Package 'GET'

November 16, 2022

Version 0.3-2

Date 2022-11-16

Title Global Envelopes

Encoding UTF-8

Maintainer Mari Myllymäki <mari.myllymaki@luke.fi>

Imports cluster, ggplot2, gridExtra, parallel, stats, utils, viridisLite

Suggests crayon, geoR, gstat, sp, fda, fda.usc, locfit, mvtnorm, patchwork, spatstat.geom, spatstat.explore, spatstat.model, testthat, R.rsp

Description Implementation of global envelopes for a set of general d-dimensional vectors T in various applications. A 100(1-alpha)% global envelope is a band bounded by two vectors such that the probability that T falls outside this envelope in any of the d points is equal to alpha. Global means that the probability is controlled simultaneously for all the d elements of the vectors. The global envelopes can be used for graphical Monte Carlo and permutation tests where the test statistic is a multivariate vector or function (e.g. goodness-of-fit testing for point patterns and random sets, functional analysis of variance, functional general linear model, n-sample test of correspondence of distribution functions), for central regions of functional or multivariate data (e.g. outlier detection, functional boxplot) and for global confidence and prediction bands (e.g. confidence band in polynomial regression, Bayesian posterior prediction). See Myllymäki and Mrkvička (2020) <arXiv:1911.06583>, Myllymäki et al. (2017) <doi:10.1111/rssb.12172>, Mrkvička and Myllymäki (2022) <arXiv:2008.10108>, Mrkvička et al. (2017) <doi:10.1007/s11222-016-9683-9>, Mrkvička et al. (2020) <doi:10.14736/kyb-2020-3-0432>, Mrkvička et al. (2021) <doi:10.1007/s11009-019-09756-y>, Mrkvička et al. (2022) <doi:10.1002/sim.9236>, Mrkvička et al. (2016) <doi:10.1016/j.spasta.2016.04.005>, Myllymäki et al. (2021) <doi:10.1016/j.spasta.2020.100436>, Dai et al. (2022) <doi:10.5772/intechopen.100124>, and Dvořák and Mrkvička (2022) <doi:10.1007/s00180-021-01134-y>.

License GPL-3

RoxygenNote 7.2.1

VignetteBuilder R.rsp

NeedsCompilation no

Author Mari Myllymäki [aut, cre],
Tomáš Mrkvička [aut],
Jiří Dvořák [ctb],
Pavel Grabarnik [ctb],
Ute Hahn [ctb],
Mikko Kuronen [ctb],
Michael Rost [ctb],
Henri Seijo [ctb]

Depends R (>= 2.10)

Repository CRAN

Date/Publication 2022-11-16 15:40:25 UTC

R topics documented:

`GET-package` *Global Envelopes*

Description

The **GET** package provides implementation of global envelopes for a set of general d-dimensional vectors T in various applications. A 100(1-alpha) the probability that T falls outside this envelope in any of the d points is equal to alpha. Global means that the probability is controlled simultaneously for all the d elements of the vectors. The global envelopes can be used for central regions of functional or multivariate data (e.g. outlier detection, functional boxplot), for graphical Monte Carlo and permutation tests where the test statistic is a multivariate vector or function (e.g. goodness-of-fit testing for point patterns and random sets, functional ANOVA, functional GLM, n-sample test of correspondence of distribution functions), and for global confidence and prediction bands (e.g. confidence band in polynomial regression, Bayesian posterior prediction).

Details

The **GET** package provides central regions (i.e. global envelopes) and global envelope tests with intrinsic graphical interpretation. The central regions can be constructed from (functional) data. The tests are Monte Carlo or permutation tests, which demand simulations from the tested null model. The methods are applicable for any multivariate vector data and functional data (after discretization).

To get an overview of the package, start R and type `library("GET")` and `vignette("GET")`.

To get examples of point pattern analysis, start R and type `library("GET")` and `vignette("pointpatterns")`.

Key functions in GET

- *Central regions* or *global envelopes* or *confidence bands*: `central_region`. E.g. 50% central region of growth curves of girls `growth`.
  - First create a curve_set of the growth curves, e.g.
    `cset <- create_curve_set(list(r = as.numeric(row.names(growth$hgtf)), obs = growth$hgtf))`
  - Then calculate 50% central region (see `central_region` for further arguments)
    `cr <- central_region(cset, coverage = 0.5)`
  - Plot the result (see `plot.global_envelope` for plotting options)
    `plot(cr)`

  It is also possible to do combined central regions for several sets of curves provided in a list for the function, see examples in `central_region`.
- *Global envelope tests*: `global_envelope_test` is the main function. E.g. A test of complete spatial randomness (CSR) for a point pattern `X`:
  `X <- spruces # an example pattern from spatstat`
  - Use the function envelope of **spatstat** to create nsim simulations under CSR and to calculate the functions you want (below K-functions by Kest). Important: use the option 'savefuns=TRUE' and specify the number of simulations `nsim`.
    `env <- envelope(X, nsim=999, savefuns = TRUE, fun = Kest, simulate = expression(runifpoint(ex = X)))`
  - Perform the test (see `global_envelope_test` for further arguments)
    `res <- global_envelope_test(env)`
  - Plot the result (see `plot.global_envelope` for plotting options)
    `plot(res)`

  It is also possible to do combined global envelope tests for several sets of curves provided in a list for the function, see examples in `global_envelope_test`.
- *Functional ordering*: `central_region` and `global_envelope_test` are based on different measures for ordering the functions (or vectors) from the most extreme to the least extreme ones. The core functionality of calculating the measures is in the function `forder`, which can be used to obtain different measures for sets of curves. Usually there is no need to call `forder` directly.
- *Functional boxplots*: `fBoxplot`
- *Adjusted* global envelope tests for composite null hypotheses

  - `GET.composite`, see a detailed example in `saplings`
- *One-way functional ANOVA*:
  - *Graphical* functional ANOVA tests: `graph.fanova`
  - Global rank envelope based on F-values: `frank.fanova`
- *Functional general linear model (GLM)*:
  - *Graphical* functional GLM: `graph.flm`
  - Global rank envelope based on F-values: `frank.flm`
  - For large data (not fitting comfortably in memory): `partial_forder`
- *Functional clustering*: `fclustering`
- Functions for performing global envelopes for specific purposes:
  - Graphical n sample test of correspondence of distribution functions: `GET.necdf`
  - Permutation-based tests of independence to samples from any bivariate distribution:
    * based on cumulative distribution function `GET.cdf`
    * in a 2D contingency table `GET.contingency`
    * based on the smoothed Q-Q plot `GET.qq`
  - Testing global and local dependence of point patterns on covariates: `GET.spatialF`
  - Testing local correlations: `GET.localcor`
  - Variogram and residual variogram with global envelopes: `GET.variogram`
- Deviation tests (for simple hypothesis): `deviation_test` (no graphical interpretation)
- Most functions accept the curves provided in a `curve_set` object. Use `create_curve_set` to create a `curve_set` object from the functions. Other formats to provide the curves to the above functions are also accepted, see the information on the help pages.

See the help files of the functions for examples.

Workflow for (single hypothesis) tests based on single functions

To perform a test you always first need to obtain the test function $T(r)$ for your data ($T_1(r)$) and for each simulation ($T_2(r), \dots, T_{s+1}(r)$) in one way or another. Given the set of the functions $T_i(r), i = 1, \dots, s+1$, you can perform a test by `global_envelope_test`.

1) The workflow when using your own programs for simulations:

- (Fit the model and) Create s simulations from the (fitted) null model.
- Calculate the functions $T_1(r), T_2(r), \dots, T_{s+1}(r)$.
- Use `create_curve_set` to create a `curve_set` object from the functions $T_i(r), i = 1, \dots, s+1$.
- Perform the test
  `res <- global_envelope_test(curve_set)`
  where `curve_set` is the 'curve_set'-object you created, and plot the result
  `plot(res)`

2) The workflow utilizing **spatstat**: start R, type `library("GET")` and `vignette("pointpatterns")`, which explains the workflow and gives many examples of point pattern analysis

Functions for modifying sets of functions

It is possible to modify the curve set $T_1(r), T_2(r), \dots, T_{s+1}(r)$ for the test.

- You can choose the interval of distances $[r_{\min}, r_{\max}]$ by `crop_curves`.
- For better visualisation, you can take $T(r) - T_0(r)$ by `residual`. Here $T_0(r)$ is the expectation of $T(r)$ under the null hypothesis.

Example data (see references on the help pages of each data set)

- `abide_9002_23`: see help page
- `adult_trees`: a point pattern of adult rees
- `cgec`: centred government expenditure centralization (GEC) ratios (see `graph.fanova`)
- `fallen_trees`: a point pattern of fallen trees
- `GDPtax`: GDP per capita with country groups and other covariates
- `imageset3`: a simulated set of images
- `rimov`: water temperature curves in 365 days of the 36 years
- `saplings`: a point pattern of saplings (see `GET.composite`)

The data sets are used to show examples of the functions of the library.

Number of functions

If the number of functions is low, the choice of the measure (or type or depth) playes a role, as explained in `vignette("GET")` (Section 2.4).

Note that the recommended minimum number of simulations for the rank envelope test (Myllymäki et al., 2017) based on a single function in spatial statistics is nsim=2499. When the number of argument values is large, also larger number simulations is needed in order to have a narrow p-interval. The "erl", "cont", "area", "qdir" and "st" global envelope tests and deviation tests can be used with a lower number of simulations, although the Monte Carlo error is obviously larger with a lower number of simulations. For increasing the number of simulations, all the global rank envelopes approach the same curves.

Mrkvička et al. (2017) discussed the number of simulations for tests based on many functions.

Documentation

Myllymäki and Mrkvička (2020) provides description of the package. The material can also be found in the corresponding vignette, which is available by starting R and typing `library("GET")` and `vignette("GET")`.

In the special case of spatial processes (spatial point processes, random sets), the functions are typically estimators of summary functions. The package supports the use of the R package **spatstat** for generating simulations and calculating estimators of the chosen summary function, but alternatively these can be done by any other way, thus allowing for any user-specified models/functions. To see examples of global envelopes for analysing point pattern data, start R, type `library("GET")` and `vignette("pointpatterns")`.

Type citation("GET") to get a full list of references.

Acknowledgements

Mikko Kuronen has made substantial contributions of code. Additional contributions and suggestions from Jiří Dvořák, Pavel Grabarnik, Ute Hahn, Michael Rost and Henri Seijo.

Author(s)

Mari Myllymäki (mari.myllymaki@luke.fi, mari.j.myllymaki@gmail.com) and Tomáš Mrkvička (mrkvicka.toma@gmail.com)

References

Dai, W., Athanasiadis, S., Mrkvička, T. (2021) A new functional clustering method with combined dissimilarity sources and graphical interpretation. Intech open, London, UK. DOI: 10.5772/intechopen.100124

Dvořák, J. and Mrkvička, T. (2022). Graphical tests of independence for general distributions. Computational Statistics 37, 671–699.

Mrkvička, T., Myllymäki, M. and Hahn, U. (2017) Multiple Monte Carlo testing, with applications in spatial point processes. Statistics & Computing 27(5), 1239-1255. doi: 10.1007/s11222-016-9683-9

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2020) A one-way ANOVA test for functional data with graphical interpretation. Kybernetika 56(3), 432-458. doi: 10.14736/kyb-2020-3-0432

Mrkvička, T., Myllymäki, M., Kuronen, M. and Narisetty, N. N. (2022) New methods for multiple testing in permutation inference for the general linear model. Statistics in Medicine 41(2), 276-297. doi: 10.1002/sim.9236

Mrkvička, T., Myllymäki, M. False discovery rate envelopes. arXiv:2008.10108 [stat.ME]

Mrkvička, T., Roskovec, T. and Rost, M. (2021) A nonparametric graphical tests of significance in functional GLM. Methodology and Computing in Applied Probability 23, 593-612. doi: 10.1007/s11009-019-09756-y

Mrkvička, T., Soubeyrand, S., Myllymäki, M., Grabarnik, P., and Hahn, U. (2016) Monte Carlo testing in spatial statistics, with applications to spatial residuals. Spatial Statistics 18, Part A, 40-53. doi: http://dx.doi.org/10.1016/j.spasta.2016.04.005

Myllymäki, M., Grabarnik, P., Seijo, H. and Stoyan. D. (2015) Deviation test construction and power comparison for marked spatial point patterns. Spatial Statistics 11, 19-34. doi: 10.1016/j.spasta.2014.11.004

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017) Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 79, 381-404. doi: 10.1111/rssb.12172

Myllymäki, M. and Mrkvička, T. (2020). GET: Global envelopes in R. arXiv:1911.06583 [stat.ME]

Myllymäki, M., Kuronen, M. and Mrkvička, T. (2020). Testing global and local dependence of point patterns on covariates in parametric models. Spatial Statistics 42, 100436. doi: 10.1016/j.spasta.2020.100436

`abide_9002_23` *Local brain activity at resting state*

Description

Imaging measurements for local brain activity at resting state

Usage

```
data("abide_9002_23")
```

Format

A list of the `curve_set` containing the data, coordinates (x,y) where the data have been observed (third dimension is 23), the discrete factor `Group` (1=Autism; 2=Control), the discrete factor `Sex` (1=Male; 2=Female), and the continuous factor `Age`.

Details

The data are a small part of ABIDE fALFF data available at ABIDE: http://fcon_1000.projects.nitrc.org/indi/abide/ fALFF: http://fcp-indi.github.io/docs/user/alff.html and distributed under the CC BY-NC-SA 3.0 license, https://creativecommons.org/licenses/by-nc-sa/3.0/.

The data are fractional Amplitude of Low Frequency Fluctuations (fALFF) (Zou et al. 2008) for Autism Brain Imaging Data Exchange collected resting state functional magnetic resonance imaging (R-fMRI) datasets (Di Martino et al. 2013). This data set in **GET** contains only a tiny part of the whole brain, namely the region 9002 (the right Cerebelum Crus 1) at slice 23 (see Figure 2 in Mrkvicka et al., 2019) for 514 individuals with the autism spectrum disorder (ASD) and 557 typical controls (TC) as specified in the given Group variable. Further the sex and age of each subject is given.

References

Di Martino, A., Yan, C., Li, Q., Denio, E., Castellanos, F., Alaerts, K., Anderson, J., Assaf, M., Bookheimer, S., Dapretto, M., et al. (2013) The autism brain imaging data exchange: towards a large-scale evaluation of the intrinsic brain architecture in autism. Molecular psychiatry.

Tzourio-Mazoyer, N., Landeau, B., Papathanassiou, D., Crivello, F., Etard, O., Delcroix, N., Mazoyer, B., and Joliot, M. (2002), Automated anatomical labeling of activations in SPM using a macroscopic anatomical parcellation of the MNI MRI single-subject brain. Neuroimage, 15, 273-289.

Zou, Q.-H., Zhu, C.-Z., Yang, Y., Zuo, X.-N., Long, X.-Y., Cao, Q.-J., Wang, Y.-F., and Zang, Y.-F. (2008), An improved approach to detection of amplitude of low-frequency fluctuation (ALFF) for resting-state fMRI: fractional ALFF. Journal of neuroscience methods, 172, 137-141.

