Package 'jlmerclusterperm'

October 7, 2023

Title Cluster-Based Permutation Analysis for Densely Sampled Time Data

Version 1.1.0

Description An implementation of fast cluster-based permutation analysis (CPA) for densely-sampled time data developed in Maris & Oostenveld, 2007 <doi:10.1016/j.jneumeth.2007.03.024>. Supports (generalized, mixed-effects) regression models for the calculation of timewise statistics. Provides both a wholesale and a piecemeal interface to the CPA procedure with an emphasis on interpretability and diagnostics. Integrates 'Julia' libraries 'MixedModels.jl' and 'GLM.jl' for performance improvements, with additional functionalities for interfacing with 'Julia' from 'R' powered by the 'JuliaConnectoR' package.

License MIT + file LICENSE

URL https://github.com/yjunechoe/jlmerclusterperm, https://yjunechoe.github.io/jlmerclusterperm/

BugReports https://github.com/yjunechoe/jlmerclusterperm/issues

Depends R (>= 3.5)

Imports backports (>= 1.1.7), cli, generics, JuliaConnectoR, lme4, stats, tools, utils

Suggests broom, broom.mixed, covr, dplyr, eyetrackingR, forcats, future, ggplot2, knitr, MASS, patchwork, readr, rmarkdown, scales, testthat (>= 3.0.0), tibble

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.2.3

SystemRequirements Julia (>= 1.8)

Collate 'jlmerclusterperm-package.R' 'aaa.R' 'utils.R' 'interop-utils.R' 'interop-utils-unexported.R' 'julia_rng.R' 'jlmer_spec.R' 'jlmer.R' 'compute_timewise_statistics.R' 'permute.R' 'permute_timewise_statistics.R'

'clusters_methods.R' 'extract_clusters.R' 'calculate_pvalue.R'
'clusterpermute.R' 'threshold_search.R' 'tidy.R' 'zzz.R'
'srr-stats-standards.R'

NeedsCompilation no

Author June Choe [aut, cre, cph] (<https://orcid.org/0000-0002-0701-921X>)

Maintainer June Choe <jchoe001@gmail.com>

Repository CRAN

Date/Publication 2023-10-07 06:30:02 UTC

R topics documented:

`calculate_clusters_pvalues` *Calculate bootstrapped p-values of cluster-mass statistics*

Description

Calculate bootstrapped p-values of cluster-mass statistics

Usage

```
calculate_clusters_pvalues(
  empirical_clusters,
  null_cluster_dists,
  add1 = FALSE
)

clusters_are_comparable(empirical_clusters, null_cluster_dists, error = FALSE)
```

Arguments

empirical_clusters	A empirical_clusters object
null_cluster_dists	A null_cluster_dists object
add1	Whether to add 1 to the numerator and denominator when calculating the p-value. Use TRUE to effectively count the observed statistic as part of the permuted null distribution (recommended with larger nsim prior to publishing results).
error	Whether to throw an error if incompatible

Value

An empirical_clusters object augmented with p-values.

See Also

extract_empirical_clusters(), extract_null_cluster_dists()

Examples

```
library(dplyr, warn.conflicts = FALSE)

# Specification object
spec <- make_jlmer_spec(
  weight ~ 1 + Diet, filter(ChickWeight, Time <= 20),
  subject = "Chick", time = "Time"
)
spec

# Make empirical clusters
empirical_statistics <- compute_timewise_statistics(spec)
empirical_clusters <- extract_empirical_clusters(empirical_statistics, threshold = 2)
empirical_clusters

# Make null cluster-mass distribution
```

```
reset_rng_state()
null_statistics <- permute_timewise_statistics(spec, nsim = 100)
null_cluster_dists <- extract_null_cluster_dists(null_statistics, threshold = 2)

# Significance test the empirical cluster(s) from each predictor against the simulated null
calculate_clusters_pvalues(empirical_clusters, null_cluster_dists)

# Set `add1 = TRUE` to normalize by adding 1 to numerator and denominator
calculate_clusters_pvalues(empirical_clusters, null_cluster_dists, add1 = TRUE)

# This sequence of procedures is equivalent to `clusterpermute()`
reset_rng_state()
clusterpermute(spec, threshold = 2, nsim = 100, progress = FALSE)

# The empirical clusters and the null cluster-mass distribution must be comparable
empirical_clusters2 <- extract_empirical_clusters(empirical_statistics, threshold = 3)
# For example, below code errors because thresholds are different (2 vs. 3)
try( calculate_clusters_pvalues(empirical_clusters2, null_cluster_dists) )

# Check for compatibility with `clusters_are_comparable()`
clusters_are_comparable(empirical_clusters, null_cluster_dists)
clusters_are_comparable(empirical_clusters2, null_cluster_dists)
```

