

Package ‘voxel’

July 22, 2025

Title Mass-Univariate Voxelwise Analysis of Medical Imaging Data

Version 1.3.5

Description Functions for the mass-univariate voxelwise analysis of medical imaging data that follows the NIFTI <<http://nifti.nimh.nih.gov>> format.

Depends R (>= 3.2.3), lmerTest(>= 3.0-1)

License GPL-2

URL <https://github.com/angelgar/voxel>

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Imports parallel,stats,mgcv,gamm4,oro.nifti,ggplot2,methods,purrr

NeedsCompilation no

Author Angel Garcia de la Garza [aut, cre],
Simon Vandekar [aut],
David Roalf [aut],
Kosha Ruparel [aut],
Ruben Gur [aut],
Raquel Gur [aut],
Theodore Satterthwaite [aut],
R. Taki Shinohara [aut]

Maintainer Angel Garcia de la Garza <agarciadlg@gmail.com>

Repository CRAN

Date/Publication 2018-06-26 04:38:20 UTC

Contents

anovagammVoxel	2
anovagamVoxel	3
anovalmerVoxel	4
anovalmVoxel	5

gamCluster	6
gammCluster	7
gammNifti	8
gamNifti	10
gamRandomise	11
lmCluster	12
lmerCluster	13
lmerNifti	14
lmNifti	15
parMap	16
plotGAM	17
plotGAMM	18

Index	20
--------------	-----------

anovagammVoxel	<i>Computes voxelwise analysis of variance (ANOVA) tables for a Generalized Additive Mixed Effects Model.</i>
----------------	---

Description

This function computes analysis of variance tables for the fitted Generalized Additive Mixed Effects (from `gamm4::gamm4`) models. The analysis will run in all voxels in the specified mask and will return a list with the ANOVA table at each voxel. Please check the `mgcv::anova.gam` documentation for further information about specific arguments used in `anova.gam`. Multi-model calls are disabled.

Usage

```
anovagammVoxel(image, mask, fourdOut = NULL, formula, randomFormula, subjData,
  dispersion = NULL, freq = FALSE, mc.preschedule = TRUE, ncores = 1,
  ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call <code>mergeNifti()</code> and merge across time.
mask	Input mask of type 'nifti' or path to mask. Must be a binary.
fourdOut	To be passed to <code>mergeNifti</code> , This is the output path to write out the fourd file. Do not include a suffix (i.e. <code>.nii.gz</code>). Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to <code>gamm4()</code>
randomFormula	Random effects formula passed to <code>gamm4()</code>
subjData	Dataframe containing all the covariates used for the analysis
dispersion	To be passed to <code>mgcv::anova.gam</code> , Defaults to NULL. Dispersion Parameter, not normally used.

freq	To be passed to mgcv::anova.gam, Defaults to FALSE. Frequentist or Bayesian approximations for p-values
mc.preschedule	To be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use
...	Additional arguments passed to gamm4()

Value

Returns list of models fitted to each voxel over the masked images passed to function.

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim = c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(data = c(rep(0,15), rep(1,1)),
                                     dim = c(4,4,4,1)))

set.seed(1)
covs <- data.frame(x = runif(25), y=runif(25), id = rep(1:5,5))
f1 <- "~ s(x) + y"
randomFormula <- "~(1|id)"
models <- anovagamVoxel(image, mask, formula = f1,
                       randomFormula = randomFormula,
                       subjData = covs, ncores = 1, REML=TRUE)
```

anovagamVoxel	<i>Computes voxelwise analysis of variance (ANOVA) tables for a Generalized Additive Model.</i>
---------------	---

Description

This function computes analysis of variance tables for the fitted models after running a Generalized Additive Model (from mgcv::gam). The analysis will run in all voxels in the mask and will return the analysis of variance table for each voxel. Please check the mgcv::anova.gam documentation for further information about specific arguments used in anova.gam. Multi-model calls are disabled.