Mrkvička, T., Myllymäki, M., Kuronen, M. and Narisetty, N. N. (2022) New methods for multiple testing in permutation inference for the general linear model. Statistics in Medicine 41(2), 276-297. doi: 10.1002/sim.9236

`adult_trees` *Adult trees data set*

Description

Adult trees data set

Usage

```
data("adult_trees")
```

Format

A `data.frame` containing the locations (x- and y-coordinates) of 67 trees in an area of 75 m x 75 m.

Details

A pattern of large trees (height > 25 m) originating from an uneven aged multi-species broadleaf nonmanaged forest in Kaluzhskie Zaseki, Russia.

The pattern is a sample part of data collected over 10 ha plot as a part of a research program headed by project leader Prof. O.V. Smirnova.

References

Grabarnik, P. and Chiu, S. N. (2002) Goodness-of-fit test for complete spatial randomness against mixtures of regular and clustered spatial point processes. Biometrika, 89, 411–421.

van Lieshout, M.-C. (2010) Spatial point process theory. In Handbook of Spatial Statistics (eds. A. E. Gelfand, P. J. Diggle, M. Fuentes and P. Guttorp), Handbooks of Modern Statistical Methods. Boca Raton: CRC Press.

See Also

saplings

Examples

```
if(require("spatstat.geom", quietly=TRUE)) {
  data("adult_trees")
  adult_trees <- as.ppp(adult_trees, W = square(75))
  plot(adult_trees)
}
```

`central_region` *Central region / Global envelope*

Description

Provides central regions or global envelopes or confidence bands

Usage

```
central_region(
  curve_sets,
  type = "erl",
  coverage = 0.5,
  alternative = c("two.sided", "less", "greater"),
  probs = c(0.25, 0.75),
  quantile.type = 7,
  central = "median",
  nstep = 2,
  ...
)
```

Arguments

`curve_sets`	A `curve_set` object or a list of `curve_set` objects.
`type`	The type of the global envelope with current options for 'rank', 'erl', 'cont', 'area', 'qdir', 'st' and 'unscaled'. See details.
`coverage`	A number between 0 and 1. The 100*coverage% central region will be calculated. A vector of values can also be provided, leading to the corresponding number of central regions.
`alternative`	A character string specifying the alternative hypothesis. Must be one of the following: "two.sided" (default), "less" or "greater". The last two options only available for types `'rank'`, `'erl'`, `'cont'` and `'area'`.
`probs`	A two-element vector containing the lower and upper quantiles for the measure 'q' or 'qdir', in that order and on the interval [0, 1]. The default values are 0.025 and 0.975, suggested by Myllymäki et al. (2015, 2017).
`quantile.type`	As type argument of `quantile`, how to calculate quantiles for 'q' or 'qdir'.
`central`	Either "mean" or "median". If the curve sets do not contain the component `theo` for the theoretical central function, then the central function (used for plotting only) is calculated either as the mean or median of functions provided in the curve sets. For 'qdir', 'st' and 'unscaled' only the mean is allowed as an option, due to their definition.
`nstep`	1 or 2 for how to contruct a combined global envelope if list of curve sets is provided. 2 (default) for a two-step combining procedure, 1 for one-step.
`...`	Ignored.

Details

Given a curve_set (see create_curve_set for how to create such an object) or an envelope object of **spatstat** or fdata object of **fda.usc**, the function central_region construcst a central region, i.e. a global envelope, from the given set of functions (or vectors).

Generally an envelope is a band bounded by the vectors (or functions) T_{low} and T_{hi}. A $100(1-\alpha)\%$ or 100*coverage% global envelope is a set (T_{low}, T_{hi}) of envelope vectors such that the probability that T_i falls outside this envelope in any of the d points of the vector T_i is less or equal to α. The global envelopes can be constructed based on different measures that order the functions from the most extreme one to the least extreme one. We use such orderings of the functions for which we are able to construct global envelopes with intrinsic graphical interpretation.

The type of the global envelope can be chosen with the argument type and the options are given in the following. Further information about the measures, on which the global envelopes are based, can be found in Myllymäki and Mrkvička (2020, Section 2.).

- 'rank': The global rank envelope proposed by Myllymäki et al. (2017) based on the extreme rank defined as the minimum of pointwise ranks.
- 'erl': The global rank envelope based on the extreme rank length (Myllymäki et al.,2017, Mrkvička et al., 2018). This envelope is constructed as the convex hull of the functions which have extreme rank length measure that is larger or equal to the critical α level of the extreme rank length measure.
- 'cont': The global rank envelope based on the continuous rank (Hahn, 2015; Mrkvička et al., 2019) based on minimum of continuous pointwise ranks. It is contructed as the convex hull in a similar way as the 'erl' envelope.
- 'area': The global rank envelope based on the area rank (Mrkvička et al., 2019) which is based on area between continuous pointwise ranks and minimum pointwise ranks for those argument (r) values for which pointwise ranks achieve the minimum (it is a combination of erl and cont). It is contructed as the convex hull in a similar way as the 'erl' and 'area' envelopes.
- 'qdir': The directional quantile envelope based on the directional quantile maximum absolute deviation (MAD) test (Myllymäki et al., 2017, 2015), which takes into account the unequal variances of the test function T(r) for different distances r and is also protected against asymmetry of distribution of T(r).
- 'st': The studentised envelope based on the studentised MAD measure (Myllymäki et al., 2017, 2015), which takes into account the unequal variances of the test function T(r) for different distances r.
- 'unscaled': The unscaled envelope (Ripley, 1981), which leads to envelopes with constant width. It corresponds to the classical maximum deviation test without scaling. This test suffers from unequal variance of T(r) over the distances r and from the asymmetry of distribution of T(r). We recommend to use the other alternatives instead. This unscaled global envelope is provided for reference.

The values of the chosen measure M are determined for each curve in the curve_set, and based on the chosen measure, the central region, i.e. the global envelope, is constructed for the given curves.

If a list of (suitable) objects are provided in the argument curve_sets, then by default (nstep = 2) the two-step combining procedure is used to construct the combined global envelope as described in Myllymäki and Mrkvička (2020, Section 2.2.). If nstep = 1 and the lengths of the multivariate vectors in each component of the list are equal, then the one-step combining procedure is used where the functions are concatenated together into a one long vector (see again Myllymäki and Mrkvička, 2020, Section 2.2.).

Value

Either an object of class `global_envelope` and or an `combined_global_envelope` object. The former class is obtained when a set of curves is provided, while the latter in the case that `curve_sets` is a list of objects. The print and plot function are defined for the returned objects (see examples).

The `global_envelope` object is essentially a data frame containing columns

- r = the vector of values of the argument r at which the test was made
- lo = the lower envelope based on the simulated functions; in case of a vector of coverage values, several 'lo' exist with names paste0("lo.", 100*coverage)
- hi = the upper envelope based on the simulated functions; in case of a vector of coverage values, several 'lo' exist with names paste0("hi.", 100*coverage)
- central = If the `curve_set` (or `envelope` object) contains a theoretical curve, then this function is used as the central curve and returned in this component. Otherwise, the central curve is the mean or median (according to the argument `central`) of the test functions T_i(r), i=2, ..., s+1. Used for visualization only.

and potentially additionally

- obs = the data function, if there is only one data function in the given `curve_sets`. Otherwise not existing.

(Most often `central_region` is directly applied to functional data where all curves are observed.) Additionally, the returned object has some attributes, where

- M = A vector of the values of the chosen measure for all the function. If there is only one observed function, then M[1] gives the value of the measure for this.
- M_alpha = The critical value of M corresponding to the 100(1-alpha)% global envelope (see Myllymäki and Mrkvička, 2020, Definition 1.1. IGI).

Further the object has some attributes for printing and plotting purposes, where `alternative`, `type`, `ties`, `alpha` correspond to those in the function call and `method` gives a name for the method. Attributes of an object `res` can be obtained using the function `attr`, e.g. `attr(res, "M")` for the values of the ordering measure.

If the given set of curves had the class `envelope` of **spatstat**, then the returned `global_envelope` object has also the class `fv` of spatstat, whereby one can utilize also the plotting functions of **spatstat**, see example in `plot.global_envelope`. However, the `envelope` objects are most often used with `global_envelope_test` and not with `central_region`. For an `fv` object, also some further attributes exists as required by `fv` of **spatstat**.

The `combined_global_envelope` is a list of `global_envelope` objects, where the components correspond to the components of `curve_sets`. The `combined_global_envelope` object constructed with `nstep = 2` contains, in addition to some conventional ones (`method`, `alternative`, `type`, `alpha`, `M`, `M_alpha`, see above), the second level envelope information as the attributes

- level2_ge = The second level envelope on which the envelope construction is based

- level2_curve_set = The second level `curve_set` from which `level2_ge` is constructed

In the case that the given curve sets are two-dimensional, i.e., their arguments values are two-dimensional, then the returned objects have in addition to the class `global_envelope` or `combined_global_envelope`, the class `global_envelope2d` or `combined_global_envelope2d`, respectively. This class is assigned for plotting purposes: For the 2d envelopes, also the default plots are 2d. Otherwise the 1d and 2d objects are similar.

References

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2020) A one-way ANOVA test for functional data with graphical interpretation. Kybernetika 56(3), 432-458. doi: 10.14736/kyb-2020-3-0432

Mrkvička, T., Myllymäki, M., Kuronen, M. and Narisetty, N. N. (2022) New methods for multiple testing in permutation inference for the general linear model. Statistics in Medicine 41(2), 276-297. doi: 10.1002/sim.9236

Myllymäki, M., Grabarnik, P., Seijo, H. and Stoyan. D. (2015). Deviation test construction and power comparison for marked spatial point patterns. Spatial Statistics 11, 19-34. doi: 10.1016/j.spasta.2014.11.004

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017). Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 79, 381-404. doi: 10.1111/rssb.12172

Myllymäki, M. and Mrkvička, T. (2020). GET: Global envelopes in R. arXiv:1911.06583 [stat.ME]

Ripley, B.D. (1981). Spatial statistics. Wiley, New Jersey.

See Also

`forder`, `global_envelope_test`

Examples

```
## A central region of a set of functions
#----------------------------------------
if(requireNamespace("fda", quietly=TRUE)) {
  curve_set <- create_curve_set(list(r=as.numeric(row.names(fda::growth$hgtf)),
                                     obs=fda::growth$hgtf))
  plot(curve_set) + ggplot2::ylab("height")
  cr <- central_region(curve_set, coverage=0.50, type="erl")
  plot(cr)
}

## Confidence bands for linear or polynomial regression
#------------------------------------------------------
# Simulate regression data according to the cubic model
# f(x) = 0.8x - 1.8x^2 + 1.05x^3 for x in [0,1]
par <- c(0,0.8,-1.8,1.05) # Parameters of the true polynomial model
res <- 100 # Resolution
x <- seq(0, 1, by=1/res); x2=x^2; x3=x^3;
f <- par[1] + par[2]*x + par[3]*x^2 + par[4]*x^3 # The true function
d <- f + rnorm(length(x), 0, 0.04) # Data
# Plot the true function and data
plot(f, type="l", ylim=range(d))
```

```
points(d)

# Estimate polynomial regression model
reg <- lm(d ~ x + x2 + x3)
ftheta <- reg$fitted.values
resid0 <- reg$residuals
s0 <- sd(resid0)

# Bootstrap regression
B <- 2000 # Number of bootstrap samples

ftheta1 <- array(0, c(B,length(x)))
s1 <- array(0,B)
for(i in 1:B) {
  u <- sample(resid0, size=length(resid0), replace=TRUE)
  reg1 <- lm((ftheta+u) ~ x + x2 + x3)
  ftheta1[i,] <- reg1$fitted.values
  s1[i] <- sd(reg1$residuals)
}

# Centering and scaling
meanftheta <- apply(ftheta1, 2, mean)
m <- array(0, c(B,length(x)))
for(i in 1:B) { m[i,] <- (ftheta1[i,]-meanftheta)/s1[i] }

# Central region computation
boot.cset <- create_curve_set(list(r=1:length(x), obs=ftheta+s0*t(m)))
cr <- central_region(boot.cset, coverage=0.95, type="erl")

# Plotting the result
plot(cr) + ggplot2::labs(x=expression(italic(x)), y=expression(italic(f(x)))) +
  ggplot2::geom_point(data=data.frame(id=1:length(d), points=d),
                      ggplot2::aes(x=id, y=points)) + # data points
  ggplot2::geom_line(data=data.frame(id=1:length(d), points=f),
                     ggplot2::aes(x=id, y=points)) # true function
```

---

cgec *Centred government expenditure centralization ratios*

Description

Centred government expenditure centralization (GEC) ratios

Usage

```
data("cgec")
```

Format

A list of two components. The first one is the `curve_set` object containing the observed values of centred GEC observed in year 1995-2016 for the above countries. The second component `group` gives the grouping.

Details

The data includes the government expenditure centralization (GEC) ratio in percent that has been centred with respect to country average in order to remove the differences in absolute values of GEC. The GEC ratio is the ratio of central government expenditure to the total general government expenditure. Data were collected from the Eurostat (2018) database. Only those European countries were included, where the data were available from 1995 to 2016 without interruption. Finally, 29 countries were classified into three groups in the following way:

- Group 1: Countries joining EC between 1958 and 1986 (Belgium, Denmark, France, Germany (until 1990 former territory of the FRG), Greece, Ireland, Italy, Luxembourg, Netherlands, Portugal, Spain, United Kingdom. These countries have long history of European integration, representing the core of integration process.
- Group 2: Countries joining the EU in 1995 (Austria, Sweden, Finland) and 2004 (Malta, Cyprus), except CEEC (separate group), plus highly economically integrated non-EU countries, EFTA members (Norway, Switzerland). Countries in this group have been, or in some case even still are standing apart from the integration mainstream. Their level of economic integration is however very high.
- Group 3: Central and Eastern European Countries (CEEC), having similar features in political end economic history. The process of economic and political integration have been initiated by political changes in 1990s. CEEC joined the EU in 2004 and 2007 (Bulgaria, Czech Republic, Estonia, Hungary, Latvia, Lithuania, Poland, Romania, Slovakia, Slovenia, data for Croatia joining in 2013 are incomplete, therefore not included).

This grouping is used in examples.

References

Eurostat (2018). "Government revenue, expenditure and main aggregates (gov10amain)”. Retrieved from https://ec.europa.eu/eurostat/data/database(26/10/2018).

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2020) A one-way ANOVA test for functional data with graphical interpretation. Kybernetika 56 (3), 432-458. doi: 10.14736/kyb-2020-3-0432

See Also

graph.fanova

Examples

```
data("cgec")
# Plot data in groups
for(i in 1:3)
  assign(paste0("p", i), plot(subset(cgec$cgec, cgec$group == i)) +
    ggplot2::labs(title=paste("Group ", i, sep=""), y="Centred GEC"))
```

```
p3
if(require("patchwork", quietly=TRUE))
  p1 + p2 + p3
```

---

combined_scaled_MAD_envelope_test

Combined global scaled maximum absolute difference (MAD) envelope tests

---

Description

Given a list of 'curve_set' objects (see `create_curve_set`), a combined global scaled (directional quantile or studentized) MAD envelope test is performed with the test functions saved in the curve set objects. Details of this combined test can be found in Mrkvicka et al. (2017). The implementation of this test is provided here for historical reasons: we recommend now instead the use of `global_envelope_test` also for combined tests; these combined tests are there implemented as described in Myllymäki and Mrkvička (2020).