`clusterpermute` *Conduct a cluster-based permutation test*

Description

Conduct a cluster-based permutation test

Usage

```
clusterpermute(
  jlmer_spec,
  family = c("gaussian", "binomial"),
  statistic = c("t", "chisq"),
  threshold,
  nsim = 100L,
  predictors = NULL,
  binned = FALSE,
  top_n = Inf,
  add1 = TRUE,
  ...,
  progress = TRUE
)
```

Arguments

jlmer_spec	Data prepped for jlmer from make_jlmer_spec()
family	A GLM family. Currently supports "gaussian" and "binomial".
statistic	Test statistic for calculating cluster mass. Can be one of "t" (default) from the regression model output or "chisq" from a likelihood ratio test (takes about twice as long to calculate).
threshold	The threshold value that the statistic must pass to contribute to cluster mass. Interpretation differs on the choice of statistic (more below): • If statistic = "t", the threshold for t-value (beta/std.err) from the regression model. • If statistic = "chisq", the threshold for the p-value of chi-squared statistics from likelihood ratio tests.
nsim	Number of simulations description
predictors	(Optional) a subset of predictors to test. Defaults to NULL which tests all predictors.
binned	Whether the data has been aggregated/collapsed into time bins. Defaults to FALSE, which requires a cluster to span at least two time points. If TRUE, allows length-1 clusters to exist.
top_n	How many clusters to return, in the order of the size of the cluster-mass statistic. Defaults to Inf which return all detected clusters.
add1	Whether to add 1 to the numerator and denominator when calculating the p-value. Use TRUE to effectively count the observed statistic as part of the permuted null distribution (recommended with larger nsim prior to publishing results).
...	Optional arguments passed to Julia for model fitting. Defaults to fast = TRUE (when family = "binomial") and progress = FALSE.
progress	Defaults to TRUE, which prints progress on each step of the cluster permutation test.

Value

A list of null_cluster_dists and empirical_clusters with p-values

See Also

compute_timewise_statistics(), permute_timewise_statistics(), extract_empirical_clusters(), extract_null_cluster_dists(), calculate_clusters_pvalues()

Examples

```
library(dplyr, warn.conflicts = FALSE)
```

```
# Specification object
spec <- make_jlmer_spec(
  weight ~ 1 + Diet, filter(ChickWeight, Time <= 20),
  subject = "Chick", time = "Time"
)
spec

# Should minimally provide `threshold` and `nsim`, in addition to the spec object
reset_rng_state()
CPA <- clusterpermute(spec, threshold = 2, nsim = 100, progress = FALSE)
CPA

# CPA is a list of `<null_cluster_dists>` and `<empirical_clusters>` objects
sapply(CPA, class)

# You can extract the individual components for further inspection
CPA$null_cluster_dists
CPA$empirical_clusters
```

`cluster_permutation_tidiers` *Tidiers for cluster permutation test objects*

Description

Tidiers for cluster permutation test objects

Usage

```
## S3 method for class 'timewise_statistics'
tidy(x, ...)

## S3 method for class 'empirical_clusters'
tidy(x, ...)

## S3 method for class 'null_cluster_dists'
tidy(x, ...)
```

Arguments

`x`	An object of class `<timewise_statistics>`, `<empirical_clusters>`, or `<null_cluster_dists>`
`...`	Unused

