Package ‘neuR’

January 13, 2016

Version 0.1

Date 2015-10-31

Title R package for neuroscience data processing and statistical
analysis of neuro-data

Author Livio Finos, Alessandro Rigon, Silvia Palmieri, Marco Zanetti, Santiago Liebrecht

Maintainer livio finos `<livio.finos@unipd.it>`

Description R package for neuroscience data processing and statistical analysis of neuro-data

License GPL (>= 2)

RoxygenNote 5.0.0

R topics documented:

`add.neuR.funct` *add funct to neuR-object*

Description

add funct to neuR-object

Usage

```
add.neuR.funct(obj, funct, out.name = funct, ...)
```

Arguments

`obj` neuR-object

`out.name` map by default. name of the function to be added in obj@function$out.name

`map` function to be returned

Value

a neuR-object

`add.neuR.map` *add volume to neuR-object*

Description

add volume to neuR-object

Usage

```
add.neuR.map(obj, map, out.name = NULL, ...)
```

Arguments

`obj` neuR-object

`map` name of the map (function) to be added. It can also be a list of maps (i.e. 3D arrays)

`out.name` equal to map (or names(map) if map is a list) by default. Name of the map to be added to the returned object.

Value

a neuR-object

compute.irc *Computes IRC (intra run correlation) of a neuR-object*

Description

Computes IRC (intra run correlation) of a neuR-object pcs array of slot data of a neuR-object (i.e. D@data$pcs)

Usage

```
## S3 method for class 'irc'
compute(D, pc.num = 1, drop.tcs = FALSE)
```

Arguments

D	a neuR-object
pc.num	1
drop.tcs	FALSE

Value

a 3D array

compute.irh *Computes IRH (intra run homogeneity) of a neuR-object*

Description

Computes IRH (intra run homogeneity) of a neuR-object pcs array of slot data of a neuR-object (i.e. D@data$pcs) indice di omogeneità dei k loadings. indice in scala 0-1. la norma essendo pari a 1, implica un momento secondo pari a 1/k. il momento secondo è decomponibile in varianza + media^2. l'indice è pari a media(loadings)^2*k. se 1 tutti i loading sono uguali tra loro (e pari a 1/sqrt(k)) se 0, i loadings hanno media 0 e la varianza vale 1/k

Usage

```
## S3 method for class 'irh'
compute(D, pc.num = 1, drop.tcs = FALSE)
```

Arguments

D	a neuR-object
pc.num	1
drop.tcs	FALSE

Value

a 3D array

`compute.pcs` *Computes principal components for each voxel/channel of a neuR-object*

Description

Computes principal components for each voxel/channel stored in tcs array of slot data of a neuR-object (i.e. D@data$tcs)

Usage

```
## S3 method for class 'pcs'
compute(D, center = TRUE, scale = FALSE, max.pc.num = 1,
  drop.tcs = TRUE, selected.volumes = NULL, ...)
```

Arguments

`D`	a neuR-object
`center`	logical, TRUE by default
`scale`	logical, FALSE by default
`max.pc.num`	1
`drop.tcs`	TRUE
`selected.volumes`	NULL or a logical vector of length n (number of volumes per block)
`...`	other

Value

a neuR-object

compute.test *Computes Tests in a neuR-object*

Description

Computes tests of a neuR-object pcs array of slot data of a neuR-object (i.e. D@data$pcs)

Usage

```
## S3 method for class 'test'
compute(D, left.array, right.formula = ~1,
  offset.values = NULL, tail = 1)
```

Arguments

`D` a neuR-object

`left.array` the name of the map in D@data to be used in the model or the array itself

`right.formula` ~1 by default. NOT IMPLEMENTED YET. The names refer to the columnames of D@info$design

`offset.values` (NULL by default, same effect as 0) A value to be subtracted to each value of the matrix (same dims of left.array) It may also be 'meanOverall' or 'meanBySubject' (i.e. average over the second dimension)

Value

a list of 3D arrays, usually a T and P one.