Usage

```
combined_scaled_MAD_envelope_test(
  curve_sets,
  type = c("qdir", "st"),
  alpha = 0.05,
  probs = c(0.025, 0.975),
  central = "mean",
  ...
)
```

Arguments

`curve_sets`	A `curve_set` (see `create_curve_set`) or an `envelope` object of **spatstat** containing a data function and simulated functions. If an envelope object is given, it must contain the summary functions from the simulated patterns which can be achieved by setting `savefuns = TRUE` when calling the `envelope` function. Alternatively, a list of `curve_set` or `envelope` objects can be given.
`type`	Either "qdir" for the direction quantile envelope test or "st" for the studentized envelope test.
`alpha`	The significance level. The 100(1-alpha)% global envelope will be calculated. If a vector of values is provided, the global envelopes are calculated for each value.
`probs`	A two-element vector containing the lower and upper quantiles for the measure 'q' or 'qdir', in that order and on the interval [0, 1]. The default values are 0.025 and 0.975, suggested by Myllymäki et al. (2015, 2017).

`central` Either "mean" or "median". If the curve sets do not contain the component `theo` for the theoretical central function, then the central function (used for plotting only) is calculated either as the mean or median of functions provided in the curve sets. For 'qdir', 'st' and 'unscaled' only the mean is allowed as an option, due to their definition.

`...` Additional parameters to be passed to `central_region`.

References

Mrkvička, T., Myllymäki, M. and Hahn, U. (2017) Multiple Monte Carlo testing, with applications in spatial point processes. Statistics & Computing 27(5): 1239–1255. DOI: 10.1007/s11222-016-9683-9

Myllymäki, M. and Mrkvička, T. (2020). GET: Global envelopes in R. arXiv:1911.06583 [stat.ME]

Examples

```
if(require("spatstat.explore", quietly=TRUE)) {
  # As an example test CSR of the saplings point pattern from spatstat by means of
  # L, F, G and J functions.
  data("saplings")
  X <- as.ppp(saplings, W=square(75))

  nsim <- 499 # Number of simulations for the tests

  # Specify distances for different test functions
  n <- 500 # the number of r-values
  rmin <- 0; rmax <- 20; rstep <- (rmax-rmin)/n
  rminJ <- 0; rmaxJ <- 8; rstepJ <- (rmaxJ-rminJ)/n
  r <- seq(0, rmax, by=rstep)    # r-distances for Lest
  rJ <- seq(0, rmaxJ, by=rstepJ) # r-distances for Fest, Gest, Jest


  # Perform simulations of CSR and calculate the L-functions
  env_L <- envelope(X, nsim=nsim,
   simulate=expression(runifpoint(ex=X)),
   fun="Lest", correction="translate",
   transform=expression(.-r), # Take the L(r)-r function instead of L(r)
   r=r,                       # Specify the distance vector
   savefuns=TRUE,             # Save the estimated functions
   savepatterns=TRUE)         # Save the simulated patterns
  # Take the simulations from the returned object
  simulations <- attr(env_L, "simpatterns")
  # Then calculate the other test functions F, G, J for each simulated pattern
  env_F <- envelope(X, nsim=nsim,
                   simulate=simulations,
                   fun="Fest", correction="Kaplan", r=rJ,
                   savefuns=TRUE)
  env_G <- envelope(X, nsim=nsim,
                   simulate=simulations,
                   fun="Gest", correction="km", r=rJ,
                  savefuns=TRUE)
```

```
  env_J <- envelope(X, nsim=nsim,
                    simulate=simulations,
                    fun="Jest", correction="none", r=rJ,
                    savefuns=TRUE)

  # Crop the curves to the desired r-interval I
  curve_set_L <- crop_curves(env_L, r_min=rmin, r_max=rmax)
  curve_set_F <- crop_curves(env_F, r_min=rminJ, r_max=rmaxJ)
  curve_set_G <- crop_curves(env_G, r_min=rminJ, r_max=rmaxJ)
  curve_set_J <- crop_curves(env_J, r_min=rminJ, r_max=rmaxJ)

  # The combined directional quantile envelope test
  res <- combined_scaled_MAD_envelope_test(
             curve_sets=list(L=curve_set_L, F=curve_set_F,
                             G=curve_set_G, J=curve_set_J),
             type="qdir")
  plot(res)
}
```

`create_curve_set` *Create a curve_set object*

Description

Create a curve_set object out of a list in the right form.

Usage

```
create_curve_set(curve_set, ...)
```

Arguments

`curve_set`	A list containing the element obs, and optionally the elements r, sim_m and theo. See details.
`...`	For expert use only.

Details

The function is used to clump together the functional data in the form that can be handled by the other **GET** functions (`forder`, `central_region`, `global_envelope_test` etc.). The function `create_curve_set` takes care of checking the content of the data, and saves relevant information of the curves for global envelope methods to be used in particular for plotting the results with graphical interpretation.

`obs` must be either

- a vector containing the data function/vector, or
- a matrix containing the s data functions/vectors, in which case it is assumed that each column corresponds to a data function/vector.

If given, r describes the 1- or 2-dimensional argument values where the functions/vectors have been observed (or simulated). It must be either

- a vector,
- a data.frame with columns "x", "y", "width" and "height", where the width and height give the width and height of the pixels placed at x and y, or
- a data.frame with columns "xmin", "xmax", "ymin" and "ymax" giving the corner coordinates of the pixels where the data have been observed.

If obs is a vector, sim_m must be a matrix containing the simulated functions. Each column is assumed to correspond to a function, and the number of rows must match the length of obs. If obs is a matrix, sim_m is ignored.

If obs is a vector, theo can be given and it should then correspond to a theoretical function (e.g., under the null hypothesis). If present, its length must match the length of obs.

Value

An object of class curve_set containing the data. If the argument values are two-dimensional, then the curve_set is additionally a curve_set2d object.

See Also

plot.curve_set, plot.curve_set2d

Examples

```
# 1d
cset <- create_curve_set(list(r = 1:10, obs = matrix(runif(10*5), ncol=5)))
plot(cset)
# 2d
cset <- create_curve_set(list(r = data.frame(x=c(rep(1:3, 3), 4), y=c(rep(1:3, each=3), 1),
                                             width=1, height=1),
                              obs = matrix(runif(10*5), ncol=5)))
plot(cset)
```

---

create_image_set *Create a curve set of images*

Description

Create a curve set consisting of a set of images, given a list containing the values of the 2d functions in the right form. Only 2d functions in a rectangular windows are supported; the values are provided in matrices (arrays). For more general 2d functions see create_curve_set.

Usage

```
create_image_set(image_set, ...)
```

Arguments

- `image_set` A list containing elements `r`, `obs`, `sim_m` and `theo`. `r`, `sim_m` and `theo` are optional, `obs` needs to be provided always. If provided, `r` must be a `list` describing the argument values where the images have been observed (or simulated). It must consist of the following two or four components: a) "x" and "y" giving the equally spaced argument values for the x- and y-coordinates (first and second dimension of the 2d functions) where the data have been observed, b) "x", "y", "width" and "height", where the width and height give the width and height of the pixels placed at x and y, or c) "xmin", "xmax", "ymin" and "ymax" giving the corner coordinates of the pixels where the data have been observed. If not given, r is set to be a list of values from 1 to the number of first/second dimension of 2d functions in `obs`. `obs` must be either a 2d matrix (dimensions matching the lengths of r vectors) or 3d array containing the observed 2d functions (the third dimension matching the number of functions). If `obs` is a 3d array, then `sim_m` is ignored. If `obs` is a 2d array, then `sim_m` must be a 3d array containing the simulated images (2d functions) (the third dimension matching the number of functions). If included, `theo` corresponds to the theoretical function (e.g., under the null hypothesis) and its dimensions must either match the dimensions of 2d functions in `obs` or it must be a constant.
- `...` Do not use. (For internal use only.)

Value

The given list as a `curve_set`.

Examples

```
a <- create_image_set(list(obs=array(runif(4*5*6), c(4,5,6))))
plot(a)
plot(a, idx=1:6)

a <- create_image_set(list(r=list(x=c(10,20,30,40), y=1:5*0.1),
                           obs=array(runif(4*5*6), c(4,5,6))))
plot(a)

a <- create_image_set(list(r=list(xmin=c(1, 2, 4, 7), xmax=c(2, 4, 7, 11),
                                  ymin=c(1,1.1,2,2.1,3), ymax=c(1.1,2,2.1,3,3.1)),
                           obs=array(runif(4*5*6), c(4,5,6))))
plot(a)
plot(a, idx=1:5)
```

`crop_curves` *Crop the curves to a certain interval*

Description

Crop the curves to a certain interval

Usage

```
crop_curves(curve_set, r_min = NULL, r_max = NULL)
```

Arguments

`curve_set`	A curve_set (see `create_curve_set`) or an `envelope` object of **spatstat**. If an envelope object is given, it must contain the summary functions from the simulated patterns which can be achieved by setting savefuns = TRUE when calling the `envelope` function.
`r_min`	The minimum radius to include.
`r_max`	The maximum radius to include.

Details

The curves can be cropped to a certain interval defined by the arguments r_min and r_max. The interval should generally be chosen carefully for classical deviation tests.

Value

A curve_set object containing the cropped summary functions and the cropped radius vector.

`deviation_test` *Deviation test*

Description

Crop the curve set to the interval of distances [r_min, r_max], calculate residuals, scale the residuals and perform a deviation test with a chosen deviation measure. The deviation tests are well known in spatial statistics; in **GET** they are provided for comparative purposes. Some (maximum type) of the deviation test have their corresponding envelope tests available, see Myllymäki et al., 2017 (and 'unscaled', 'st' and 'qdir' in `global_envelope_test`).

Usage

```
deviation_test(
  curve_set,
  r_min = NULL,
  r_max = NULL,
  use_theo = TRUE,
  scaling = "qdir",
  measure = "max",
  savedevs = FALSE
)
```

Arguments

`curve_set`	A residual curve_set object. Can be obtained by using residual().
`r_min`	The minimum radius to include.
`r_max`	The maximum radius to include.
`use_theo`	Whether to use the theoretical summary function or the mean of the functions in the curve_set.
`scaling`	The name of the scaling to use. Options include 'none', 'q', 'qdir' and 'st'. 'qdir' is default.
`measure`	The deviation measure to use. Default is 'max'. Must be one of the following: 'max', 'int' or 'int2'.
`savedevs`	Logical. Should the global rank values k_i, i=1,...,nsim+1 be returned? Default: FALSE.

Details

The deviation test is based on a test function $T(r)$ and it works as follows:

1) The test function estimated for the data, $T_1(r)$, and for nsim simulations from the null model, $T_2(r), ..., T_{nsim+1}(r)$, must be saved in 'curve_set' and given to the deviation_test function.

2) The deviation_test function then

- Crops the functions to the chosen range of distances $[r_{\min}, r_{\max}]$.
- If the curve_set does not consist of residuals (see residual), then the residuals $d_i(r) = T_i(r) - T_0(r)$ are calculated, where $T_0(r)$ is the expectation of $T(r)$ under the null hypothesis. If use_theo = TRUE, the theoretical value given in the curve_set$theo is used for as $T_0(r)$, if it is given. Otherwise, $T_0(r)$ is estimated by the mean of $T_j(r)$, $j = 2, ..., nsim + 1$.
- Scales the residuals. Options are
  - 'none' No scaling. Nothing done.
  - 'q' Quantile scaling.
  - 'qdir' Directional quantile scaling.
  - 'st' Studentised scaling.

  See for details Myllymäki et al. (2013).
- Calculates the global deviation measure u_i, $i = 1, ..., nsim + 1$, see options for 'measure'.
  - 'max' is the maximum deviation measure

    $$u_i = \max_{r \in [r_{\min}, r_{\max}]} |w(r)(T_i(r) - T_0(r))|$$

  - 'int2' is the integral deviation measure

    $$u_i = \int_{r_{\min}}^{r_{\max}} (w(r)(T_i(r) - T_0(r)))^2 dr$$

  - 'int' is the 'absolute' integral deviation measure

    $$u_i = \int_{r_{\min}}^{r_{\max}} |w(r)(T_i(r) - T_0(r))| dr$$

- Calculates the p-value.

Currently, there is no special way to take care of the same values of $T_i(r)$ occuring possibly for small distances. Thus, it is preferable to exclude from the test the very small distances r for which ties occur.

Value

If 'savedevs=FALSE' (default), the p-value is returned. If 'savedevs=TRUE', then a list containing the p-value and calculated deviation measures u_i, $i = 1, ..., nsim + 1$ (where u_1 corresponds to the data pattern) is returned.

References

Myllymäki, M., Grabarnik, P., Seijo, H. and Stoyan. D. (2015). Deviation test construction and power comparison for marked spatial point patterns. Spatial Statistics 11: 19-34. doi: 10.1016/j.spasta.2014.11.004

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017). Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 79: 381–404. doi: 10.1111/rssb.12172

Examples

```
## Testing complete spatial randomness (CSR)
#-------------------------------------------
if(require("spatstat.explore", quietly=TRUE)) {
  pp <- unmark(spruces)
  nsim <- 999

  # Generate nsim simulations under CSR, calculate L-function for the data and simulations
  env <- envelope(pp, fun="Lest", nsim=nsim, savefuns=TRUE, correction="translate")
  # The deviation test using the integral deviation measure
  res <- deviation_test(env, measure='int')
  res
  # or
  res <- deviation_test(env, r_min=0, r_max=7, measure='int2')
}
```

`fallen_trees` *Fallen trees*

Description

Fallen trees

Usage

```
data("fallen_trees")
```

Format

A `list` of two data frames, where `trees` contains the locations (x and y coordinates) and heights (=marks) of 232 trees in a window with polygonal boundary, and `window` species the polygonal window (see examples).

Details

The dataset comprised the locations and heights of 232 trees, which fell during two large wind gusts (1967 and 1990) in the west of France (Pontailler et al., 1997). The study area was a biological reserve, which had been preserved for at least four centuries, with little human influence for a long period (Guinier, 1950). Thus, the forest stand followed almost natural dynamics. It was an uneven-aged beech stand with a few old oaks.

The data was analysed in Myllymäki et al. (2017, Supplementary material).

References

Guinier, P. (1950) Foresterie et protection de la nature. l'exemple de fontainebleau. Rev Forestière Fr., II, 703-717.

Pontailler, J.-Y., Faille, A. and Lemée, G. (1997) Storms drive successional dynamics in natural forests: a case study in fontainebleau forest (france). Forest Ecol. Manag., 98, 1-15.

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017). Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 79: 381–404. doi: 10.1111/rssb.12172

Examples

```
data("fallen_trees")
if(require("spatstat.geom", quietly=TRUE)) {
  fallen_trees <- as.ppp(fallen_trees$trees, W = owin(poly=fallen_trees$window))
  plot(fallen_trees)
}
```

---

fBoxplot *Functional boxplot*

Description

Functional boxplot based on central region computed by a specified measure. The options of the measures can be found in `central_region`.