Value

A data frame

Examples

```
library(dplyr, warn.conflicts = FALSE)

# Specification object
spec <- make_jlmer_spec(
  weight ~ 1 + Diet, filter(ChickWeight, Time <= 20),
  subject = "Chick", time = "Time"
)
spec

# Method for `<timewise_statistics>`
empirical_statistics <- compute_timewise_statistics(spec)
class(empirical_statistics)
tidy(empirical_statistics)

reset_rng_state()
null_statistics <- permute_timewise_statistics(spec, nsim = 100)
class(null_statistics)
tidy(null_statistics)

# Method for `<empirical_clusters>`
empirical_clusters <- extract_empirical_clusters(empirical_statistics, threshold = 2)
class(empirical_clusters)
tidy(empirical_clusters)

# Method for `<null_cluster_dists>`
null_cluster_dists <- extract_null_cluster_dists(null_statistics, threshold = 2)
class(null_cluster_dists)
tidy(null_cluster_dists)
```

---

compute_timewise_statistics

Fit Julia regression models to each time point of a time series data

---

Description

Fit Julia regression models to each time point of a time series data

Usage

```
compute_timewise_statistics(
  jlmer_spec,
  family = c("gaussian", "binomial"),
  statistic = c("t", "chisq"),
  ...
)
```

Arguments

`jlmer_spec`	Data prepped for jlmer from `make_jlmer_spec()`
`family`	A GLM family. Currently supports "gaussian" and "binomial".
`statistic`	Test statistic for calculating cluster mass. Can be one of `"t"` (default) from the regression model output or `"chisq"` from a likelihood ratio test (takes about twice as long to calculate).
`...`	Optional arguments passed to Julia for model fitting. Defaults to `fast = TRUE` (when `family = "binomial"`) and `progress = FALSE`.

Value

A predictor-by-time matrix of cluster statistics, of class `timewise_statistics`.

See Also

`jlmer()`, `make_jlmer_spec()`

Examples

```
library(dplyr, warn.conflicts = FALSE)

# Specification object
spec <- make_jlmer_spec(
  weight ~ 1 + Diet, filter(ChickWeight, Time <= 20),
  subject = "Chick", time = "Time"
)
spec

# Predictor x Time matrix of t-statistics from regression output
empirical_statistics <- compute_timewise_statistics(spec)
round(empirical_statistics, 2)

# Collect as dataframe with `tidy()`
empirical_statistics_df <- tidy(empirical_statistics)
empirical_statistics_df

# Timewise statistics are from regression models fitted to each time point
```

```
# - Note the identical statistics at `Time == 0`
empirical_statistics_df %>%
  filter(time == 0)
to_jlmer(weight ~ 1 + Diet, filter(ChickWeight, Time == 0)) %>%
  tidy() %>%
  select(term, statistic)
```

`extract_empirical_clusters` *Detect largest clusters from a time sequence of predictor statistics*

Description

Detect largest clusters from a time sequence of predictor statistics

Usage

```
extract_empirical_clusters(
  empirical_statistics,
  threshold,
  binned = FALSE,
  top_n = Inf
)
```

Arguments

`empirical_statistics`	A predictor-by-time matrix of empirical timewise statistics.
`threshold`	The threshold value that the statistic must pass to contribute to cluster mass. Interpretation differs on the choice of statistic (more below): • If `statistic = "t"`, the threshold for t-value (beta/std.err) from the regression model. • If `statistic = "chisq"`, the threshold for the p-value of chi-squared statistics from likelihood ratio tests.
`binned`	Whether the data has been aggregated/collapsed into time bins. Defaults to `FALSE`, which requires a cluster to span at least two time points. If `TRUE`, allows length-1 clusters to exist.
`top_n`	How many clusters to return, in the order of the size of the cluster-mass statistic. Defaults to `Inf` which return all detected clusters.

Value

An `empirical_clusters` object.

