Package ‘jlmerclusterperm’

June 27, 2023

Title  Cluster-Based Permutation Analysis for Densely Sampled Time Data

Version  1.0.0

Description  An implementation of fast cluster-based permutation analysis (CPA) for densely-sampled time data developed in Maris & Oostenveld, 2007 <doi:10.1016/j.neumeth.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  cli, generics, JuliaConnectoR, lme4, parallel, stats, 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'
calculate_clusters_pvalues

Calculate bootstrapped p-values of cluster-mass statistics

Description

Calculate bootstrapped p-values of cluster-mass statistics

Usage

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

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

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

equal_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

# 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**

```r
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"
)

# 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

```r
## 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(
compute_timewise_statistics

```r
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**

```r
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 `\texttt{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 \texttt{Time == 0}
empirical_statistics_df %>%
  filter(time == 0)
```
extract_empirical_clusters

Detect largest clusters from a time sequence of predictor statistics

desc

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

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

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

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

# Collect as dataframe with `\(\text{tidy}\)`
empirical_clusters_df <- tidy(empirical_clusters)

# 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**

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

**Arguments**

- `null_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)
**Fit a Julia regression model using jlmer specifications**

**Description**

Fit a Julia regression model using jlmer specifications

**Usage**

```r
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**

```r
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**

```r
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

```r
## 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

```r
# 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")
```
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_rngInterface 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)
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,
permute_by_predictor

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
permute_timewise_statistics

Simulate cluster-mass statistics via bootstrapped permutations

Description

Simulate cluster-mass statistics via bootstrapped permutations
permute_timewise_statistics

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)
to_jlmer

Fit a Julia regression model using lme4 syntax

**Description**

Fit a Julia regression model using lme4 syntax

**Usage**

```r
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.
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

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

**Arguments**

- **empirical_statistics**
  A predictor-by-time matrix of empirical timewise statistics.

- **null_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**

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

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

# Compute timewise statistics for the observed and permuted data
empirical_statistics <- compute_timewise_statistics(spec)
reset_rng_state()
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)
```
Index

calculate_clusters_pvalues, 2
calculate_clusters_pvalues(), 5
cluster_permutation_tidiers, 6
clusterpermute, 4
clusters_are_comparable
   (calculate_clusters_pvalues), 2
compute_timewise_statistics, 7
compute_timewise_statistics(), 5, 9

extract_empirical_clusters, 9
extract_empirical_clusters(), 3, 5
extract_null_cluster_dists, 10
extract_null_cluster_dists(), 3, 5

generate_rng_seed(julia_rng), 15
generate_rng_state(julia_rng), 15
glance.jlmer_mod(julia_model_tidiers), 13

tidy.jlmer_mod(julia_model_tidiers), 13
tidy.null_cluster_dists
   (cluster_permutation_tidiers), 6
tidy.timewise_statistics
   (cluster_permutation_tidiers), 6
tojlmer, 21
walk_threshold_steps, 22

jlmer, 12
jlmer(), 8, 22
jlmerclusterperm_setup, 13
julia_model_tidiers, 13
julia_progress, 14
julia_rng, 15

make_jlmer_spec, 17
make_jlmer_spec(), 8, 12, 20, 22

permute_by_predictor, 18
permute_timewise_statistics, 19
permute_timewise_statistics(), 5, 11
reset_rng_state(julia_rng), 15

set_rng_seed(julia_rng), 15
set_rng_state(julia_rng), 15

tidy.empirical_clusters
   (cluster_permutation_tidiers), 6