Usage

```
fBoxplot(curve_sets, factor = 1.5, coverage = 0.5, ...)
```

Arguments

curve_sets	A curve_set object or a list of curve_set objects.
factor	The constant factor to inflate the central region to produce a functional boxplot and determine fences for outliers. Default is 1.5 as in a classical boxplot.
coverage	A number between 0 and 1. The 100*coverage% central region will be calculated. A vector of values can also be provided, leading to the corresponding number of central regions.
...	Additional parameters to be passed to central_region, which is responsible for calculating the central region (global envelope) on which the functional boxplot is based.

Examples

```
if(requireNamespace("fda", quietly=TRUE)) {
  years <- paste(1:18)
  curves <- fda::growth[['hgtf']][years,]
  # Heights
  cset1 <- create_curve_set(list(r = as.numeric(years),
                               obs = curves))
  bp <- fBoxplot(cset1, coverage=0.50, type="area", factor=1)
  plot(bp)

  # Considering simultaneously heights and height differences
  cset2 <- create_curve_set(list(r = as.numeric(years[-1]),
             obs = curves[-1,] - curves[-nrow(curves),]))
  csets <- list(Height=cset1, Change=cset2)
  res <- fBoxplot(csets, type='area', factor=1.5)
  plot(res) + ggplot2::labs(x="Age (years)", y="")
}
```

---

fclustering *Functional clustering*

Description

Functional clustering based on a specified measure. The options of the measures can be found in central_region.

Usage

```
fclustering(curve_sets, k, type = c("area", "st", "erl", "cont"), ...)
```

Arguments

curve_sets	A curve_set object or a list of curve_set objects to which the functional clustering is to be applied. If list of curve_set objects is provided, then the joined functional clustering is applied, which provides an equal weight combination of curve_set objects, if the curve_set objects contain the same numbers of elements (same lengths of vector r).

k	The number of clusters.
type	The measure which is used to compute the dissimilarity matrix. The preferred options are "area" and "st", but "erl" and "cont" can be also used with caution.
...	Additional parameters to be passed to central_region, which is responsible for calculating the central region (global envelope) on which the functional clustering is based.

Details

Functional clustering joins the list of curve_set objects in one curve_set with long functions and applies on the differences of all functions the specified measure. This provides a dissimilarity matrix which is used in partitioning around medoids procedure. The resulting clusters can then be shown by plotting the function respectively for each curve_set. Thus for each curve_set, the panel with all the medoids is shown followed by all clusters represented by central region, medoid and all curves belonging to it, when the result object is plotted.

If there are less than three curves in some of the groups, then the central region is not plotted. This leads to a warning message from ggplot2.

Value

An object having the class fclust, containing

- curve_sets = The set(s) of functions determined for clustering
- k = Number of clusters
- type = Type of clustering method
- triangineq = The proportion of combinations of functions which satisfies the triangular inequality. The triangular inequality must hold to ensure the chosen measure forms a metric. In some weird cases it does not hold for 'area' measure, therefore this check is provided to ensure the data forms metric with the 'area' measure. The triangineq must be 1 to ensure the inequality holds for all functions.
- dis = The joined dissimilarity matrix
- pam = Results of the partitioning around medoids (pam) method applied on the joined functions with the dissimilarity matrix (dis). See pam.

References

Dai, W., Athanasiadis, S., Mrkvička, T. (2021) A new functional clustering method with combined dissimilarity sources and graphical interpretation. Intech open, London, UK. DOI: 10.5772/intechopen.100124

See Also

central_region, plot.fclust

Examples

```
# Read raw data from population growth rdata
# with countries over million inhabitants
data("popgrowthmillion")

# Create centred data
m <- apply(popgrowthmillion, 2, mean) # Country-wise means
cpopgrowthmillion <- popgrowthmillion
for(i in 1:dim(popgrowthmillion)[1]) {
  cpopgrowthmillion[i,] <- popgrowthmillion[i,] - m
}

# Create scaled data
t2 <- function(v) { sqrt(sum(v^2)) }
s <- apply(cpopgrowthmillion, 2, t2)
spopgrowthmillion <- popgrowthmillion
for(i in 1:dim(popgrowthmillion)[1]) {
  spopgrowthmillion[i,] <- cpopgrowthmillion[i,]/s
}

# Create curve sets
r <- 1951:2015

cset1 <- create_curve_set(list(r = r, obs = popgrowthmillion))
cset2 <- create_curve_set(list(r = r, obs = spopgrowthmillion))
csets <- list(Raw = cset1, Shape = cset2)

# Functional clustering with respect to joined "st" difference measure
# and "joined" central regions of each group
res <- fclustering(csets, k=3, type="area")
p <- plot(res, plotstyle = "marginal", coverage = 0.5)
p[[1]] # Central functions
p[[2]] # Groups: central functions and regions
# To collect the two figures into one use, e.g., patchwork:
if(require("patchwork", quietly=TRUE)) {
  p[[1]] + p[[2]] + plot_layout(widths = c(1, res$k))
}
# Silhouette plot of pam
plot(res$pam)
```

---

fdr_envelope *The FDR envelope*

Description

Calculate the FDR envelope based on the ATSE or IATSE algorithm of Mrkvička and Myllymäki (2022).

Usage

```
fdr_envelope(
  curve_sets,
  alpha = 0.05,
  alternative = c("two.sided", "less", "greater"),
  algorithm = c("IATSE", "ATSE"),
  lower = NULL,
  upper = NULL
)
```

Arguments

`curve_sets`	A `curve_set` (see `create_curve_set`) or an `envelope` object of **spatstat** containing the observed function and the functions from which the envelope is to be constructed. Alternatively, a list of appropriate objects can be given.
`alpha`	The significance level. The 100(1-alpha)% global envelope will be calculated. If a vector of values is provided, the global envelopes are calculated for each value.
`alternative`	A character string specifying the alternative hypothesis. Must be one of the following: "two.sided" (default), "less" or "greater". The last two options only available for types `'rank'`, `'erl'`, `'cont'` and `'area'`.
`algorithm`	Either "IATSE" or "ATSE" standing for the iteratively adaptive two-stage envelope and the adaptive two-stage envelope, respectively, see Mrkvička and Myllymäki (2022).
`lower`	A single number (or a vector of suitable length) giving a lower bound for the functions. Used only for the extension, see Mrkvička and Myllymäki (2022, p. 6).
`upper`	A single number (or a vector of suitable length) giving an upper bound for the functions.

References

Mrkvička and Myllymäki (2022). False discovery rate envelopes. arXiv:2008.10108 [stat.ME]

Examples

```
# A GLM example
data(rimov)
nsim <- 1000 # Number of simulations

res <- graph.flm(nsim=nsim,
                 formula.full = Y~Year,
                 formula.reduced = Y~1,
                 typeone = "fdr",
                 curve_sets = list(Y=rimov),
                 factors = data.frame(Year = 1979:2014))
plot(res)
```

forder *Functional ordering*

Description

Calculates different measures for ordering the functions (or vectors) from the most extreme to least extreme one

Usage

```
forder(
  curve_sets,
  measure = "erl",
  scaling = "qdir",
  alternative = c("two.sided", "less", "greater"),
  use_theo = TRUE,
  probs = c(0.025, 0.975),
  quantile.type = 7
)
```

Arguments

curve_sets	A curve_set object or a list of curve_set objects.
measure	The measure to use to order the functions from the most extreme to the least extreme one. Must be one of the following: 'rank', 'erl', 'cont', 'area', 'max', 'int', 'int2'. Default is 'erl'.
scaling	The name of the scaling to use if measure is 'max', 'int' or 'int2'. Options include 'none', 'q', 'qdir' and 'st', where 'qdir' is the default.
alternative	A character string specifying the alternative hypothesis. Must be one of the following: "two.sided" (default), "less" or "greater". The last two options only available for types 'rank', 'erl', 'cont' and 'area'.
use_theo	Logical. When calculating the measures 'max', 'int', 'int2', should the theoretical function from curve_set be used (if 'theo' provided), see deviation_test.
probs	A two-element vector containing the lower and upper quantiles for the measure 'q' or 'qdir', in that order and on the interval [0, 1]. The default values are 0.025 and 0.975, suggested by Myllymäki et al. (2015, 2017).
quantile.type	As type argument of quantile, how to calculate quantiles for 'q' or 'qdir'.

Details

Given a curve_set (see create_curve_set for how to create such an object) or an envelope object of **spatstat**, which contains curves $T_1(r), \dots, T_s(r)$, the functions are ordered from the most extreme one to the least extreme one by one of the following measures (specified by the argument measure). Note that 'erl', 'cont' and 'area' were proposed as a refinement to the extreme ranks 'rank', because the extreme ranks can contain many ties. All of these completely non-parametric measures are smallest for the most extreme functions and largest for the least extreme ones, whereas the deviation measures (`'max'`, `'int'` and `'int2'`) obtain largest values for the most extreme functions.

- `'rank'`: extreme rank (Myllymäki et al., 2017). The extreme rank R_i is defined as the minimum of pointwise ranks of the curve $T_i(r)$, where the pointwise rank is the rank of the value of the curve for a specific r-value among the corresponding values of the s other curves such that the lowest ranks correspond to the most extreme values of the curves. How the pointwise ranks are determined exactly depends on the whether a one-sided (`alternative` is "less" or "greater") or the two-sided test (`alternative="two.sided"`) is chosen.
- `'erl'`: extreme rank length (Myllymäki et al., 2017). Considering the vector of pointwise ordered ranks $\mathbf{R}_i$ of the ith curve, the extreme rank length measure R_i^{erl} is equal to

$$R_i^{erl} = \frac{1}{s} \sum_{j=1}^{s} \mathbf{1}(\mathbf{R}_j" < "\mathbf{R}_i)$$

  where $\mathbf{R}_j" < "\mathbf{R}_i$ if and only if there exists $n \leq d$ such that for the first k, $k < n$, pointwise ordered ranks of $\mathbf{R}_j$ and $\mathbf{R}_i$ are equal and the n'th rank of $\mathbf{R}_j$ is smaller than that of $\mathbf{R}_i$. The scaling by

$$s$$

  is applied to normalize the ranks following Mrkvička et al. (2019) and Narisetty and Nair (2016).
- `'cont'`: continuous rank (Hahn, 2015; Mrkvička et al., 2019) based on minimum of continuous pointwise ranks
- `'area'`: area rank (Mrkvička et al., 2019) based on area between continuous pointwise ranks and minimum pointwise ranks for those argument (r) values for which pointwise ranks achieve the minimum (it is a combination of erl and cont)
- `'max'` and `'int'` and `'int2'`: Further options for the `measure` argument that can be used together with `scaling`. See the help in `deviation_test` for these options of `measure` and `scaling`. These measures are largest for the most extreme functions and smallest for the least extreme ones. The arguments `use_theo` and `probs` are relevant for these measures only (otherwise ignored).

For details see Myllymäki and Mrkvička et al. (2020, Section 2)

Value

A vector containing one of the above mentioned measures k for each of the functions in the curve set. If the component `obs` in the curve set is a vector, then its measure will be the first component (named 'obs') in the returned vector.

References

Hahn U (2015). "A note on simultaneous Monte Carlo tests." Technical report, Centre for Stochastic Geometry and advanced Bioimaging, Aarhus University.

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2020) A one-way ANOVA test for functional data with graphical interpretation. Kybernetika 56(3), 432-458. doi: 10.14736/kyb-2020-3-0432

Mrkvička, T., Myllymäki, M., Kuronen, M. and Narisetty, N. N. (2022) New methods for multiple testing in permutation inference for the general linear model. Statistics in Medicine 41(2), 276-297. doi: 10.1002/sim.9236

Myllymäki, M., Grabarnik, P., Seijo, H. and Stoyan. D. (2015). Deviation test construction and power comparison for marked spatial point patterns. Spatial Statistics 11, 19-34. doi: 10.1016/j.spasta.2014.11.004

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017). Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 79, 381-404. doi: 10.1111/rssb.12172

Narisetty, N. N. and Nair, V. J. (2016) Extremal depth for functional data and applications. Journal of the American Statistical Association 111, 1705-1714.

See Also

partial_forder

Examples

```
if(requireNamespace("fda", quietly = TRUE)) {
  # Consider ordering of the girls in the Berkeley Growth Study data
  # available from the R package fda, see ?growth, according to their
  # annual heights or/and changes within years.
  # First create sets of curves (vectors), for raw heights and
  # for the differences within the years
  years <- paste(1:18)
  curves <- fda::growth[['hgtf']][years,]
  cset1 <- create_curve_set(list(r = as.numeric(years),
                                 obs = curves))
  cset2 <- create_curve_set(list(r = as.numeric(years[-1]),
                                 obs = curves[-1,] - curves[-nrow(curves),]))

 # Order the girls from most extreme one to the least extreme one, below using the 'area' measure
  # a) according to their heights
  forder(cset1, measure = 'area')
  # Print the 10 most extreme girl indices
  order(forder(cset1, measure = 'area'))[1:10]
  # b) according to the changes (print indices)
  order(forder(cset2, measure = 'area'))[1:10]
  # c) simultaneously with respect to heights and changes (print indices)
  csets <- list(Height = cset1, Change = cset2)
  order(forder(csets, measure = 'area'))[1:10]
}
```

---

frank.fanova *Rank envelope F-test*

Description

A one-way functional ANOVA based on the rank envelope applied to F values

Usage

```
frank.fanova(
  nsim,
  curve_set,
  groups,
  typeone = c("fwer", "fdr"),
  variances = "equal",
  test.equality = c("mean", "var", "cov"),
  cov.lag = 1,
  savefuns = FALSE,
  ...
)
```

Arguments

nsim	The number of random permutations.
curve_set	The original data (an array of functions) provided as a curve_set object (see create_curve_set) or a fdata object (see fdata). The curve set should include the argument values for the functions in the component r, and the observed functions in the component obs.
groups	The original groups (a factor vector representing the assignment to groups).
typeone	Character string indicating which type I error rate to control, either the familywise error rate ('fwer') or false discovery rate ('fdr'). Further arguments to the FWER or FDR envelope can be passed in argument GET.args. If 'fwer', the type of the envelope can be chosen by specifying the argument type in GET.args.
variances	Either "equal" or "unequal". If "equal", then the traditional F-values are used. If "unequal", then the corrected F-values are used. The current implementation uses lm to get the corrected F-values.
test.equality	A character with possible values mean (default), var and cov. If mean, the functional ANOVA is performed to compare the means in the groups. If var, then the equality of variances of the curves in the groups is tested by performing the graphical functional ANOVA test on the functions $$Z_{ij}(r) = T_{ij}(r) - \bar{T}_j(r).$$ If cov, then the equality of lag cov.lag covariance is tested by performing the fANOVA with $$W_{ij}(r) = \sqrt{\|V_{ij}(r)\| \cdot sign(V_{ij}(r))},$$ where $$V_{ij}(r) = (T_{ij}(r) - \bar{T}_j(r))((T_{ij}(r+s) - \bar{T}_j(r+s))).$$ See Mrkvicka et al. (2020) for more details.
cov.lag	The lag of the covariance for testing the equality of covariances, see test.equality.
savefuns	Logical. If TRUE, then the functions from permutations are saved to the attribute simfuns.
...	Additional parameters to be passed to global_envelope_test (if typeone = "fwer") or fdr_envelope (if typeone = "fdr").