Usage

```
anovagamVoxel(image, mask, foundOut = NULL, formula, subjData,
              dispersion = NULL, freq = FALSE, mc.preschedule = TRUE, ncores = 1,
              ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call mergeNifti() and merge across time.
mask	Input mask of type 'nifti' or path to mask. Must be a binary mask

fourdOut	To be passed to mergeNifti, This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to gam()
subjData	Dataframe containing all the covariates used for the analysis
dispersion	To be passed to mgcv::anova.gam, Defaults to NULL. Dispersion Parameter, not normally used.
freq	To be passed to mgcv::anova.gam, Defaults to FALSE. Frequentist or Bayesian approximations for p-values
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use
...	Additional arguments passed to gam()

Value

Returns list of models fitted to each voxel over the masked images passed to function.

Examples

```
image <- oro.nifti::nifti(img = array(1:200, dim = c(2,2,2,25)))
mask <- oro.nifti::nifti(img = array(0:1, dim = c(2,2,2,1)))
set.seed(1)
covs <- data.frame(x = runif(25), y=runif(25))
fm1 <- "~ s(x) + y"
models <- anovagamVoxel(image=image, mask=mask,
                        formula=fm1, subjData=covs, ncores = 1)
```

anovalmerVoxel	<i>Computes voxelwise analysis of variance (ANOVA) tables for a Linear Mixed Effects Model.</i>
----------------	---

Description

This function computes analysis of variance tables for the fitted models after running a Linear Mixed Effect Model using the lmerTest() function and the anova function in that package. The analysis will run in all voxels in the mask and will return the analysis of variance table for each voxel. Please check the lmerTest documentation for further information about specific arguments used in anova.lmerModLmerTest. Multi-model calls are disabled.

Usage

```
anovalmerVoxel(image, mask, fourdOut = NULL, formula, subjData,
               ddf = "Satterthwaite", type = 3, mc.preschedule = TRUE, ncores = 1,
               ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call mergeNifti() and merge across time.
mask	Input mask of type 'nifti' or path to mask. Must be a binary mask
fourdOut	To be passed to mergeNifti, This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to lmer()
subjData	Dataframe containing all the covariates used for the analysis
ddf	Which approximation of DDF to be used. To be passed to anova.lmerModLmerTest. Defaults to "Satterthwaite"
type	Type of hypothesis to be test (defined from SAS terminology). Defaults to 3. To be passed to anova.lmerModLmerTest
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use
...	Additional arguments passed to lmer()

Value

Returns list of models fitted to each voxel over the masked images passed to function.

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim = c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(c(rep(0,15), rep(1,1)), dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25), y = runif(25), id = rep(1:5,5))
fm1 <- "~ x + y + (1|id)"
models <- anovaImVoxel(image, mask, formula = fm1, subjData = covs, ncores = 1, REML=TRUE)
```

anovaImVoxel	<i>Computes voxelwise analysis of variance (ANOVA) tables for a Linear Model.</i>
--------------	---

Description

This function computes analysis of variance tables for the fitted models after running a Linear Model using the stats::lm() function. The analysis will run in all voxels in the mask and will return the analysis of variance table for each voxel. Please check the stats documentation for further information about specific arguments used in stats::anova.lm(). Multi-model calls are disabled.

Usage

```
anovaImVoxel(image, mask, fourdOut = NULL, formula, subjData,
  mc.preschedule = TRUE, ncores = 1, ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will all mergeNifti() and merge across time.
mask	Input mask of type 'nifti' or path to mask. Must be a binary mask
fourdOut	To be passed to mergeNifti, This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to lm()
subjData	Dataframe containing all the covariates used for the analysis
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use
...	Additional arguments passed to lm()

Value

Returns list of models fitted to each voxel over the masked images passed to function.

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim =c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(0:1, dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25), y=runif(25))
fm1 <- "~ x + y"
models <- anovaImVoxel(image=image, mask=mask,
  formula=fm1, subjData=covs, ncores = 1)
```

gamCluster	<i>Run a Generalized Additive Model on the mean intensity over a region of interest</i>
------------	---

Description

This function is able to run a Generalized Additive Model (GAM) using the mgcv package. All clusters must be labeled with integers in the mask passed as an argument.

Usage

```
gamCluster(image, mask, fourdOut = NULL, formula, subjData,
  mc.preschedule = TRUE, ncores = 1, ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call mergeNiftis() and merge across time.
mask	Input mask of type 'nifti' or path to mask. All clusters must be labeled with integers in the mask passed as an argument
fourdOut	To be passed to mergeNifti, This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to gam()
subjData	Dataframe containing all the covariates used for the analysis
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use for the analysis
...	Additional arguments passed to gam()

Value

Returns list of models fitted to the mean voxel intensity over region of interest.