See Also

compute_timewise_statistics()

Examples

```
library(dplyr, warn.conflicts = FALSE)

# Specification object
spec <- make_jlmer_spec(
  weight ~ 1 + Diet, filter(ChickWeight, Time <= 20),
  subject = "Chick", time = "Time"
)
spec

# Empirical clusters are derived from the timewise statistics
empirical_statistics <- compute_timewise_statistics(spec)
empirical_clusters <- extract_empirical_clusters(empirical_statistics, threshold = 2)
empirical_clusters

# Collect as dataframe with `tidy()`
empirical_clusters_df <- tidy(empirical_clusters)
empirical_clusters_df

# Changing the `threshold` value identifies different clusters
extract_empirical_clusters(empirical_statistics, threshold = 1)

# A predictor can have zero or multiple clusters associated with it
extract_empirical_clusters(empirical_statistics, threshold = 3)
```

---

extract_null_cluster_dists

Construct a null distribution of cluster-mass statistics

Description

Construct a null distribution of cluster-mass statistics

Usage

```
extract_null_cluster_dists(null_statistics, threshold, binned = FALSE)
```

Arguments

`null_statistics`
: A simulation-by-time-by-predictor 3D array of null (permuted) timewise statistics.

`threshold`
: The threshold value that the statistic must pass to contribute to cluster mass. Interpretation differs on the choice of statistic (more below):

- If `statistic = "t"`, the threshold for t-value (beta/std.err) from the regression model.
- If `statistic = "chisq"`, the threshold for the p-value of chi-squared statistics from likelihood ratio tests.

`binned`
: Whether the data has been aggregated/collapsed into time bins. Defaults to `FALSE`, which requires a cluster to span at least two time points. If `TRUE`, allows length-1 clusters to exist.

Value

A `null_cluster_dists` object.

See Also

`permute_timewise_statistics()`

Examples

```
library(dplyr, warn.conflicts = FALSE)

# Specification object
spec <- make_jlmer_spec(
  weight ~ 1 + Diet, filter(ChickWeight, Time <= 20),
  subject = "Chick", time = "Time"
)
spec

# Null cluster-mass distributions are derived from the permuted timewise statistics
reset_rng_state()
null_statistics <- permute_timewise_statistics(spec, nsim = 100)
null_cluster_dists <- extract_null_cluster_dists(null_statistics, threshold = 2)
null_cluster_dists

# Collect as dataframe with `tidy()`
# - Each simulation contributes one (largest) cluster-mass statistic to the null
# - When no clusters are found, the `sum_statistic` value is zero
null_cluster_dists_df <- tidy(null_cluster_dists)
null_cluster_dists_df

# Changing the `threshold` value changes the shape of the null
```

```
extract_null_cluster_dists(null_statistics, threshold = 1)
extract_null_cluster_dists(null_statistics, threshold = 3)
```

`jlmer` *Fit a Julia regression model using jlmer specifications*

Description

Fit a Julia regression model using jlmer specifications

Usage

```
jlmer(jlmer_spec, family = c("gaussian", "binomial"), ..., progress = FALSE)
```

Arguments

`jlmer_spec`	Data prepped for jlmer from `make_jlmer_spec()`
`family`	A GLM family. Currently supports "gaussian" and "binomial".
`...`	Optional arguments passed to Julia for model fitting.
`progress`	If `TRUE`, prints the timing of iterations.

Value

A `jlmer_mod` object.

See Also

`make_jlmer_spec()`

Examples

```
jlmerclusterperm_setup(verbose = FALSE)

# Fitting a regression model with a specification object
spec <- make_jlmer_spec(weight ~ 1 + Diet, ChickWeight)
jlmer(spec)

# `lm()` equivalent
summary(lm(weight ~ 1 + Diet, ChickWeight))$coef
```

`jlmerclusterperm_setup` *Initial setup for the jlmerclusterperm package*

Description

Initial setup for the jlmerclusterperm package

Usage

```
jlmerclusterperm_setup(..., restart = TRUE, verbose = TRUE)
```

Arguments

`...`	Ignored.
`restart`	Whether to set up a fresh Julia session, given that one is already running. If `FALSE` and `jlmerclusterperm_setup()` has already been called, nothing happens.
`verbose`	Print progress and messages from Julia in the console