Details

The test assumes that there are J groups which contain $n_1, \ldots, n_J$ functions $T_{ij}, i = \ldots, J, j = 1, \ldots, n_j$. The functions should be given in the argument x, and the groups in the argument groups. The test assumes that there exists non random functions $\mu(r)$ and $\mu_i(r)$ such that

$$T_{ij}(r) = \mu(r) + \mu_i(r) + e_{ij}(r), i = 1, \ldots, J, j = 1, \ldots, n_j$$

where $e_{ij}(r)$ are independent and normally distributed. The test vector is

$$\mathbf{T} = (F(r_1), F(r_2), \ldots, F(r_K)),$$

where $F(r_i)$ stands for the F-statistic. The simulations are performed by permuting the test functions. Further details can be found in Mrkvička et al. (2020).

The argument `variances="equal"` means that equal variances across groups are assumed. The correction for unequal variances can be done by using the corrected F-statistic (option `variances="unequal"`).

Unfortunately this test is not able to detect which groups are different from each other.

References

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2020) A one-way ANOVA test for functional data with graphical interpretation. Kybernetika 56 (3), 432-458. doi: 10.14736/kyb-2020-3-0432

See Also

graph.fanova

Examples

```
data("rimov")
groups <- factor(c(rep(1, times=12), rep(2, times=12), rep(3, times=12)))
res <- frank.fanova(nsim = 2499, curve_set = rimov, groups = groups)

plot(res)

data("imageset3")
res2 <- frank.fanova(nsim = 19, # Increase nsim for serious analysis!
                     curve_set = imageset3$image_set,
                     groups = imageset3$Group)
plot(res2)
plot(res2, fixedscales=FALSE)
```

`frank.flm` *F rank functional GLM*

Description

Multiple testing in permutation inference for the general linear model (GLM)

Usage

```
frank.flm(
  nsim,
  formula.full,
  formula.reduced,
  typeone = c("fwer", "fdr"),
  curve_sets,
  factors = NULL,
  savefuns = TRUE,
  lm.args = NULL,
  GET.args = NULL,
  mc.cores = 1,
  mc.args = NULL,
  cl = NULL,
  method = c("best", "simple", "mlm", "complex", "lm")
)
```

Arguments

`nsim` The number of random permutations.

`formula.full` The formula specifying the general linear model, see `formula` in `lm`.

`formula.reduced` The formula of the reduced model with nuisance factors only. This model should be nested within the full model.

`typeone` Character string indicating which type I error rate to control, either the familywise error rate ('fwer') or false discovery rate ('fdr'). Further arguments to the FWER or FDR envelope can be passed in argument `GET.args`. If 'fwer', the type of the envelope can be chosen by specifying the argument `type` in `GET.args`.

`curve_sets` A named list of sets of curves giving the dependent variable (Y), and possibly additionally factors whose values vary across the argument values of the functions. The dimensions of the elements should match with each other. Note that factors that are fixed across the functions can be given in the argument `factors`. Also `fdata` objects allowed.

`factors` A data frame of factors. An alternative way to specify factors when they are constant for all argument values of the functions. The number of rows of the data frame should be equal to the number of curves. Each column should specify the values of a factor.

`savefuns` Logical or "return". If TRUE, then the functions from permutations are saved to the attribute simfuns. If "return", then the function returns the permutations in a curve_set, instead of the result of the envelope test on those; this can be used by `partial_forder`.

`lm.args` A named list of additional arguments to be passed to `lm`. See details.

`GET.args` A named list of additional arguments to be passed to `global_envelope_test`.

`mc.cores` The number of cores to use, i.e. at most how many child processes will be run simultaneously. Must be at least one, and parallelization requires at least two cores. On a Windows computer mc.cores must be 1 (no parallelization). For details, see `mclapply`, for which the argument is passed. Parallelization can be used in generating simulations and in calculating the second stage tests.

`mc.args` A named list of additional arguments to be passed to `mclapply`. Only relevant if `mc.cores` is more than 1.

`cl` Allows parallelization through the use of `parLapply` (works also in Windows), see the argument `cl` there, and examples.

`method` For advanced use.

Details

The function `frank.flm` performs a nonparametric test of significance of a covariate in the functional GLM. Similarly as in the graphical functional GLM (`graph.flm`), the Freedman-Lane algorithm (Freedman and Lane, 1983) is applied to permute the functions (to obtain the simulations under the null hypothesis of "no effects"); consequently, the test achieves the desired significance level only approximately. If the reduced model contains only a constant, then the algorithm corresponds to simple permutation of raw data. In contrast to the graphical functional GLM, the F rank functional GLM is based on the F-statistics that are calculated at each argument value of the functions. The global envelope test is applied to the observed and simulated F-statistics. The test is able to find if the factor of interest is significant and also which argument values of the functional domain are responsible for the potential rejection.

The specification of the full and reduced formulas is important. The reduced model should be nested within the full model. The full model should include in addition to the reduced model the interesting factors whose effects are under investigation.

There are different versions of the implementation depending on the application.

- If all the covariates are constant across the functions, i.e. they can be provided in the argument `factors`, and there are no extra arguments given by the user in `lm.args`, then a fast implementation is used to directly compute the F-statistics.
- If all the covariates are constant across the functions, but there are some extra arguments, then a linear model is fitted separately by least-squares estimation to the data at each argument value of the functions fitting a multiple linear model by `lm`. The possible extra arguments passed in `lm.args` to `lm` must be of the form that `lm` accepts for fitting a multiple linear model. In the basic case, no extra arguments are needed.
- If some of the covariates vary across the space, i.e. they are provided in the list of curve sets in the argument `curve_sets` together with the dependent functions, but there are no extra arguments given by the user in `lm.args`, there is a rather fast implementation of the F-value calculation (which does not use `lm`).
- If some of the covariates vary across the space and there are user specified extra arguments given in `lm.args`, then the implementation fits a linear model at each argument value of the functions using `lm`, which can be rather slow. The arguments `lm.args` are passed to `lm` for fitting each linear model.

By default the fastest applicable method is used. This can be changed by setting `method` argument. The cases above correspond to "simple", "mlm", "complex" and "lm". Changing the default can be useful for checking the validity of the implementation.

Value

A `global_envelope` object, which can be printed and plotted directly.

References

Mrkvička, T., Myllymäki, M., Kuronen, M. and Narisetty, N. N. (2022) New methods for multiple testing in permutation inference for the general linear model. Statistics in Medicine 41(2), 276-297. doi: 10.1002/sim.9236

Freedman, D., & Lane, D. (1983) A nonstochastic interpretation of reported significance levels. Journal of Business & Economic Statistics 1(4), 292-298. doi:10.2307/1391660

Examples

```
data("GDPtax")
factors.df <- data.frame(Group = GDPtax$Group, Tax = GDPtax$Profittax)

nsim <- 999
res.tax_within_group <- frank.flm(nsim = nsim,
  formula.full = Y~Group+Tax+Group:Tax,
  formula.reduced = Y~Group+Tax,
  curve_sets = list(Y=GDPtax$GDP),
  factors = factors.df)
plot(res.tax_within_group)

# Image set examples
data("abide_9002_23")
iset <- abide_9002_23$curve_set

res.F <- frank.flm(nsim = 19, # Increase nsim for serious analysis!
  formula.full = Y ~ Group + Age + Sex,
  formula.reduced = Y ~ Age + Sex,
  curve_sets = list(Y = iset),
  factors = abide_9002_23[['factors']],
  GET.args = list(type = "area"))
plot(res.F)
```

GDPtax *GDP per capita with country groups and profit tax*

Description

GDP per capita with country groups and profit tax

Usage

```
data("GDPtax")
```

Format

A list of a three components. The first one (`GDP`) is a `curve_set` object with components `r` and `obs` containing the years of observations and the GDP curves, i.e. the observed values of GDP in those years. Each column of `obs` contains the GDP for the years for a particular country (seen as column names). The country grouping is given in the list component `Group` and the profit tax in `Profittax`.

Details

The data includes the GDP per capita (current US$) for years 1980-2017 (World Bank national accounts data, and OECD National Accounts data files). The data have been downloaded from the webpage https://datamarket.com/data/set/15c9/gdp-per-capita-current-us#!ds=15c9!hd1&display=line, distributed under the CC-BY 4.0 (https://datacatalog.worldbank.org/public-licenses#cc-by). From the same webpage the profit tax in 2010 (World Bank, Doing Business Project (http://www.doingbusiness.org/ExploreTopics/I and Total tax rate (were downloaded. Furthermore, different country groups were formed from countries for which the GDP was available for 1980-2017 and profit tax for 2010:

- Group 1 (Major Advanced Economies (G7)): "Canada", "France", "Germany", "Italy", Japan"
- Group 2 (Euro Area excluding G7): "Austria", "Belgium", "Cyprus", "Finland", "Greece", "Ireland", "Luxembourg", "Netherlands", "Portugal", "Spain"
- Group 3 (Other Advanced Economies (Advanced Economies excluding G7 and Euro Area)): "Australia", "Denmark", "Iceland", "Norway", "Sweden", "Switzerland"
- Group 4 (Emerging and Developing Asia): "Bangladesh", "Bhutan", "China", "Fiji", "India", "Indonesia", "Malaysia", "Nepal", "Philippines", "Thailand", "Vanuatu"

References

World Bank national accounts data, and OECD National Accounts data files. URL: https://data.worldbank.org/indicator/NY.C
World Bank, Doing Business Project (http://www.doingbusiness.org/ExploreTopics/PayingTaxes/).
URL: https://data.worldbank.org/indicator/IC.TAX.PRFT.CP.ZS

See Also

`graph.flm`

Examples

```
data("GDPtax")
# Plot data in groups
for(i in 1:4)
  assign(paste0("p", i), plot(subset(GDPtax$GDP, GDPtax$Group == i)) +
    ggplot2::labs(title=paste("Group ", i, sep=""), y="GDP"))
p4
if(require("patchwork", quietly=TRUE))
  p1 + p2 + p3 + p4
```

`GET.cdf` *Test of independence based on cumulative distribution function*

Description

Permutation-based test of independence in a bivariate vector using the empirical joint cumulative distribution function as the test statistic.

Usage

```
GET.cdf(X, ngrid = c(20, 20), nsim = 999, seq.x = NULL, seq.y = NULL, ...)
```

Arguments

`X`	A matrix with n rows and 2 columns. Each row contains one bivariate observation.
`ngrid`	Vector with two elements, giving the number of grid points to be used in the test statistic for each of the two marginals. The default is 20 in each marginal.
`nsim`	The number of random permutations used.
`seq.x`	For the first marginal, the values at which the empirical cumulative distribution function will be evaluated. If NULL (the default), sequence of quantiles will be used, equidistant in terms of probability.
`seq.y`	For the second marginal, the values at which the empirical cumulative distribution function will be evaluated. If NULL (the default), sequence of quantiles will be used, equidistant in terms of probability.
`...`	Additional parameters to be passed to `global_envelope_test`. In particularly, `alpha` specifies the nominal significance level of the test, and `type` the type of the global envelope test.

Details

Permutation-based test of independence in a bivariate sample, based on empirical joint cumulative distribution function computed on a grid of `ngrid[1]` times `ngrid[2]` arguments. The grid points are chosen according to the quantiles of the marginal distributions.

If the observed data are the pairs $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, the permutations are obtained by randomly permuting the values in the second marginal, i.e. $\{(X_1, Y_{\pi(1)}), \ldots, (X_n, Y_{\pi(n)})\}$.

The test itself is performed using the global envelope test of the chosen version, see the argument `type` of `global_envelope_test`.

References

Dvořák, J. and Mrkvička, T. (2022). Graphical tests of independence for general distributions. Computational Statistics 37, 671–699.

Examples

```
# Generate sample data
data <- matrix(rnorm(n=200), ncol=2) %*% matrix(c(1,1,0,1), ncol=2)
plot(data)

# Compute the CDF test and plot the significant regions
res <- GET.cdf(data, ngrid=c(20,15), nsim=1999)

plot(res)

# Extract the p-value
attr(res,"p")
```

GET.composite *Adjusted global envelope tests*

Description

Adjusted global envelope tests for composite null hypothesis.

Usage

```
GET.composite(
  X,
  X.ls = NULL,
  nsim = 499,
  nsimsub = nsim,
  simfun = NULL,
  fitfun = NULL,
  calcfun = function(X) {
     X
 },
  testfuns = NULL,
  ...,
  type = "erl",
  alpha = 0.05,
  alternative = c("two.sided", "less", "greater"),
  probs = c(0.025, 0.975),
  r_min = NULL,
  r_max = NULL,
  take_residual = FALSE,
  save.cons.envelope = savefuns,
  savefuns = FALSE,
  verbose = TRUE,
  MrkvickaEtal2017 = FALSE,
  mc.cores = 1L
)
```

Arguments

`X`	An object containing the data in some form. A `curve_set` (see `create_curve_set`) or an `envelope` object (of the **spatstat** package), as the `curve_sets` argument of `global_envelope_test` (need to provide `X.ls`), or a fitted point process model of **spatstat** (e.g. object of class `ppm` or `kppm`), or a point pattern object of class `ppp` of **spatstat**, or another data object (need to provide `simfun`, `fitfun`, `calcfun`).
`X.ls`	A list of objects as `curve_sets` argument of `global_envelope_test`, giving the second stage simulations, see details.
`nsim`	The number of simulations to be generated in the primary test. Ignored if `X.ls` provided.
`nsimsub`	Number of simulations in each basic test. There will be `nsim` repetitions of the basic test, each involving `nsimsub` simulated realisations. Total number of simulations will be nsim * (nsimsub + 1).
`simfun`	A function for generating simulations from the null model. If given, this function is called by `replicate(n=nsim, simfun(simfun.arg), simplify=FALSE)` to make nsim simulations. Here `simfun.arg` is obtained by `fitfun(X)`.
`fitfun`	A function for estimating the parameters of the null model. The function should return the fitted model in the form that it can be directly passed to `simfun` as its argument.
`calcfun`	A function for calculating a summary function from a simulation of the model. The default is the identity function, i.e. the simulations from the model are functions themselves. The use of `calcfun` is still experimental. Preferably provide `X` and `X.ls` instead, if `X` is not a point pattern or fitted point process model object of **spatstat**.
`testfuns`	A list of lists of parameters to be passed to the `envelope` function of **spatstat** if `X` is a point pattern of a fitted point process model of **spatstat**. A list of parameters should be provided for each test function that is to be used in the combined test.
`...`	Additional parameters to the `envelope` function of **spatstat** in the case where only one test function is used. In that case, this is an alternative to providing the parameters in the argument testfuns. If `envelope` is also used to generate simulations under the null hypothesis (if simfun not provided), then also recall to specify how to generate the simulations.
`type`	The type of the global envelope with current options for 'rank', 'erl', 'cont', 'area', 'qdir', 'st' and 'unscaled'. See details.
`alpha`	The significance level. The 100(1-alpha)% global envelope will be calculated. If a vector of values is provided, the global envelopes are calculated for each value.
`alternative`	A character string specifying the alternative hypothesis. Must be one of the following: "two.sided" (default), "less" or "greater". The last two options only available for types `'rank'`, `'erl'`, `'cont'` and `'area'`.
`probs`	A two-element vector containing the lower and upper quantiles for the measure 'q' or 'qdir', in that order and on the interval [0, 1]. The default values are 0.025 and 0.975, suggested by Myllymäki et al. (2015, 2017).

r_min The minimum argument value to include in the test.

r_max The maximum argument value to include in the test. in calculating functions by the envelope function of **spatstat**.

take_residual Logical. If TRUE (needed for visual reasons only) the mean of the simulated functions is reduced from the functions in each first and second stage test.

save.cons.envelope
Logical flag indicating whether to save the unadjusted envelope test results.

savefuns Logical flag indicating whether to save all the simulated function values. Similar to the same argument of the envelope function of **spatstat**.

verbose Logical flag indicating whether to print progress reports during the simulations. Similar to the same argument of envelope function of **spatstat**.