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim =c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(1:4, dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25))
fm1 <- "~ s(x)"
models <- gammCluster(image=image, mask=mask,
                      formula=fm1, subjData=covs, ncores = 1, method="REML")
```

gammCluster	<i>Run a Generalized Additive Mixed Effects Model on the mean intensity over a region of interest</i>
-------------	---

Description

This function is able to run a Generalized Additive Mixed Effects Model (GAMM) using the gamm4() function. All clusters or Regions of Interest must be labeled with integers in the mask passed as an argument.

Usage

```
gammCluster(image, mask, fourdOut = NULL, formula, randomFormula, subjData,
            mc.preschedule = TRUE, ncores = 1, ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call mergeNifti() and merge across time.
mask	Input mask of type 'nifti' or path to mask. All clusters must be labeled with integers in the mask passed as an argument
fourdOut	To be passed to mergeNifti, This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to gamm4()
randomFormula	Random effects formula passed to gamm4()
subjData	Dataframe containing all the covariates used for the analysis
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use for the analysis
...	Additional arguments passed to gamm4()

Value

Returns list of models fitted to the mean voxel intensity a region or interest.

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim = c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(c(rep(0,14),1,2), dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25), id = rep(1:5,5))
fm1 <- "~ s(x)"
randomFormula <- "~(1|id)"
models <- gammCluster(image, mask, formula = fm1,
                      randomFormula = randomFormula, subjData = covs, ncores = 1, REML=TRUE)
```

gammNIfTI	<i>Wrapper to run a Generalized Additive Mixed Effects model on an Nifti and output a parametric map</i>
-----------	--

Description

This function is able to run a Generalized Additive Model (GAMM) using the gamm4() function. The analysis will run in all voxels within the mask and will return parametric and smooth coefficients. The function will create parametric maps according to the model selected. The function will return a p-map, t-map, z-map, p-adjusted-map for parametric terms and p-map, z-map, p-adjusted-map for smooth terms. You can select which type of p-value correction you want done on the map

Usage

```
gammNIfTI(image, mask, fourdOut = NULL, formula, randomFormula, subjData,
  mc.preschedule = TRUE, ncores = 1, method = "none", residual = FALSE,
  outDir = NULL, ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call mergeNifti() and merge across time.
mask	Input mask of type 'nifti' or path to mask. Must be a binary mask
fourdOut	To be passed to mergeNifti, This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to gamm4()
randomFormula	Random effects formula passed to gamm4()
subjData	Dataframe containing all the covariates used for the analysis
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use
method	which method of correction for multiple comparisons (default is none)
residual	If set to TRUE then residuals maps will be returned along parametric maps
outDir	Path to the folder where to output parametric maps (Default is Null, only change if you want to write maps out)
...	Additional arguments passed to gamm4()

Value

Returns Parametric maps of the fitted models over the NIfTI image

Examples

```
image <- oro.nifti::nifti(img = array(rnorm(1600, sd=10), dim =c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(c(rep(0,14), rep(1,2)), dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25), y = runif(25), id = rep(1:5,5))
fm1 <- "~ s(x) + s(y)"
randomFormula <- "~(1|id)"
Maps <- gammNIfTI(image, mask, formula = fm1,
  randomFormula = randomFormula, subjData = covs, ncores = 1,
  method="fdr", REML=TRUE)
```

gamNIfTI	<i>Wrapper to run a Generalized Additive model on a NIfTI image and output parametric maps</i>
----------	--

Description

This function is able to run a Generalized Additive Model (GAM) using the mgcv package. The analysis will run in all voxels in in the mask and will return parametric and smooth coefficients. The function will create parametric maps according to the model selected. The function will return a p-map, t-map, z-map, p-adjusted-map for parametric terms and p-map, z-map, p-adjusted-map for smooth terms. You can select which type of p-value correction you want done on the map.

Usage

```
gamNIfTI(image, mask, fourdOut = NULL, formula, subjData,
  mc.preschedule = TRUE, ncores = 1, method = "none", residual = FALSE,
  outDir = NULL, ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call mergeNifti() and merge across time.
mask	Input mask of type 'nifti' or path to mask. Must be a binary mask
fourdOut	To be passed to mergeNifti, This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to gam()
subjData	Dataframe containing all the covariates used for the analysis
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use
method	which method of correction for multiple comparisons (default is none)
residual	If set to TRUE then residuals maps will be returned along parametric maps
outDir	Path to the folder where to output parametric maps (Default is Null, only change if you want to write maps out)
...	Additional arguments passed to gam()

Value

Parametric maps of the fitted models

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim = c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(0:1, dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25), y = rnorm(25))
fm1 <- "~ x + s(y)"
Maps <- gamNIfTI(image=image, mask=mask,
                 formula=fm1, subjData=covs, ncores = 1, method="fdr")
```

gamRandomise

Generate FSL Randomise call for a GAM Model

Description

This function is able to generate all the necessary files to run randomise with a GAM Model This script will write out all design and contrast files This function will run a f-test to compare a full and reduced model (a model with and without spline)