Value

TRUE

Examples

```
options("jlmerclusterperm.nthreads" = 2)
jlmerclusterperm_setup(verbose = FALSE)
```

`julia_model_tidiers` *Tidier methods for Julia regression models*

Description

Tidier methods for Julia regression models

Usage

```
## S3 method for class 'jlmer_mod'
tidy(x, effects = c("var_model", "ran_pars", "fixed"), ...)

## S3 method for class 'jlmer_mod'
glance(x, ...)
```

Arguments

`x`	An object of class `jlmer_mod`
`effects`	One of "var_model", "ran_pars", or "fixed"
`...`	Unused

Value

A data frame

Examples

```
# Fixed-effects only model
mod1 <- to_jlmer(weight ~ 1 + Diet, ChickWeight)
tidy(mod1)
glance(mod1)

# Mixed model
mod2 <- to_jlmer(weight ~ 1 + Diet + (1 | Chick), ChickWeight)
tidy(mod2)
glance(mod2)

# Select which of fixed/random effects to return
tidy(mod2, effects = "fixed")
tidy(mod2, effects = "ran_pars")
```

julia_progress *Set/get options for Julia progress bar*

Description

Set/get options for Julia progress bar

Usage

```
julia_progress(show, width)
```

Arguments

show	Whether to show the progress bar. You may also pass in a list of "show" and "width".
width	Width of the progress bar. If "auto", adjusts the progress bar width to fit the console.

Value

Previous values for show and width

Examples

```
# Show current progress options
julia_progress()

# Set options and save previous options
old_progress_opts <- julia_progress(show = FALSE, width = 100)
julia_progress()

# Restoring progress settings by passing a list of old options
old_progress_opts
julia_progress(old_progress_opts)
identical(julia_progress(), old_progress_opts)

# Alternatively, reset to default settings using this syntax:
julia_progress(show = TRUE, width = "auto")
```

julia_rng *Interface to the Julia RNG*

Description

Interface to the Julia RNG

Usage

```
set_rng_state(i)

reset_rng_state()

get_rng_state()

set_rng_seed(seed)

get_rng_seed()
```

Arguments

i	Counter number
seed	Seed

Value

The current seed or counter

Examples

```
# RNG initializes to seed=1 counter=0
get_rng_seed()
get_rng_state()

# setter/getter for RNG counter
set_rng_state(123)
get_rng_state()

# setter/getter for RNG seed
set_rng_seed(2)
get_rng_seed()

# restore to initial setting (seed=1, counter=0)
set_rng_seed(1)
set_rng_state(0)
```

julia_setup_ok *Check Julia setup requirements for jlmerclusterperm*

Description

Check Julia setup requirements for jlmerclusterperm

Usage

```
julia_setup_ok()
```

Value

Boolean

Examples

```
julia_setup_ok()
```

make_jlmer_spec *Create a specifications object for fitting regression models in Julia*

Description

Create a specifications object for fitting regression models in Julia

Usage

```
make_jlmer_spec(
  formula,
  data,
  subject = NULL,
  trial = NULL,
  time = NULL,
  drop_terms = NULL,
  ...
)
```

Arguments

formula	Model formula in R syntax
data	A data frame
subject	Column for subjects in the data.
trial	Column for trials in the data. Must uniquely identify a time series within subject (for example, the column for items in a counterbalanced design where each subject sees exactly one item).
time	Column for time in the data.
drop_terms	(Optional) any terms to drop from the reconstructed model formula
...	Unused, for extensibility.

Value

An object of class `jlmer_spec`.