MrkvickaEtal2017
Logical. If TRUE, type is "st" or "qdir" and several test functions are used, then the combined scaled MAD envelope presented in Mrkvička et al. (2017) is calculated. Otherwise, the two-step procedure described in global_envelope_test is used for combining the tests. Default to FALSE. The option is kept for historical reasons.

mc.cores The number of cores to use, i.e. at most how many child processes will be run simultaneously. Must be at least one, and parallelization requires at least two cores. On a Windows computer mc.cores must be 1 (no parallelization). For details, see mclapply, for which the argument is passed. Parallelization can be used in generating simulations and in calculating the second stage tests.

Details

The specification of X, X.ls, fitfun, simfun is important:

- If X.ls is provided, then the global envelope test is calculated based on functions in these objects. X should be a curve_set (see create_curve_set) or an envelope object of **spatstat** including the observed function and simulations from the tested model. X.ls should be a list of curve_set or envelope (of R package **spatstat**) objects, where each component contains an "observed" function f that has been simulated from the model fitted to the data and the simulations that have been obtained from the same model that has been fitted to the "observed" f. The user has the responsibility that the functions have been generated correctly, the test is done based on these provided simulations. See the examples.
- Otherwise, if simfun and fitfun are provided, X can be general. The function fitfun is used for fitting the desired model M and the function simfun for simulating from a fitted model M. These functions should be coupled with each other such that the object returned by fitfun is directly accepted as the (single) argument in simfun. In the case, that the global envelope should not be calculated directly for X (X is not a function), calcfun can be used to specify how to calculate the function from X and from simulations generated by simfun. Special attention is needed in the correct specification of the functions, see examples.
- Otherwise, X should be either a fitted (point process) model object or a ppp object of the R package **spatstat**.
  - If X is a fitted (point process) model object of the R package **spatstat**, then the simulations from this model are generated and summary functions for testing calculated by the

envelope function of **spatstat**. Which summary function to use and how to calculate it, can be passed to envelope either in ... or testfuns. Unless otherwise specified the default function of envelope, i.g. the K-function, is used. The argument testfuns should be used to specify the test functions in the case where one wants to base the test on several test functions.

– If X is a ppp object of **spatstat**, then the envelope function is used for simulations and model fitting and the complete spatial randomness (CSR) is tested (with intensity parameter).

For the rank envelope test, the global envelope test is the test described in Myllymäki et al. (2017) with the adjustment of Baddeley et al. (2017). For other test types, the test (also) uses the two-stage procedure of Dao and Genton (2014) with the adjustment of Baddeley et al. (2017) as descripbed in Myllymäki and Mrkvička (2020).

See examples also in saplings.

Value

An object of class global_envelope or combined_global_envelope, which can be printed and plotted directly. See global_envelope_test.

References

Baddeley, A., Hardegen, A., Lawrence, T., Milne, R. K., Nair, G. and Rakshit, S. (2017). On two-stage Monte Carlo tests of composite hypotheses. Computational Statistics and Data Analysis 114: 75-87. doi: http://dx.doi.org/10.1016/j.csda.2017.04.003

Dao, N.A. and Genton, M. (2014). A Monte Carlo adjusted goodness-of-fit test for parametric models describing spatial point patterns. Journal of Graphical and Computational Statistics 23, 497-517.

Mrkvička, T., Myllymäki, M. and Hahn, U. (2017) Multiple Monte Carlo testing, with applications in spatial point processes. Statistics & Computing 27(5): 1239-1255. DOI: 10.1007/s11222-016-9683-9

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017). Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 79: 381-404. doi: 10.1111/rssb.12172

Myllymäki, M. and Mrkvička, T. (2020). GET: Global envelopes in R. arXiv:1911.06583 [stat.ME]

See Also

global_envelope_test, plot.global_envelope, saplings

Examples

```
# Graphical normality test (Myllymaki and Mrkvicka, 2020, Section 3.3.)
#=========================
if(require("fda.usc", quietly=TRUE)) {
  data("poblenou")
  dat <- poblenou[['nox']][['data']][,'H10']
  n <- length(dat)
```

```
  # The number of simulations
  nsim <- nsimsub <- 199


  set.seed(200127)
  # General setup
  #==============
  # 1. Fit the model
  mu <- mean(dat)
  sigma <- sd(dat)
  # 2. Simulate a sample from the fitted null model and
  #    compute the test vectors for data (obs) and each simulation (sim)
  r <- seq(min(dat), max(dat), length=100)
  obs <- stats::ecdf(dat)(r)
  sim <- sapply(1:nsimsub, function(i) {
    x <- rnorm(n, mean = mu, sd = sigma)
    stats::ecdf(x)(r)
  })
  cset <- create_curve_set(list(r = r, obs = obs, sim_m = sim))

  # 3. Simulate another sample from the fitted null model.
  # 4. Fit the null model to each of the patterns,
  #    simulate a sample from the null model,
  #    and compute the test vectors for all.
  cset.ls <- list()
  for(rep in 1:nsim) {
    x <- rnorm(n, mean = mu, sd = sigma)
    mu2 <- mean(x)
    sigma2 <- sd(x)
    obs2 <- stats::ecdf(x)(r)
    sim2 <- sapply(1:nsimsub, function(i) {
      x2 <- rnorm(n, mean = mu2, sd = sigma2)
      stats::ecdf(x2)(r)
    })
    cset.ls[[rep]] <- create_curve_set(list(r = r, obs = obs2, sim_m = sim2))
  }
  # Perform the adjusted test
  res <- GET.composite(X = cset, X.ls = cset.ls, type = 'erl')
  plot(res) + ggplot2::labs(x = "NOx", y = "Ecdf")
}

# Example of a point pattern data
#===============================
# Test the fit of a Matern cluster process.

if(require("spatstat.model", quietly=TRUE)) {
  data("saplings")
  saplings <- as.ppp(saplings, W = square(75))

  # First choose the r-distances
  rmin <- 0.3; rmax <- 10; rstep <- (rmax-rmin)/500
  r <- seq(0, rmax, by = rstep)
```

```
  # Number of simulations
  nsim <- 19 # Increase nsim for serious analysis!

  # Option 1: Give the fitted model object to GET.composite
  #-----------------------------------------------------------
  # This can be done and is preferable when the model is
  # a point process model of spatstat.
  # 1. Fit the Matern cluster process to the pattern
  # (using minimum contrast estimation with the K-function)
  M1 <- kppm(saplings~1, clusters = "MatClust", statistic = "K")
  summary(M1)
  # 2. Make the adjusted global area rank envelope test using the L(r)-r function
  adjenvL <- GET.composite(X = M1, nsim = nsim,
              testfuns = list(L =list(fun="Lest", correction="translate",
                            transform=expression(.-r), r=r)), # passed to envelope
              type = "area", r_min = rmin, r_max = rmax)
  # Plot the test result
  plot(adjenvL)

  # Option 2: Generate the simulations "by yourself"
  #--------------------------------------------------
  # and provide them as curve_set or envelope objects
  # Preferable when you want to have a control
  # on the simulations yourself.
  # 1. Fit the model
  M1 <- kppm(saplings~1, clusters = "MatClust", statistic = "K")
  # 2. Generate nsim simulations by the given function using the fitted model
  X <- envelope(M1, nsim = nsim, savefuns = TRUE,
                fun = "Lest", correction = "translate",
                transform = expression(.-r), r = r)
  plot(X)
  # 3. Create another set of simulations to be used to estimate
  # the second-state p-value (as proposed by Baddeley et al., 2017).
  simpatterns2 <- simulate(M1, nsim = nsim)
  # 4. Calculate the functions for each pattern
  simf <- function(rep) {
    # Fit the model to the simulated pattern Xsims[[rep]]
    sim_fit <- kppm(simpatterns2[[rep]], clusters = "MatClust", statistic = "K")
    # Generate nsim simulations from the fitted model
    envelope(sim_fit, nsim = nsim, savefuns = TRUE,
             fun = "Lest", correction = "translate",
             transform = expression(.-r), r = r)
  }
  X.ls <- parallel::mclapply(X = 1:nsim, FUN = simf, mc.cores = 1) # list of envelope objects
  # 5. Perform the adjusted test
  res <- GET.composite(X = X, X.ls = X.ls, type = "area",
                    r_min = rmin, r_max = rmax)
  plot(res)
}
```

GET.contingency *Test of independence in a 2D contingency table*

Description

Permutation-based test of independence in a 2D contingency table, using the matrix of observed counts as the test statistic.

Usage

```
GET.contingency(X, nsim = 999, ...)
```

Arguments

`X`	A matrix with n rows and 2 columns. Each row contains one bivariate observation.
`nsim`	The number of random permutations used.
`...`	Additional parameters to be passed to `global_envelope_test`. In particularly, `alpha` specifies the nominal significance level of the test, and `type` the type of the global envelope test.

Details

Permutation-based test of independence in a 2D contingency table, using the matrix of observed counts as the test statistic.

If the observed data are the pairs $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, the permutations are obtained by randomly permuting the values in the second marginal, i.e. $\{(X_1, Y_{\pi(1)}), \ldots, (X_n, Y_{\pi(n)})\}$.

The test itself is performed using the global envelope test in the chosen version. Text output can be printed in the console by typing the object name. The cells in which the observed value exceeds the upper envelope printed in red, and cells in which the observed value is lower than the lower envelope printed in cyan. Standard output of the global envelope test is also returned and can be plotted or analyzed accordingly.

References

Dvořák, J. and Mrkvička, T. (2022). Graphical tests of independence for general distributions. Computational Statistics 37, 671–699.

Examples

```
# Generate sample data:
data <- matrix(c(sample(4, size=100, replace=TRUE), sample(2, size=100, replace=TRUE)), ncol=2)
data[,2] <- data[,2] + data[,1]

# Observed contingency table (with row names and column names)
table(data[,1], data[,2])

# Permutation-based envelope test
res <- GET.contingency(data, nsim=999)

res
plot(res)
```

```
# Extract the p-value
attr(res,"p")
```

GET.localcor *The test of local correlations*

Description

The test of local correlations using Vilodomat et al. (2014) procedure for resamples and the FDR envelope of Mrkvička and Myllymäki (2022).

Usage

```
GET.localcor(
  data,
  Delta,
  nsim = 1000,
  typeone = c("fdr", "fwer"),
  varying.bandwidth = FALSE,
  bandwidth.nn = 0.1,
  bandwidth.h = 5.281,
  maxk = 300,
  savefuns = FALSE,
  N_s = 1000,
  mc.cores = 1L,
  mc.args = NULL,
  cl = NULL,
  notest = FALSE,
  ...
)
```

Arguments

data	A data.frame where the first two columns correspond to the values of the two random fields, whose correlations are to be studied, and the third and fourth columns correspond to the x- and y-coordinates where these random fields have been observed. In addition, the width and height of the pixels at each (x,y) can be given in the fifth and sixth column. Warning: no checks for the data input.
Delta	A smoothing parameter of the local correlation. According to Vilodomat et al. (2014): Delta is a set of values for the proportion of neighbors to consider for the smoothing step. No default. The user may have to experiment with different values to find one suitable for their data.
nsim	The number of resamples.
typeone	Character string indicating which type I error rate to control, either the familywise error rate ('fwer') or false discovery rate ('fdr'). Further arguments to the FWER or FDR envelope can be passed in argument `GET.args`. If 'fwer', the type of the envelope can be chosen by specifying the argument `type` in `GET.args`.

`varying.bandwidth`
: Logical, whether to use a varying bandwidth to calculate the local correlations or not. See Vilodomat et al. (2014).

`bandwidth.nn`
: Nearest neighbor component of the smoothing parameter for varying bandwidth to be passed to the argument `nn` of the function `lp` of the **locfit** package. The user may have to experiment with different values to find one suitable for their data. Default set to to 0.1 according to Vilodamat et al. (2014, supporting information).

`bandwidth.h`
: Non-varying bandwidth, to be passed to the argument `h` of the function `lp` of **locfit**. The user may have to experiment with different values to find one suitable for their data. Default to 5.281 according to Vilodamat et al. (2014, supporting information).

`maxk`
: See `locfit` and `locfit.raw` of **locfit**. Default here to 300 following Vilodomat et al. (2014).

`savefuns`
: Logical. If TRUE, then the functions from permutations are saved to the attribute simfuns.

`N_s`
: If the number of observations is bigger than `N_s`, following Vilodomat et al. (2014) a subsample of size N_s is taken every time when a variogram is calculated.

`mc.cores`
: The number of cores to use, i.e. at most how many child processes will be run simultaneously. Must be at least one, and parallelization requires at least two cores. On a Windows computer mc.cores must be 1 (no parallelization). For details, see `mclapply`, for which the argument is passed. Parallelization can be used in generating simulations and in calculating the second stage tests.

`mc.args`
: A named list of additional arguments to be passed to `mclapply`. Only relevant if `mc.cores` is more than 1.

`cl`
: Allows parallelization through the use of `parLapply` (works also in Windows), see the argument `cl` there, and examples.

`notest`
: Logical. FALSE means that the test is done. TRUE allows to calculate only local correlation for the data, which can be beneficial for choosing the bandwidths before running the test. If TRUE, then only the observed local correlations will be returned.

`...`
: Additional parameters to be passed to `fdr_envelope` (if typeone = "fdr") or to `global_envelope_test` (if typeone = "fwer").

Details

The code is a modification of the supporting information code of Vilodomat et al. (2014) available at https://doi.org/10.1111/biom.12139. The modification includes the FDR (or FWER, if specified by the argument `typeone`) envelopes for the test of local correlations, i.e. multiple testing correction and graphical illustration of the test results.

Variograms are calculated using the package **geoR** and the local correlations using the R package **locfit**. These packages should be installed to use `GET.localcor`.

Currently the data is provided in the format of Vilodomat et al. (2014, Supporting information). Additionally width and height of area represented by a data point can be provided, see the argument `data`. This information is used for plotting purposes when plotting the output by `plot()`.

Examples will be provided in a vignette.

Value

A global envelope object (with possible additional classes), see description of main components in global_envelope (Value). Additional attributes: p_global contains the Monte Carlo p-value for the global test of correlation. cor_global and cor_global_sim contain the value of the correlation for data and permuted data, respectively. If savefuns = TRUE, then permutations contain the permuted values of the first random field according to Viladomat et al. (2014) procedure, and cset contains all the local correlations for the data and permuted data in a curve_set object (see create_curve_set).