Usage

```
gamRandomise(image, maskPath = NULL, formulaFull, formulaRed, subjData,
             outDir, nsim = 500, thresh = 0.01, run = FALSE)
```

Arguments

image	Input path of 'nifti' image or vector of path(s) to images. If multiple paths, the script will all mergeNiftis() and merge across time.
maskPath	to mask. Must be a binary mask
formulaFull	Must be the formula of the full model (i.e. "~s(age,k=5)+sex+mprage_antsCT_vol_TBV")
formulaRed	Must be the formula of the reduced model (i.e. "~sex+mprage_antsCT_vol_TBV")
subjData	Dataframe containing all the covariates used for the analysis
outDir	output directory for randomise
nsim	Number of simulations
thresh	significance threshold
run	FALSE will only print randomise command but won't it

Value

Return randomise command

Examples

```
## Not run:

subjData = mgcv::gamSim(1,n=400,dist="normal",scale=2)
OutDirRoot="Output Directory"
maskName="Path to mask"
imagePath="Path to output"
covsFormula=~s(age,k=5)+sex+mprage_antsCT_vol_TBV"
redFormula=~sex+mprage_antsCT_vol_TBV"

gamRandomise(image = imagePath, maskPath = maskName, formulaFull = covsFormula,
              formulaRed = redFormula, subjData = subjData, outDir = OutDirRoot)

## End(Not run)
```

lmCluster

Run a Linear Model on the mean intensity over a region of interest

Description

This function is able to run a Linear Model using the stats package. All clusters must be labeled with integers in the mask passed as an argument.

Usage

```
lmCluster(image, mask, fourdOut = NULL, formula, subjData,
          mc.preschedule = TRUE, ncores = 1, ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call mergeNifti() and merge across time.
mask	Input mask of type 'nifti' or path to mask. All clusters must be labeled with integers in the mask passed as an argument
fourdOut	To be passed to mergeNifti. This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to lm()
subjData	Dataframe containing all the covariates used for the analysis
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use
...	Additional arguments passed to lm()

Value

Returns list of models fitted to the mean voxel intensity a region or interest.

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim =c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(1:4, dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25))
fm1 <- "~ x"
models <- lmCluster(image=image, mask=mask,
                    formula=fm1, subjData=covs, ncores = 1)
```

lmerCluster	<i>Run a Linear Mixed Effects Model on the mean intensity over a region of interest</i>
-------------	---

Description

This function is able to run a LME using the lmer() function. All clusters or region of interest must be labeled with integers in the mask passed as an argument. The function relies on lmerTest to create p-values using the Satterthwaite Approximation.

Usage

```
lmerCluster(image, mask, fourdOut = NULL, formula, subjData,
            mc.preschedule = TRUE, ncores = 1, ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call mergeNifti() and merge across time.
mask	Input mask of type 'nifti' or path to mask. All clusters must be labeled with integers in the mask passed as an argument
fourdOut	To be passed to mergeNifti, This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to lmer()
subjData	Dataframe containing all the covariates used for the analysis
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use
...	Additional arguments passed to lmer()

Value

Returns list of models fitted to the mean voxel intensity a region or interest.

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim = c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(c(rep(0,14),1,2), dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25), id = rep(1:5,5))
fm1 <- "~ x + (1|id)"
models <- lmerCluster(image, mask, formula = fm1, subjData = covs, ncores = 1, REML=TRUE)
```

lmerNIfTI

Run a Linear Mixed Effects Model on a NIfTI image and output a parametric maps

Description

This function is able to run a Linear Mixed Effect Model using the `lmer()` function. The function relies on `lmerTest` to create p-values using the Satterthwaite Approximation. The analysis will run in all voxels in in the mask and will return parametric coefficients. The function will create parametric maps according to the model selected. The function will return a p-map, t-map, z-map, p-adjusted-map for parametric terms and p-map, z-map, p-adjusted-map for smooth terms. You can select which type of p-value correction you want done on the map.