Examples

```
# Bare specification object (minimal spec for fitting a global model)
spec <- make_jlmer_spec(weight ~ 1 + Diet, ChickWeight)
spec

# Constraints on specification for CPA:
# 1) The combination of `subject`, `trial`, and `time` must uniquely identify rows in the data
# 2) `time` must have constant sampling rate (i.e., evenly spaced values)
spec_wrong <- make_jlmer_spec(
  weight ~ 1 + Diet, ChickWeight,
  time = "Time"
)
unique(ChickWeight$Time)

# Corrected specification for the above
spec_correct <- make_jlmer_spec(
  weight ~ 1 + Diet, subset(ChickWeight, Time <= 20),
  subject = "Chick", time = "Time"
)
spec_correct
```

---

permute_by_predictor *Permute data while respecting grouping structure(s) of observations*

Description

Permute data while respecting grouping structure(s) of observations

Usage

```
permute_by_predictor(
  jlmer_spec,
  predictors,
  predictor_type = c("guess", "between_participant", "within_participant"),
  n = 1L
)
```

Arguments

jlmer_spec	Data prepped for jlmer from make_jlmer_spec()
predictors	A vector of terms from the model. If multiple, the must form the levels of one predictor.
predictor_type	Whether the predictor is "between_participant" or "within_participant". Defaults to "guess".
n	Number of permuted samples of the data to generate. Defaults to 1L.

Value

A long dataframe of permuted re-samples with .id column representing replication IDs.

Examples

```
# Example data setup
chickweights_df <- ChickWeight
chickweights_df <- chickweights_df[chickweights_df$Time <= 20, ]
chickweights_df$DietInt <- as.integer(chickweights_df$Diet)
head(chickweights_df)

# Example 1: Spec object using the continuous `DietInt` predictor
chickweights_spec1 <- make_jlmer_spec(
  formula = weight ~ 1 + DietInt,
  data = chickweights_df,
  subject = "Chick", time = "Time"
)
chickweights_spec1

# Shuffling `DietInt` values guesses `predictor_type = "between_participant"`
reset_rng_state()
spec1_perm1 <- permute_by_predictor(chickweights_spec1, predictors = "DietInt")
# This calls the same shuffling algorithm for CPA in Julia, so counter is incremented
get_rng_state()

# Shuffling under shared RNG state reproduces the same permutation of the data
reset_rng_state()
spec1_perm2 <- permute_by_predictor(chickweights_spec1, predictors = "DietInt")
```

```
identical(spec1_perm1, spec1_perm2)

# Example 2: Spec object using the multilevel `Diet` predictor
chickweights_spec2 <- make_jlmer_spec(
  formula = weight ~ 1 + Diet,
  data = chickweights_df,
  subject = "Chick", time = "Time"
)
chickweights_spec2

# Levels of a category are automatically shuffled together
reset_rng_state()
spec2_perm1 <- permute_by_predictor(chickweights_spec2, predictors = "Diet2")
reset_rng_state()
spec2_perm2 <- permute_by_predictor(chickweights_spec2, predictors = c("Diet2", "Diet3", "Diet4"))
identical(spec2_perm1, spec2_perm2)
```

`permute_timewise_statistics` *Simulate cluster-mass statistics via bootstrapped permutations*

Description

Simulate cluster-mass statistics via bootstrapped permutations

Usage

```
permute_timewise_statistics(
  jlmer_spec,
  family = c("gaussian", "binomial"),
  statistic = c("t", "chisq"),
  nsim = 100L,
  predictors = NULL,
  ...
)
```

Arguments

`jlmer_spec`	Data prepped for jlmer from `make_jlmer_spec()`
`family`	A GLM family. Currently supports "gaussian" and "binomial".
`statistic`	Test statistic for calculating cluster mass. Can be one of `"t"` (default) from the regression model output or `"chisq"` from a likelihood ratio test (takes about twice as long to calculate).
`nsim`	Number of simulations description

predictors	(Optional) a subset of predictors to test. Defaults to NULL which tests all predictors.
...	Optional arguments passed to Julia for model fitting. Defaults to fast = TRUE (when family = "binomial") and progress = FALSE.

Value

A simulation-by-time-by-predictor 3D array of cluster statistics, of class timewise_statistics.