References

Viladomat, J., Mazumder, R., McInturff, A., McCauley, D.J. and Hastie, T. (2014). Assessing the significance of global and local correlations under spatial autocorrelation: A nonparametric approach. Biometrics 70, 409-418. doi: 10.1111/biom.12139

Mrkvička and Myllymäki (2022). False discovery rate envelopes. arXiv:2008.10108 [stat.ME]

GET.necdf *Graphical n sample test of correspondence of distribution functions*

Description

Compare the distributions of two (or more) samples.

Usage

```
GET.necdf(
  x,
  r = seq(min(unlist((lapply(x, min)))), max(unlist((lapply(x, max)))), length = 100),
  contrasts = FALSE,
  nsim,
  ...
)
```

Arguments

x	A list of numeric vectors, one for each sample.
r	The sequence of argument values at which the distribution functions are to be compared. The default is 100 equally spaced values between the minimum and maximum over all groups.
contrasts	Logical. FALSE and TRUE specify the two test functions as described in description part of this help file.
nsim	The number of random permutations.
...	Additional parameters to be passed to global_envelope_test (if typeone = "fwer") or fdr_envelope (if typeone = "fdr").

Details

A global envelope test can be performed to investigate whether the n distribution functions differ from each other and how do they differ. This test is a generalization of the two-sample Kolmogorov-Smirnov test with a graphical interpretation. We assume that the observations in the sample i are an i.i.d. sample from the distribution $F_i(r), i = 1, \dots, n$, and we want to test the hypothesis

$$F_1(r) = \cdots = F_n(r).$$

If `contrasts = FALSE` (default), then the test statistic is taken to be

$$\mathbf{T} = (\hat{F}_1(r), \dots, \hat{F}_n(r))$$

where $\hat{F}_i(r) = (\hat{F}_i(r_1), \dots, \hat{F}_i(r_k))$ is the ecdf of the ith sample evaluated at argument values $r = (r_1, \dots, r_k)$. This is our recommended test function for the test. Another possibility is given by `contrasts = TRUE`, and then the test statistic is contructed from all pairwise differences,

$$\mathbf{T} = (\hat{F}_1(r) - \hat{F}_2(r), \hat{F}_1(r) - \hat{F}_3(r), \dots, \hat{F}_{n-1}(r) - \hat{F}_n(r))$$

The simulations under the null hypothesis that the distributions are the same are obtained by permuting the individuals of the groups. The default number of permutation, if nsim is not specified, is n*1000 - 1 for the case `contrasts = FALSE` and (n*(n-1)/2)*1000 - 1 for the case `contrasts = TRUE`, where n is the length of x.

Examples

```
if(require(fda, quietly=TRUE)) {
  # Heights of boys and girls at age 10
  f.a <- growth$hgtf["10",] # girls at age 10
  m.a <- growth$hgtm["10",] # boys at age 10
  # Empirical cumulative distribution functions
  plot(ecdf(f.a))
  plot(ecdf(m.a), col='grey70', add=TRUE)
  # Create a list of the data
  fm.list <- list(Girls=f.a, Boys=m.a)

  res_m <- GET.necdf(fm.list)
  plot(res_m)
  res_c <- GET.necdf(fm.list, contrasts=TRUE)
  plot(res_c)



  # Heights of boys and girls at age 14
  f.a <- growth$hgtf["14",] # girls at age 14
  m.a <- growth$hgtm["14",] # boys at age 14
  # Empirical cumulative distribution functions
  plot(ecdf(f.a))
  plot(ecdf(m.a), col='grey70', add=TRUE)
  # Create a list of the data
  fm.list <- list(Girls=f.a, Boys=m.a)

  res_m <- GET.necdf(fm.list)
```

```
  plot(res_m)
  res_c <- GET.necdf(fm.list, contrasts=TRUE)
  plot(res_c)


}
```

GET.qq *Test of independence based on the smoothed Q-Q plot*

Description

Permutation-based test of independence in a bivariate vector using the smoothed Q-Q plot as the test statistic.

Usage

```
GET.qq(
  X,
  ngrid = c(64, 64),
  nsim = 999,
  sigma = NULL,
  atoms.x = NULL,
  atoms.y = NULL,
  ...
)
```

Arguments

X	A matrix with n rows and 2 columns. Each row contains one bivariate observation.
ngrid	Vector with two elements, giving the number of grid points to be used in the test statistic for each of the two marginals. The default is 64 in each marginal.
nsim	The number of random permutations used.
sigma	Standard deviation of the smoothing kernel to be used for smoothing the Q-Q plot when computing the test statistic. If NULL, sensible default value is used based on the number of observations.
atoms.x	Vector specifying atomic values in the first marginal. See Examples.
atoms.y	Vector specifying atomic values in the second marginal. See Examples.
...	Additional parameters to be passed to global_envelope_test. In particularly, alpha specifies the nominal significance level of the test, and type the type of the global envelope test.

Details

Permutation-based test of independence in a bivariate sample, based on Q-Q representation and estimate of the intensity function computed on a regular grid of `ngrid[1]` times `ngrid[2]` points.

If the observed data are the pairs $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, the permutations are obtained by randomly permuting the values in the second marginal, i.e. $\{(X_1, Y_{\pi(1)}), \ldots, (X_n, Y_{\pi(n)})\}$.

The test itself is performed using the global envelope test in the chosen version.

References

Dvořák, J. and Mrkvička, T. (2022). Graphical tests of independence for general distributions. Computational Statistics 37, 671–699.

Examples

```
# Generate sample data
data <- matrix(rnorm(n=200), ncol=2) %*% matrix(c(1,1,0,1), ncol=2)

plot(data)

# Compute the QQ test and plot the significant regions
res <- GET.qq(data, ngrid=c(30,20), nsim=999)

plot(res)
# Extract the p-value
attr(res,"p")

# With atoms, independent
data <- cbind(rnorm(n=100), sample(4, size=100, replace=TRUE))
plot(data)
res <- GET.qq(data, nsim=999, atoms.y=c(1,2,3,4))

plot(res)


# With atoms, dependent
data <- cbind(sort(rnorm(n=100)), sort(sample(4, size=100, replace=TRUE)))
plot(data)
res <- GET.qq(data, nsim=999, atoms.y=c(1,2,3,4))
plot(res)


# Atoms in both variables
data <- cbind(rnorm(n=100), rnorm(n=100)) %*% matrix(c(1,1,0,1), ncol=2)
data[,1][data[,1]<=-1] <- -1
data[,2][data[,2]<=-0.5] <- -0.5
plot(data)

# Perform the test
res <- GET.qq(data, nsim=999, atoms.x=c(-1), atoms.y=c(-0.5), sigma=NULL)

plot(res)
```

GET.spatialF *Testing global and local dependence of point patterns on covariates*

Description

Compute the spatial F- and S-statistics and perform the one-stage global envelope tests proposed by Myllymäki et al. (2020).

Usage

```
GET.spatialF(
  X,
  formula.full,
  formula.reduced,
  fitfun,
  covariates,
  nsim,
  bw = spatstat.explore::bw.scott(X),
  bw.S = bw,
  dimyx = NULL,
  ...
)
```

Arguments

X	A ppp object of **spatstat** representing the observed point pattern.
formula.full	A formula for the trend of the full model.
formula.reduced	A formula for the trend of the reduced model that is a submodel of the full model.
fitfun	A function of a point pattern, model formula and covariates, giving a fitted model object that can be used with `simulate`.
covariates	A list of covariates.
nsim	The number of simulations.
bw	The bandwidth for smoothed residuals.
bw.S	The radius for the local S(u)-statistic.
dimyx	Pixel array dimensions for smoothed residuals. See `as.mask` of **spatstat**.
...	Additional arguments to be passed to `global_envelope_test`.

Value

list with three components

- F = the global envelope test based on the F(u) statistic
- S = the global envelope test based on the S(u) statistic
- coef = the coefficients of the full model given by fitfun

References

Myllymäki, M., Kuronen, M. and Mrkvička, T. (2020). Testing global and local dependence of point patterns on covariates in parametric models. Spatial Statistics 42, 100436. doi: 10.1016/j.spasta.2020.100436

Examples

```
if(require("spatstat.model", quietly=TRUE)) {
  # Example of tropical rain forest trees
  data("bei")

  fullmodel <- ~ grad
  reducedmodel <- ~ 1
  fitppm <- function(X, model, covariates) {
    ppm(X, model, covariates=covariates)
  }


  nsim <- 19 # Increase nsim for serious analysis!
  res <- GET.spatialF(bei, fullmodel, reducedmodel, fitppm, bei.extra, nsim)

  plot(res$F)
  plot(res$S)


  # Example of forest fires
  data("clmfires")
  # Choose the locations of the lightnings in years 2004-2007:
  pp.lightning <- unmark(subset(clmfires, cause == "lightning" &
                   date >= "2004-01-01" & date < "2008-01-01"))

  covariates <- clmfires.extra$clmcov100
 covariates$forest <- covariates$landuse == "conifer" | covariates$landuse == "denseforest" |
                        covariates$landuse == "mixedforest"

  fullmodel <- ~ elevation + landuse
  reducedmodel <- ~ landuse
  nsim <- 19 # Increase nsim for serious analysis!
  res <- GET.spatialF(pp.lightning, fullmodel, reducedmodel, fitppm, covariates, nsim)
  plot(res$F)
  plot(res$S)

  # Examples of the fitfun functions for clustered and regular processes
  # fitfun for the log Gaussian Cox Process with exponential covariance function
  fitLGCPexp <- function(X, model, covariates) {
    kppm(X, model, clusters="LGCP", model="exponential", covariates=covariates)
  }
  # fitfun for the hardcore process with hardcore radius 0.01
  fitHardcore <- function(X, model, covariates) {
    ppm(X, model, interaction=Hardcore(0.01), covariates=covariates)
  }

}
```

GET.variogram *Variogram and residual variogram with global envelopes*

Description

The function accompanies the function variogram with global envelopes that are based on permutations of the variable(s) or residuals for which the variogram is calculated. Therefore, one can inspect the hypothesis of "no spatial autocorrelation" of the variable or the residuals of the fitted model.

Usage

```
GET.variogram(
  object,
  nsim = 999,
  data = NULL,
  ...,
  GET.args = NULL,
  savefuns = TRUE
)
```

Arguments

object	An object of class gstat or a variogram.formula. In the first case, direct (residual) variograms are calculated for the variable defined in object. Only one variable allowed. In the second case, a formula defining the response vector and (possible) regressors, in case of absence of regressors, use e.g. z~1. See variogram.
nsim	The number of permutations.
data	A data frame where the names in formula are to be found. If NULL, the data are assumed to be found in the object.
...	Additional parameters to be passed to variogram.
GET.args	A named list of additional arguments to be passed to global_envelope_test.
savefuns	Logical. If TRUE, then the functions from permutations are saved to the attribute simfuns.

Examples

```
if(require("sp", quietly=TRUE) & require("gstat", quietly=TRUE)) {
  # Examples from gstat complemented with global envelopes
  #-------------------------------------------------------
  data("meuse")
  coordinates(meuse) <- ~x+y
  # topsoil zinc concentration, mg kg-1 soil ("ppm")
  bubble(meuse, "zinc",
         col=c("#00ff0088", "#00ff0088"), main="zinc concentrations (ppm)")
```

```
  # Variogram can be calculated as follows by the function variogram of the gstat package.
  # The function variogram takes a formula as its first argument:
  # log(zinc)~1 means that we assume a constant trend for the variable log(zinc).
  lzn.vgm <- variogram(object=log(zinc)~1, data=meuse)
  plot(lzn.vgm)
  # Variogram with global envelopes is as easy:
  lzn.vgm.GET <- GET.variogram(object=log(zinc)~1, data=meuse)

  plot(lzn.vgm.GET)

  # Instead of the constant mean, denoted by ~1, a mean function can
  # be specified, e.g. using ~sqrt(dist) as a predictor variable:
  lznr.vgm <- variogram(log(zinc)~sqrt(dist), meuse)
  # In this case, the variogram of residuals with respect
  # to a fitted mean function are shown.
  plot(lznr.vgm)
  # The variogram with global envelopes (obtained by permuting the residuals):
  lznr.vgm.GET <- GET.variogram(object=log(zinc)~sqrt(dist), data=meuse)

  plot(lznr.vgm.GET)

  # Directional variograms
  lzn.dir <- variogram(object=log(zinc)~1, data=meuse, alpha=c(0, 45, 90, 135))
  plot(lzn.dir)
  # with global envelopes
  lzn.dir.GET <- GET.variogram(object=log(zinc)~1, data=meuse, alpha=c(0, 45, 90, 135))

  plot(lzn.dir.GET)

  # Use instead gstat objects
  g <- gstat(id="ln.zinc", formula=log(zinc)~1, data=meuse)
  # or: g <- gstat(id="ln.zinc", formula=log(zinc)~sqrt(dist), data=meuse)
  # The variogram
  plot(variogram(g))
  # The variogram with global envelopes:
  g.GET <- GET.variogram(object=g)

  plot(g.GET)
}
```

`global_envelope_test` *Global envelope test*

Description

Global envelope test, global envelopes and p-values

Usage

```
global_envelope_test(
```

```
  curve_sets,
  type = "erl",
  alpha = 0.05,
  alternative = c("two.sided", "less", "greater"),
  ties = "erl",
  probs = c(0.025, 0.975),
  quantile.type = 7,
  central = "mean",
  nstep = 2,
  ...
)
```

Arguments

`curve_sets`	A `curve_set` (see `create_curve_set`) or an `envelope` object of **spatstat** containing a data function and simulated functions. If an envelope object is given, it must contain the summary functions from the simulated patterns which can be achieved by setting `savefuns = TRUE` when calling the `envelope` function. Alternatively, a list of `curve_set` or `envelope` objects can be given.
`type`	The type of the global envelope with current options for 'rank', 'erl', 'cont', 'area', 'qdir', 'st' and 'unscaled'. See details.
`alpha`	The significance level. The 100(1-alpha)% global envelope will be calculated. If a vector of values is provided, the global envelopes are calculated for each value.
`alternative`	A character string specifying the alternative hypothesis. Must be one of the following: "two.sided" (default), "less" or "greater". The last two options only available for types `'rank'`, `'erl'`, `'cont'` and `'area'`.
`ties`	The method to obtain a unique p-value when `type = 'rank'`. Possible values are 'midrank', 'random', 'conservative', 'liberal' and 'erl'. For 'conservative' the resulting p-value will be the highest possible. For 'liberal' the p-value will be the lowest possible. For 'random' the rank of the obs within the tied values is uniformly sampled so that the resulting p-value is at most the conservative option and at least the liberal option. For 'midrank' the mid-rank within the tied values is taken. For 'erl' the extreme rank length p-value is calculated. The default is 'erl'.
`probs`	A two-element vector containing the lower and upper quantiles for the measure 'q' or 'qdir', in that order and on the interval [0, 1]. The default values are 0.025 and 0.975, suggested by Myllymäki et al. (2015, 2017).
`quantile.type`	As type argument of `quantile`, how to calculate quantiles for 'q' or 'qdir'.
`central`	Either "mean" or "median". If the curve sets do not contain the component `theo` for the theoretical central function, then the central function (used for plotting only) is calculated either as the mean or median of functions provided in the curve sets. For 'qdir', 'st' and 'unscaled' only the mean is allowed as an option, due to their definition.
`nstep`	1 or 2 for how to contruct a combined global envelope if list of curve sets is provided. 2 (default) for a two-step combining procedure, 1 for one-step.
`...`	Additional parameters to be passed to `central_region`.