Usage

```
lmerNIfTI(image, mask, fourdOut = NULL, formula, subjData,
  mc.preschedule = TRUE, ncores = 1, method = "none", residual = FALSE,
  outDir = NULL, ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call <code>mergeNifti()</code> and merge across time.
mask	Input mask of type 'nifti' or path to mask. Must be a binary mask
fourdOut	To be passed to <code>mergeNifti</code> , This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to <code>lmer()</code>
subjData	Dataframe containing all the covariates used for the analysis
mc.preschedule	Argument to be passed to <code>mclapply</code> , whether or not to preschedule the jobs. More info in <code>parallel::mclapply</code>
ncores	Number of cores to use
method	which method of correction for multiple comparisons (default is none)
residual	If set to TRUE then residuals maps will be returned along parametric maps
outDir	Path to the folder where to output parametric maps (Default is Null, only change if you want to write maps out)
...	Additional arguments passed to <code>lmer()</code>

Value

Returns parametric maps of the fitted models

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim = c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(c(rep(0,14),1,1), dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25), id = rep(1:5,5))
fm1 <- "~ x + (1|id)"
Maps <- lmerNIfTI(image, mask, formula = fm1, subjData = covs, method="fdr", ncores = 1)
```

lmNIfTI

*Wrapper to run a model on a NIfTI image and output parametric maps***Description**

This function is able to run a Linear Model using the stats package. The analysis will run in all voxels in in the mask and will return parametric coefficients. The function will create parametric maps according to the model selected. The function will return a p-map, t-map, z-map, p-adjusted-map for parametric terms and p-map, z-map, p-adjusted-map for smooth terms. You can select which type of p-value correction you want done on the map.

Usage

```
lmNIfTI(image, mask, fourdOut = NULL, formula, subjData,
  mc.preschedule = TRUE, ncores = 1, method = "none", residual = FALSE,
  outDir = NULL, ...)
```

Arguments

image	Input image of type 'nifti' or vector of path(s) to images. If multiple paths, the script will call mergeNifti() and merge across time.
mask	Input mask of type 'nifti' or path to mask. Must be a binary mask
fourdOut	To be passed to mergeNifti, This is the path and file name without the suffix to save the fourd file. Default (NULL) means script won't write out 4D image.
formula	Must be a formula passed to lm()
subjData	Dataframe containing all the covariates used for the analysis
mc.preschedule	Argument to be passed to mclapply, whether or not to preschedule the jobs. More info in parallel::mclapply
ncores	Number of cores to use
method	which method of correction for multiple comparisons (default is none)
residual	If set to TRUE then residuals maps will be returned along parametric maps

outDir	Path to the folder where to output parametric maps (Default is Null, only change if you want to write maps out)
...	Additional arguments passed to lm()

Value

Return parametric maps of the fitted models

Examples

```
image <- oro.nifti::nifti(img = array(1:1600, dim = c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(0:1, dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25), y = runif(25))
fm1 <- "~ x + y"
Maps <- lmNIFTI(image=image, mask=mask,
               formula=fm1, subjData=covs, ncores = 1, method="fdr")
```

parMap

*Create parametric maps***Description**

This function create parametric maps according from model parametric tables or analysis of variance tables. The function will return a p-map, t-map, signed z-map, p-adjusted-map for parametric terms and p-map, z-map, p-adjusted-map for smooth terms. Additionally the function will return a p-map, F-map, p-to-z-map, and p-adjusted-map if the input is ANOVA. You can select which type of p-value correction you want done on the map. The z-maps are signed just like FSL.

Usage

```
parMap(parameters, mask, method = "none", outDir = NULL)
```

Arguments

parameters	list of parametric and smooth table coefficients or ANOVA (like the output from vlmParam, vgamParam, anovalmVoxel)
mask	Input mask of type 'nifti' or path to one. Must be a binary mask or a character. Must match the mask passed to one of vlmParam, vgamParam, vgammm4Param, vlmerParam
method	which method of correction for multiple comparisons (default is none)
outDir	Path to the folder where to output parametric maps (Default is Null, only change if you want to write maps out)

Value

Return parametric maps of the fitted models

Examples

```

image <- oro.nifti::nifti(img = array(1:1600, dim =c(4,4,4,25)))
mask <- oro.nifti::nifti(img = array(0:1, dim = c(4,4,4,1)))
set.seed(1)
covs <- data.frame(x = runif(25), y = runif(25))
fm1 <- "~ x + y"
models <- vlmParam(image=image, mask=mask,
                   formula=fm1, subjData=covs, ncores = 1)
Maps <- parMap(models, mask, method="fdr")

