Package 'nat.templatebrains'

September 16, 2014

Type Package

Title NeuroAnatomy Toolbox (nat) Extension for Handling Template Brains

Version 0.4.1

Date 2014-09-13

Author James Manton and Greg Jefferis

Maintainer James Manton <ajd.manton@googlemail.com>

Description Extends package nat (NeuroAnatomy Toolbox) by providing objects and functions for handling template brains.

License GPL-3

Depends R (>= 2.10), rgl, nat (>= 1.5.9)

Suggests testthat

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-16 18:40:27

R topics documented:

nat.templatebrains-package

NeuroAnatomy Toolbox add-on package for handling template brains

Description

This package provides a class `templatebrain` that stores key information about referennce brains along with helpfer functions to simplify transformation of data between template brains (a.k.a bridging) and mirroring of data within a template brain. Presently there

Helper functions

Easy-to-use functions for transforming data from one template brain to another, displaying slices alongside 3D data, etc. are provided. See especially `xform_brain`, `mirror_brain` and `plot3d.templatebrain`.

Package options

- options('nat.templatebrains.regdirs') specifies a character vector of directories containing registrations.

Note that registration directories should be specified as full paths and will be searched in the order that they are listed. Registrations should therefore have globally unique names.

See Also

`nat`

Examples

```
## Not run:
options(nat.templatebrains.regdirs=union(
  getOption('nat.templatebrains.regdirs'), "/my/new/path"))

## End(Not run)
```

as.templatebrain

Use image file or other object to initialise template brain

Description

Use image file or other object to initialise template brain

Usage

```
as.templatebrain(x, ...)

## S3 method for class 'character'
as.templatebrain(x, ...)

## S3 method for class 'im3d'
as.templatebrain(x, name = NULL, regName = NULL, ...)
```

Arguments

x	object used to construct the templatebrain, either a character vector with the path to a file or an im3d object.
...	additional named arguments passed to methods and then on to templatebrain that will be added as fields to the templatebrain object.
name,regName	name and short name of the template brain. Will use the filename (minus final extension) by default for both fields.

Value

A list with class templatebrain.

See Also

im3d

Examples

```
# Make templatebrain object using image info from the template brain NRRD file
nhdr=system.file('images','FCWB.nhdr', package='nat.templatebrains')
as.templatebrain(nhdr, name = "FlyCircuit Whole Brain")
```

---

bridging_sequence *Find sequence of one or more bridging registrations*

Description

This function is primarily intended for developer use (it is used inside xform_brain) but may be useful for end users.

Usage

```
bridging_sequence(sample, reference, via = NULL, checkboth = FALSE,
  mustWork = FALSE)
```

Arguments

sample	source template brain (e.g. IS2) that data is currently in.
reference	target template brain (e.g. IS2) that data should be transformed into.
via	optional intermediate brain to use when there is no direct bridging registration.
checkboth	Whether to look for registrations in both directions. The default (checkboth=FALSE) will only return registrations in the forward direction (see details).
mustWork	Whether to error out if appropriate registrations are not found.

Details

When checkboth=FALSE, only registrations that can be directly used to map image data from sample to reference are returned. When working with 3D points, use checkboth=TRUE. Note that all possible directories will first be scanned for registrations in the preferred direction and then rescanned for the opposite direction if nothing is found.

registration direction

When mapping points from JFRC2 -> IS2 -> FCWB (i.e. sample=JFRC2, via=IS2, ref=FCWB) the command line passed to CMTK's streamxform should look like: streamxform -- JFRC2_IS2.list --inverse FCWB_IS2.list However when mapping image data the command line for CMTK's reformatx should look like: reformatx -- JFRC2_IS2.list --inverse FCWB_IS2.list and the corresponding output might look like list(JFRC2 = structure("/GD/dev/R/nat.flybrains/inst/extdata/bridgingregistrations/JFRC2_IS2.list

Examples

```
## Not run:
bridging_sequence(sample=JFRC2, ref=FCWB, checkboth = T)
bridging_sequence(sample=JFRC2, via=IS2, ref=FCWB, checkboth = T)

## End(Not run)
```

---

display_slice *Display an image slice in 3D*

Description

Display an image slice in 3D

Usage

```
display_slice(brain, slice, ...)
```

Arguments

brain	template brain (e.g. IS2) of the slice.
slice	path to the slice PNG image to display.
...	extra arguments to pass to persp3d.

