current group of blocks as a list of names. Depending of the option from option dialog the command will save (default) or not the block themselves. If you want to make sure that the blocks are saved regardless that option use spnSaveBlock("<ALL>") after with this command. Blocks always are saved in the same directory of their group of blocks file.

Usage

```
spnSaveGroupOfBlocks(Filename = "")
```

Arguments

Filename	- character string. The extension (*.GBK) is not a must. All the filenames or directories must be with forward slashes e.g. D:/Spectrino/Blocks/Test.GBK. If the path is missing the default block path is assumed. If the argument Filename is missing, the filename when it was opened is assumed.

Value

spnSaverGroupOfBlocks returns TRUE if successful, FALSE - otherwise.

Author(s)

Teodor Krastev

See Also

spnOpenBlock , spnOpenGroupOfBlocks , spnSaveBlock

Examples

```
# Initialization of Spectrino
spnNew()

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

# save test group
```

```
spnSaveGroupOfBlocks("")

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

---

spnSaveGrp *Save a spec-group in the top tab/spec-tree*

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 empty string has to be provided) 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 the top (active) 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 top spec-tree*

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; semicolumn separated list of options as they are in Preprocess section of the top 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)
```

spnSetProperty *Set a property value in a block*

Description

Set a property value in a block. The properties of a block can be configured from the block menu (configure).

Usage

```
spnSetProperty(Block, Prop, Value)
```

Arguments

Block	- the name(character string) of a block.
Prop	- the name(character string) of a property.
Value	- the value of the property, the type depends of the type of property in the block.

Value

spnSetProperty returns TRUE if successful, FALSE - otherwise.

Author(s)

Teodor Krastev

See Also

spnGetProperty

Examples

```
# Initialization of Spectrino
spnNew()

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

# set a property
spnSetProperty("test1","objective.0",3.58)

# get a property
spnGetProperty("test1","objective.0")

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

spnSetSpcChecked *Set checked state of spec(s) in the top tab/spec-tree*

Description

Set Spc spectrum of Grp spec-group checkbox(es) 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 application to be visible or hidden*

Description

Set Spectrino application to be visible or hidden, if Visible = TRUE shows Spectrino else hides Spectrino. Gets back the current visibility. If you want to use Spectrino application only as a data-storage, 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)
```

spnSourceBlock *Get the source from a block and optionally execute it*

Description

Get the srcIdx source from a block and optionally execute it namespace/enviroment you are calling the function from.

Usage

```
spnSourceBlock(Block, srcIdx, Eval = TRUE)
```

Arguments

Block	- character string, a name of a block.
srcIdx	- integer (1,2,3); Gets a source (code) from srcIdx tab of the respective block
Eval	- execute the code in namespace/enviroment you are calling the function from.

Value

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

Author(s)

Teodor Krastev

See Also

spnLogBlock , spnChartBlock

Examples

```
# Initialization of Spectrino
spnNew()

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

# gets source from tab 1 source of the block and execute it
spnSourceBlock("test3", 1, TRUE)

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

spnValidation *Validation of Spectrino*

Description

Spectrino validation - not conclusive, it tests only the most common functions and modes. If both Spec and Block are FALSE, it tests only the Spectrino object, connection and Spectrino app. presence.

Usage

```
spnValidation(Spec, Block)
```

Arguments

Spec	- logical (default is TRUE) option to include (or not) a test of most used spectral commands
Block	- logical (default is TRUE) option to include (or not) a test of most used block of prop. commands

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()

# test of the Spectrino object, connection and Spectrino app. presence
spnValidation(FALSE,FALSE)

# Release of Spectrino
spnFree(TRUE)
```

Index