```

plotGAM

*GAM plotting using ggplot2***Description**

GAM plotting using ggplot2

Usage

```

plotGAM(gamFit, smooth.cov, groupCovs = NULL, orderedAsFactor = T,
        rawOrFitted = F, plotCI = T)

```

Arguments

gamFit	fitted gam model as produced by mgcv::gam()
smooth.cov	(character) name of smooth term to be plotted
groupCovs	(character) name of group variable to plot by, if NULL (default) then there are no groups in plot
orderedAsFactor	if TRUE then the model is refitted with ordered variables as factors.
rawOrFitted	If FALSE (default) then only smooth terms are plotted; if rawOrFitted = "raw" then raw values are plotted against smooth; if rawOrFitted = "fitted" then fitted values are plotted against smooth
plotCI	if TRUE (default) upper and lower confidence intervals are added at 2 standard errors above and below the mean

Value

Returns a ggplot object that can be visualized using the print() function

See Also

Other Plotting: [plotGAMM](#)

Examples

```

data <- data.frame(x = rep(1:20, 2), group = rep(1:2, each = 20))
set.seed(1)
data$y <- (data$x^2)*data$group*3 + rnorm(40, sd = 200)
data$group <- ordered(data$group)

gam <- mgcv::gam(y ~ s(x) + group, data=data)

plot1 <- plotGAM(gamFit = gam, smooth.cov = "x", groupCovs = NULL,
                 rawOrFitted = "raw", plotCI=TRUE, orderedAsFactor = FALSE)
gam <- mgcv::gam(y ~ s(x) + group + s(x, by=group), data=data)
plot2 <- plotGAM(gamFit = gam, smooth.cov = "x", groupCovs = "group",
                 rawOrFitted = "raw", orderedAsFactor = FALSE)

```

plotGAMM

*GAMM plotting using ggplot2***Description**

GAMM plotting using ggplot2

Usage

```

plotGAMM(gammFit, smooth.cov, groupCovs = NULL, orderedAsFactor = F,
         rawOrFitted = F, plotCI = T, grouping = NULL)

```

Arguments

gammFit	fitted gam model as produced by gamm4::gamm()
smooth.cov	(character) name of smooth term to be plotted
groupCovs	(character) name of group variable to plot by, if NULL (default) then there are no groups in plot
orderedAsFactor	Disabled
rawOrFitted	If FALSE (default) then only smooth terms are plotted; if rawOrFitted = "raw" then raw values are plotted against smooth; if rawOrFitted = "fitted" then fitted values are plotted against smooth
plotCI	if TRUE (default) upper and lower confidence intervals are added at 2 standard errors above and below the mean
grouping	(character) Name of variable that you want to use as the group argument in ggplot2::aes(), useful for better visualization of longitudinal data, (default is NULL)

Value

Returns a ggplot object that can be visualized using the print() function

See Also

Other Plotting: [plotGAM](#)

Examples

```
set.seed(1)
data <- data.frame(x = (seq(.25,25, .25) + rnorm(100)), group = rep(1:2, 5), z=rnorm(100),
                  index.rnorm = rep(rnorm(50, sd = 50), 2), index.var = rep(1:50, 2))
data$y <- (data$x)*data$group*10 + rnorm(100, sd = 700) + data$index.rnorm + data$z
data$group <- ordered(data$group)

gamm <- gamm4::gamm4(y ~ + s(x) + s(x, by=group) + z + group, data=data, random = ~ (1|index.var))

plot <- plotGAMM(gammFit <- gamm, smooth.cov <- "x", groupCovs = "group",
                plotCI <- T, rawOrFitted = "raw", grouping = "index.var")

plot2 <- plotGAMM(gammFit <- gamm, smooth.cov <- "x", groupCovs = "group",
                 plotCI <- T, rawOrFitted = "fitted", grouping = "index.var")
```

Index

[anovagammVoxel](#), [2](#)
[anovagamVoxel](#), [3](#)
[anovalmVoxel](#), [4](#)
[anovalmVoxel](#), [5](#)

[gamCluster](#), [6](#)
[gammCluster](#), [7](#)
[gammNifTI](#), [8](#)
[gamNifTI](#), [10](#)
[gamRandomise](#), [11](#)

[lmCluster](#), [12](#)
[lmerCluster](#), [13](#)
[lmerNifTI](#), [14](#)
[lmNifTI](#), [15](#)

[parMap](#), [16](#)
[plotGAM](#), [17](#), [19](#)
[plotGAMM](#), [17](#), [18](#)