FCWB.demo *Sample template brain: FlyCircuit Whole Brain*

Description

This is a sample template brain for testing purposes which is equivalent to the FCWB template brain defined by the `nat.flybrains`, which should be considered the canonical version.

mirror_brain *Mirror 3D object around a given axis, optionally using a warping registration*

Description

Mirror 3D object around a given axis, optionally using a warping registration

Usage

```
mirror_brain(x, brain, mirrorAxis = c("X", "Y", "Z"), transform = c("warp",
  "affine", "flip"), ...)
```

Arguments

`x`	the 3D object to be mirrored.
`brain`	source template brain (e.g. IS2) that data is in.
`mirrorAxis`	the axis to mirror (default `"X"`).
`transform`	whether to use warp (default) or affine component of registration, or simply flip about midplane of axis.
`...`	extra arguments to pass to `mirror`.

Examples

```
data(FCWB.demo)
# Simple mirror along the x i.e. medio-lateral axis
kcs20.flip=mirror_brain(kcs20, FCWB.demo, transform='flip')

## full non-rigid mirroring to account for differences in shape/centering of
## template brain.
## Depends on nat.flybrains package and system CMTK installation
## Not run:
library(nat.flybrains)
kcs20.right=mirror_brain(kcs20, FCWB, .progress='text')
plot3d(kcs20, col='red')
plot3d(kcs20.right, col='green')
# include surface plot of brain
```

```
plot3d(FCWB)

# compare simple flip with full mirror
# this template brain is highly symmetric so these are almost identical
clear3d()
plot3d(kcs20.flip, col='blue')
plot3d(kcs20.right, col='green')

# convert to JFRC2 and do the same
kcs20.jfrc2=xform_brain(kcs20, sample = FCWB, reference=JFRC2, .progress='text')
kcs20.jfrc2.right=mirror_brain(kcs20.jfrc2, JFRC2, .progress='text')
kcs20.jfrc2.flip=mirror_brain(kcs20.jfrc2, JFRC2, transform='flip')
clear3d()
# this time there is a bigger difference between the two transformations
plot3d(kcs20.jfrc2.flip, col='blue')
plot3d(kcs20.jfrc2.right, col='green')
# plot mushroom body neuropils as well
plot3d(JFRC2NP.surf, "MB.*_R", alpha=0.3, col='grey')

# compare Euclidean distance between corresponding points in all neurons
diffs=xyzmatrix(kcs20.jfrc2.flip)-xyzmatrix(kcs20.jfrc2.right)
hist(sqrt(rowSums(diffs^2)), xlab='Distance /microns')

## End(Not run)
```

`plot3d.templatebrain` *Plot 3D surface of a template brain*

Description

Plot 3D surface of a template brain

Usage

```
## S3 method for class 'templatebrain'
plot3d(x, col = "grey", alpha = 0.3, ...)
```

Arguments

`x`	the template brain to plot.
`col`	the color of the surface.
`alpha`	the alpha value of the surface.
`...`	extra arguments to pass to `plot3d`.

Details

This function will work immediately for the standard templatebrain defined in the package documentation. If passed an object called e.g. FCWB it expects to find another object named FCWB.surf containing the surface information. If you follow this naming convention for user-defined refbrains it will work for them as well.

templatebrain *Construct templatebrain object for an image registration template*

Description

`templatebrain` objects encapsulate key information for the reference brain in an image registration. Usually this will be a standard template brain used for many registrations. **It will normally be much more convenient to use** `as.templatebrain` **methods to convert an image file or an im3d object into a** `templatebrain`.

Usage

```
templatebrain(name, regName = name, type = NULL, sex = NULL,
  dims = NULL, BoundingBox = NULL, voxdims = NULL, units = NULL,
  description = NULL, ...)
```

Arguments

`name`	the full name of the template.
`regName`	the short name. This will be the stem used to prefix registrations (e.g. JFRC2_someimage.list) for this template brain and likely also the stem of the template brain image (e.g. JFRC2.nrrd).
`type`	one of `c('single brain', 'average')`, indicating whether the template brain has been created from just one image, or is the average of multiple images.
`sex`	the sex of the template brain. For templates with `type=='average'`, the possibility of `sex='intersex'` exists.
`dims`	dimensions of the image (number of voxels)
`BoundingBox`	physical dimensions of the image (see `boundingbox`)
`voxdims`	physical spacing between voxels
`units`	units of physical measurements (e.g. microns)
`description`	details of the template.
`...`	additional named arguments that will be added as fields to the `templatebrain` object

Details

A variety of methods are availabe to work on templatebrain objects. See `templatebrain-meths` for basic methods. The two main functions that are availavle for using templatebrains are `xform_brain` and `mirror_brain`.