See Also

make_jlmer_spec()

Examples

```
library(dplyr, warn.conflicts = FALSE)

# Specification object
spec <- make_jlmer_spec(
  weight ~ 1 + Diet, filter(ChickWeight, Time <= 20),
  subject = "Chick", time = "Time"
)
spec

# Simulation x Time x Predictor array of t-statistics from regression output
reset_rng_state()
null_statistics <- permute_timewise_statistics(spec, nsim = 3)
round(null_statistics, 2)

# Collect as dataframe with `tidy()`
permuted_timewise_stats_df <- tidy(null_statistics)
permuted_timewise_stats_df

# Permutations ran under the same RNG state are identical
reset_rng_state()
null_statistics2 <- permute_timewise_statistics(spec, nsim = 3)
identical(null_statistics, null_statistics2)

get_rng_state()
null_statistics3 <- permute_timewise_statistics(spec, nsim = 3)
identical(null_statistics, null_statistics3)
```

`to_jlmer` *Fit a Julia regression model using lme4 syntax*

Description

Fit a Julia regression model using lme4 syntax

Usage

```
to_jlmer(
  formula,
  data,
  family = c("gaussian", "binomial"),
  jlmer_spec_opts = list(),
  ...,
  progress = FALSE
)
```

Arguments

`formula`	Model formula in R syntax
`data`	A data frame
`family`	A GLM family. Currently supports "gaussian" and "binomial".
`jlmer_spec_opts`	List of options passed to `make_jlmer_spec()`
`...`	Optional arguments passed to Julia for model fitting.
`progress`	If `TRUE`, prints the timing of iterations.

Value

A `jlmer_mod` object.

See Also

`jlmer()`, `make_jlmer_spec()`

Examples

```
jlmerclusterperm_setup(verbose = FALSE)

# Fitting a regression model with R formula syntax
to_jlmer(weight ~ 1 + Diet, ChickWeight)
```

```
# `lm()` equivalent
summary(lm(weight ~ 1 + Diet, ChickWeight))$coef

# Fitting a mixed model with {lme4} syntax
to_jlmer(weight ~ 1 + Diet + (1 | Chick), ChickWeight)

# Passing MixedModels.jl fit options to the `...`
to_jlmer(weight ~ 1 + Diet + (1 | Chick), ChickWeight, REML = TRUE)
```

`walk_threshold_steps` *Test the probability of cluster-mass statistics over a range of threshold values*

Description

Test the probability of cluster-mass statistics over a range of threshold values

Usage

```
walk_threshold_steps(
  empirical_statistics,
  null_statistics,
  steps,
  top_n = Inf,
  binned = FALSE,
  add1 = TRUE,
  progress = TRUE
)
```

Arguments

`empirical_statistics`	A predictor-by-time matrix of empirical timewise statistics.
`null_statistics`	A simulation-by-time-by-predictor 3D array of null (permuted) timewise statistics.
`steps`	A vector of threshold values to test
`top_n`	How many clusters to return, in the order of the size of the cluster-mass statistic. Defaults to `Inf` which return all detected clusters.
`binned`	Whether the data has been aggregated/collapsed into time bins. Defaults to `FALSE`, which requires a cluster to span at least two time points. If `TRUE`, allows length-1 clusters to exist.

add1	Whether to add 1 to the numerator and denominator when calculating the p-value. Use TRUE to effectively count the observed statistic as part of the permuted null distribution (recommended with larger nsim prior to publishing results).
progress	Whether to display a progress bar

Value

A data frame of predictor clusters-mass statistics by threshold.

Examples

```
# Specification object
spec <- make_jlmer_spec(
  weight ~ 1 + Diet, subset(ChickWeight, Time <= 20),
  subject = "Chick", time = "Time"
)
spec

# Compute timewise statistics for the observed and permuted data
empirical_statistics <- compute_timewise_statistics(spec)
null_statistics <- permute_timewise_statistics(spec, nsim = 100)

# Test cluster mass/probability under different threshold values
walk_threshold_steps(empirical_statistics, null_statistics, steps = 1:3,
                     progress = FALSE)
```

Index