extractVxls *extractVxls*

Description

select voxels (or time courses of voxels) from a matrix of coordinates

Usage

```
extractVxls(xyz, D)
```

Arguments

`xyz` a nX3 matrix of n coordinates

`D` a neuR-object

`tcs` any proper array e.g. from D@data$tcs

`mask` usually D@mask

`get.neuR.map` *get functions to compute maps neuR-object*

Description

NOT IMPLEMENTED YET: out.funct.name=map the name of the function to be generated (eg different names depending on the parameters)

NOT IMPLEMENTED YET: out.funct.name=map the name of the function to be generated (eg different names depending on the parameters)

Usage

```
get.neuR.funct(funct, ...)

get.neuR.map(obj, map, recompute = FALSE, ...)
```

Arguments

`funct`	
`...`	are used to set the parameters of the map
`obj`	neuR-object
`map`	function name to be returned or the array itself
`recompute`	FALSE

Value

a function

a volume

`names.map` *dimnames of the maps*

Description

dimnames of the maps

Usage

```
## S3 method for class 'map'
names(object)
```

Arguments

`object`	a neuR-object

Value

a list of dimnames

`neuR` *neuR: R package for neuroscience data processing and statistical analysis of neuro-data*

Description

The neuR package provides functions to deal with fMRI, fNIRS, EEG data.

`neuR.object-class` *An S4 class to store fNIRS, fMRI, EEG data*

Description

An S4 class to store fNIRS, fMRI, EEG data

Slots

`data` list of 3D arrays

`mask`

`info`

`pixelize` *pixelize*

Description

reduce a neuR-object

Usage

```
pixelize(D, reduce.by = 2)
```

Arguments

`D`	a neuR-object
`reduce.by`	(integer) resizing factor. If it is a vector, the tree coordinates indicate the rescaling to bi applied to each dimension. If it is a scalar, the same rescaling is applied to the 3 dims.

`read.fMRI.data` *Reads .nii and .img/.hdr file data and save to neuR-object*

Description

Reads .nii and .img/.hdr file data and save to neuR-object

Usage

```
read.fMRI.data(path = ".", pattern = "s.*\\.img", files = NULL,
  mask = "constant", info = NULL, silent = FALSE, exclude.files = c(),
  header.file = NULL)
```

Arguments

`path`	"."
`pattern`	"s.*\.img"
`files`	NULL
`mask`	a file name, a array, equal to 'constant' (look for non constant voxels) or a scalar (values equal to this number are out of the brain)
`info`	NULL
`silent`	FALSE
`exclude.files`	vector of ids of files to exclude
`header.file`	name of the file from which the head should be read. if NULL (default) the head of the first file (not excluded by exclude.files) is used

Value

a neuR-object

`reshapeTcs2blocks` *Reshapes D@data$tcs in a 3D (time X voxel/channel X block)*

Description

Reshapes D@data$tcs in a 3D (time X voxel/channel X block)

Usage

```
reshapeTcs2blocks(D, blocks)
```

Arguments

`D`	a neuR-ogject
`blocks`	if scalar, it is the number of volumes in each blocks

Value

a neuR-object

`scale.neuR` *scale.neuR*

Description

same as scale but does not divide

Usage

```
## S3 method for class 'neuR'
scale(X, center = TRUE, scale = FALSE)
```

`summary.map` *summary of all maps*

Description

summary of all maps

Usage

```
summary.map(object, maps = NULL)
```

Arguments

`object`	a neuR-object
`maps`	NULL (default) or a vecotr of names of maps

Value

a table of summary stats for each volume

write.volumes *writes .nii and .img/.hdr file file from a neuR-object*

Description

writes .nii and .img/.hdr file file from a neuR-object. occhio agli header, per ora file.type pari al formato della immagine che ha generato l'header.

Usage

```
write.volumes(D, which.maps = NULL, file.root.name = "V",
  file.type = "nifti", into.path = ".")
```

Arguments

D	neuR-object
which.maps	by default: all maps in D
file.root.name	"V"
file.type	"nifti"
into.path	"."

Index