`templatebrain` objects are only useful for transformation processes when the `BoundingBox` is specified to define the physical extent of the volume. We use the definition of the Amira 3D visualisation and analysis software. This corresponds to the **node** centers option in the NRRD format. The bounding box can be obtained from nrrd or amiramesh format files. See `boundingbox` for details.

Value

A list with class templatebrain.

See Also

as.templatebrain, templatebrain-meths, xform_brain, mirror_brain.

---

templatebrain-meths *Template brain methods*

Description

is.templatebrain tests if object is of class templatebrain

as.character.templatebrain converts template brain to character vector representation (normally used to extract the short name i.e. regName).

print.templatebrain prints templatebrain information in human-readable form

as.im3d converts a template brain to a nat::im3d object; this is probably useful for developers.

origin extracts the space origin of a templatebrain object.

dim extracts the dimensions (in number of pixels) of the image associated with a templatebrain object.

voxdims extracts the dimensions (in number of pixels) of the image associated with a templatebrain object.

boundingbox extracts the boundingbox (in calibrated spatial units, typically microns) of the image associated with a templatebrain object. See boundingbox for details.

Usage

```
is.templatebrain(x)

## S3 method for class 'templatebrain'
as.character(x, field = c("regName", "name"), ...)

## S3 method for class 'templatebrain'
print(x, ...)

## S3 method for class 'templatebrain'
as.im3d(x, ...)

## S3 method for class 'templatebrain'
origin(x, ...)

## S3 method for class 'templatebrain'
dim(x, ...)
```

```
## S3 method for class 'templatebrain'
voxdims(x, ...)

## S3 method for class 'templatebrain'
boundingbox(x, ...)
```

Arguments

x	An object (usually a templatebrain)
field	Which field to use (defaults to 'regName')
...	additional arguments for methods

Value

logical

character vector

See Also

im3d

origin

voxdims

boundingbox

Examples

```
data(FCWB.demo)
is.templatebrain(FCWB.demo)
origin(FCWB.demo)
dim(FCWB.demo)
voxdims(FCWB.demo)
boundingbox(FCWB.demo)
```

---

xform_brain *Transform 3D object between template brains*

Description

Transform 3D object between template brains

Usage

```
xform_brain(x, sample, reference, via = NULL, ...)
```

Arguments

x	the 3D object to be transformed
sample	source template brain (e.g. IS2) that data is currently in.
reference	target template brain (e.g. IS2) that data should be transformed into.
via	optional intermediate brain to use when there is no direct bridging registration.
...	extra arguments to pass to xform.

Details

NB the sample and reference brains can either be templatebrain objects or a character string containing the short name of the template e.g. "IS2".

Examples

```
## depends on nat.flybrains package and system CMTK installation
## Not run:
library(nat.flybrains)
# Plot Kenyon cells in their original FCWB template brain
nopen3d()
plot3d(kcs20)
plot3d(FCWB)
# Convert to JFCR2 template brain
kcs20.jfrc2=xform_brain(kcs20, sample = FCWB, reference=JFRC2, .progress='text')
# now plot in the new JFRC2 space
nopen3d()
plot3d(kcs20.jfrc2)
plot3d(JFRC2)
# compare with the untransformed neurons
plot3d(kcs20)
# plot with neuropil sub regions for the left mushroom body
clear3d()
plot3d(kcs20.jfrc2)
# nb "MB.*_L" is a regular expression
plot3d(JFRC2NP.surf, "MB.*_L", alpha=0.3)
# compare with originals - briging registration is no perfect in peduncle
nopen3d()
plot3d(kcs20)
plot3d(FCWBNP.surf, "MB.*_L", alpha=0.3)


## End(Not run)
```

Index