Details

Given a curve_set (see create_curve_set for how to create such an object) or an envelope object of **spatstat**, which contains both the data curve (or function or vector) $T_1(r)$ (in the component obs) and the simulated curves $T_2(r), \ldots, T_{s+1}(r)$ (in the component sim_m), the function global_envelope_test performs a global envelope test. The functionality of the function is rather similar to the function central_region, but in addition to ordering the functions from the most extreme one to the least extreme one using different measures and providing the global envelopes with intrinsic graphical interpretation, p-values are calculated for the test. Thus, while central_region can be used to construct global envelopes in a general setting, the function global_envelope_test is devoted to testing as its name suggests.

The function global_envelope_test is the main function for global envelope tests (for simple hypotheses). Different type of global envelope tests can be performed. We use such ordering of the functions for which we are able to construct global envelopes with intrinsic graphical interpretation.

- 'rank': the completely non-parametric rank envelope test (Myllymäki et al., 2017) based on minimum of pointwise ranks
- 'erl': the completely non-parametric rank envelope test based on extreme rank lengths (Myllymäki et al., 2017; Mrkvička et al., 2018) based on number of minimal pointwise ranks
- 'cont': the completely non-parametric rank envelope test based on continuous rank (Hahn, 2015; Mrkvička et al., 2019) based on minimum of continuous pointwise ranks
- 'area': the completely non-parametric rank envelope test based on area rank (Mrkvička et al., 2019) based on area between continuous pointwise ranks and minimum pointwise ranks for those argument (r) values for which pointwise ranks achieve the minimum (it is a combination of erl and cont)
- "qdir", the directional quantile envelope test, protected against unequal variance and asymmetry of T(r) for different distances r (Myllymäki et al., 2015, 2017)
- "st", the studentised envelope test, protected against unequal variance of T(r) for different distances r (Myllymäki et al., 2015, 2017)
- "unscaled", the unscaled envelope (providing a baseline) that has a contant width and that corresponds to the classical maximum deviation test (Ripley, 1981).

The first four types are global rank envelopes. The 'rank' envelope test is a completely non-parametric test, which provides the 100(1-alpha) T(r) on the chosen interval of distances and associated p-values. The other three are modifications of 'rank' to treat the ties in the extreme rank ordering on which the 'rank' test is based on. The last three envelopes are global scaled maximum absolute difference (MAD) envelope tests. The unscaled envelope test leads to envelopes with constant width over the distances r. Thus, it suffers from unequal variance of T(r) over the distances r and from the asymmetry of distribution of T(r). We recommend to use the other global envelope tests available. The unscaled envelope is provided as a reference.

See Myllymäki and Mrkvička (2020, Section 2.), i.e. vignette("GET"), for more detailed description of the measures and the corresponding envelopes.

See vignette("pointpatterns") for examples of point pattern analyses.

Value

Either an object of class "global_envelope" or "combined_global_envelope", similarly as the objects returned by central_region.

The `global_envelope` is essentially a data frame containing columns

- the values of the argument r at which the test was made, copied from the argument `curve_sets` with the corresponding names
- obs = values of the data function, copied from the argument `curve_sets` (unlike for central regions, `obs` always exists for a global envelope test)
- lo = the lower envelope; in case of a vector of alpha values, several 'lo' exist with names paste0("lo.", 100*(1-alpha))
- hi = the upper envelope; in case of a vector of alpha values, several 'lo' exist with names paste0("hi.", 100*(1-alpha))
- central = a central curve as specified in the argument `central`.

Moreover, the returned object has the same attributes as the `global_envelope` object returned by `central_region` and in addition

- p = A point estimate for the p-value (default is the mid-rank p-value).

and in the case that `type = 'rank'` also

- p_interval = The p-value interval $[p_{liberal}, p_{conservative}]$.
- ties = As the argument `ties`.

The `combined_global_envelope` is a list of `global_envelope` objects containing the above mentioned columns and which all together form the global envelope. It has the same attributes as described in `central_region`, and in addition also the p-value `p`. The 2d classes are attached as described in `central_region`.

Procedure

1) First the curves are ranked from the most extreme one to the least extreme one by a measure that is specified by the argument `type`. The options are

- 'rank': extreme ranks (Myllymäki et al., 2017)
- 'erl': extreme rank lengths (Myllymäki et al., 2017; Mrkvička et al., 2018)
- 'cont': continuous ranks (Hahn, 2015; Mrkvička et al., 2019)
- 'area': area ranks (Mrkvička et al., 2019)
- 'qdir': the directional quantile maximum absolute deviation (MAD) measure (Myllymäki et al., 2015, 2017)
- 'st': the studentized MAD measure (Myllymäki et al., 2015, 2017)
- 'unscaled': the unscaled MAD measure (Ripley, 1981)

2) Based on the measures used to rank the functions, the 100(1-alpha)% global envelope is provided. It corresponds to the 100*coverage% central region.

3) P-values: In the case `type="rank"`, based on the extreme ranks $k_i, i = 1, ..., s+1$, the p-interval is calculated. Because the extreme ranks contain ties, there is not just one p-value. The p-interval is given by the most liberal and the most conservative p-value estimate. Also a single p-value is calculated. By default this single p-value is the extreme rank length p-value ("erl") as specified by the argument `ties`. If the case of other measures, a (single) p-value based on the given ordering of the functions is calculated and returned in the attribute `p`.

Number of simulations

For the global "rank" envelope test, Myllymäki et al. (2017) recommended to use at least 2500 simulations for testing at the significance level alpha = 0.05 for single function tests, based on experiments with summary functions for point processes evaluated approximately at 500 argument values. In this case, the width of the p-interval associated with the extreme rank measure tended to be smaller than 0.01. The tests 'erl', 'cont' and 'area', similarly as the MAD deviation/envelope tests 'qdir', 'st' and 'unscaled', allow in principle a lower number of simulations to be used than the test based on extreme ranks ('rank'), because no ties occur for these measures. If affordable, we recommend in any case some thousands of simulations for all the measures to achieve a good power and repeatability of the test. If the dimension of the test functions is higher, also the number of simulations should preferably be higher.

Tests based on several functions

If a list of (suitable) objects are provided in the argument curve_sets, then by default (nstep = 2) the two-step combining procedure is used to perform the combined global test as described in Myllymäki and Mrkvička (2020). If nstep = 1 and the lengths of the multivariate vectors in each component of the list are equal, then the one-step combining procedure is used where the functions are concatenated together into a one long vector.

References

Mrkvička, T., Myllymäki, M. and Hahn, U. (2017). Multiple Monte Carlo testing, with applications in spatial point processes. Statistics & Computing 27(5), 1239-1255. doi: 10.1007/s11222-016-9683-9

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2020) A one-way ANOVA test for functional data with graphical interpretation. Kybernetika 56(3), 432-458. doi: 10.14736/kyb-2020-3-0432

Mrkvička, T., Myllymäki, M., Kuronen, M. and Narisetty, N. N. (2022) New methods for multiple testing in permutation inference for the general linear model. Statistics in Medicine 41(2), 276-297. doi: 10.1002/sim.9236

Myllymäki, M., Grabarnik, P., Seijo, H. and Stoyan. D. (2015). Deviation test construction and power comparison for marked spatial point patterns. Spatial Statistics 11, 19-34. doi: 10.1016/j.spasta.2014.11.004

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017). Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 79, 381–404. doi: 10.1111/rssb.12172

Myllymäki, M. and Mrkvička, T. (2020). GET: Global envelopes in R. arXiv:1911.06583 [stat.ME]

Ripley, B.D. (1981). Spatial statistics. Wiley, New Jersey.

See Also

plot.global_envelope, central_region, GET.composite

Examples

```
# Goodness-of-fit testing for simple hypothesis
if(require("spatstat.explore", quietly=TRUE)) {
  # Testing complete spatial randomness (CSR)
```

```
#==========================================
X <- unmark(spruces)

nsim <- 1999 # Number of simulations


# Illustration of general workflow for simple hypotheses
#=======================================================
# First illustrate the general workflow for the test by this example
# of CSR test for a point pattern X using the empirical L-function.
# Define the argument values at which the functions are evaluated
obs.L <- Lest(X, correction="translate")
r <- obs.L[['r']]
# The test function for the data
obs <- obs.L[['trans']] - r
# Prepare simulations and calculate test functions for them at same r as 'obs'
sim <- matrix(nrow=length(r), ncol=nsim)
for(i in 1:nsim) {
  sim.X <- runifpoint(ex=X) # simulation under CSR
  sim[, i] <- Lest(sim.X, correction="translate", r=r)[['trans']] - r
}
# Create a curve_set containing argument values, observed and simulated functions
cset <- create_curve_set(list(r=r, obs=obs, sim_m=sim))
# Perform the test
res <- global_envelope_test(cset, type="erl")
plot(res) + ggplot2::ylab(expression(italic(hat(L)(r)-r)))

# Simple hypothesis for a point pattern utilizing the spatstat package
#=====================================================================
# Generate nsim simulations under CSR, calculate L-function for the data and simulations
env <- envelope(X, fun="Lest", nsim=nsim,
                savefuns=TRUE, # save the functions
                correction="translate", # edge correction for L
                transform=expression(.-r), # centering
                simulate=expression(runifpoint(ex=X))) # Simulate CSR
# The rank envelope test (ERL)
res <- global_envelope_test(env, type="erl")
# Plot the result
plot(res)

## Advanced use:
# Choose the interval of distances [r_min, r_max] (at the same time create a curve_set from 'env')
cset <- crop_curves(env, r_min=1, r_max=7)
# Do the rank envelope test (erl)
res <- global_envelope_test(cset, type="erl")
plot(res) + ggplot2::ylab(expression(italic(L(r)-r)))

# A combined global envelope test
#================================
# As an example test CSR of the saplings point pattern by means of
# L, F, G and J functions.
data(saplings)
X <- as.ppp(saplings, W=square(75))
```

```
  nsim <- 499 # Number of simulations

  # Specify distances for different test functions
  n <- 500 # the number of r-values
  rmin <- 0; rmax <- 20; rstep <- (rmax-rmin)/n
  rminJ <- 0; rmaxJ <- 8; rstepJ <- (rmaxJ-rminJ)/n
  r <- seq(0, rmax, by=rstep)    # r-distances for Lest
  rJ <- seq(0, rmaxJ, by=rstepJ) # r-distances for Fest, Gest, Jest


  # Perform simulations of CSR and calculate the L-functions
  env_L <- envelope(X, nsim=nsim,
   simulate=expression(runifpoint(ex=X)),
   fun="Lest", correction="translate",
   transform=expression(.-r), # Take the L(r)-r function instead of L(r)
   r=r,                       # Specify the distance vector
   savefuns=TRUE,             # Save the estimated functions
   savepatterns=TRUE)         # Save the simulated patterns
  # Take the simulations from the returned object
  simulations <- attr(env_L, "simpatterns")
  # Then calculate the other test functions F, G, J for each simulated pattern
  env_F <- envelope(X, nsim=nsim, simulate=simulations,
                    fun="Fest", correction="Kaplan", r=rJ,
                    savefuns=TRUE)
  env_G <- envelope(X, nsim=nsim, simulate=simulations,
                    fun="Gest", correction="km", r=rJ,
                    savefuns=TRUE)
  env_J <- envelope(X, nsim=nsim, simulate=simulations,
                    fun="Jest", correction="none", r=rJ,
                    savefuns=TRUE)

  # Crop the curves to the desired r-interval I
  curve_set_L <- crop_curves(env_L, r_min=rmin, r_max=rmax)
  curve_set_F <- crop_curves(env_F, r_min=rminJ, r_max=rmaxJ)
  curve_set_G <- crop_curves(env_G, r_min=rminJ, r_max=rmaxJ)
  curve_set_J <- crop_curves(env_J, r_min=rminJ, r_max=rmaxJ)

  res <- global_envelope_test(curve_sets=list(L=curve_set_L, F=curve_set_F,
                                              G=curve_set_G, J=curve_set_J))
  plot(res)
  plot(res, labels=c("L(r)-r", "F(r)", "G(r)", "J(r)"))
}

# A test based on a low dimensional random vector
#=================================================
# Let us generate some example data.
X <- matrix(c(-1.6,1.6),1,2) # data pattern X=(X_1,X_2)
if(requireNamespace("mvtnorm", quietly=TRUE)) {
  Y <- mvtnorm::rmvnorm(200,c(0,0),matrix(c(1,0.5,0.5,1),2,2)) # simulations
 plot(Y, xlim=c(min(X[,1],Y[,1]), max(X[,1],Y[,1])), ylim=c(min(X[,2],Y[,2]), max(X[,2],Y[,2])))
  points(X, col=2)
```

```
# Test the null hypothesis is that X is from the distribution of Y's (or if it is an outlier).

 # Case 1. The test vector is (X_1, X_2)
 cset1 <- create_curve_set(list(r=1:2, obs=as.vector(X), sim_m=t(Y)))
 res1 <- global_envelope_test(cset1)
 plot(res1)

 # Case 2. The test vector is (X_1, X_2, (X_1-mean(Y_1))*(X_2-mean(Y_2))).
 t3 <- function(x, y) { (x[,1]-mean(y[,1]))*(x[,2]-mean(y[,2])) }
cset2 <- create_curve_set(list(r=1:3, obs=c(X[,1],X[,2],t3(X,Y)), sim_m=rbind(t(Y), t3(Y,Y))))
 res2 <- global_envelope_test(cset2)
 plot(res2)
}
```

---

graph.fanova — *One-way graphical functional ANOVA*

Description

One-way ANOVA tests for functional data with graphical interpretation

Usage

```
graph.fanova(
  nsim,
  curve_set,
  groups,
  typeone = c("fwer", "fdr"),
  variances = "equal",
  contrasts = FALSE,
  n.aver = 1L,
  mirror = FALSE,
  savefuns = FALSE,
  test.equality = c("mean", "var", "cov"),
  cov.lag = 1,
  ...
)
```

Arguments

`nsim`	The number of random permutations.
`curve_set`	The original data (an array of functions) provided as a `curve_set` object (see `create_curve_set`) or a fdata object (see `fdata`). The curve set should include the argument values for the functions in the component `r`, and the observed functions in the component `obs`.
`groups`	The original groups (a factor vector representing the assignment to groups).

typeone — Character string indicating which type I error rate to control, either the familywise error rate ('fwer') or false discovery rate ('fdr'). Further arguments to the FWER or FDR envelope can be passed in argument `GET.args`. If 'fwer', the type of the envelope can be chosen by specifying the argument `type` in `GET.args`.

variances — Either "equal" or "unequal". If "unequal", then correction for unequal variances as explained in details will be done. Only relevant for the case `test.equality = "means"` (default).

contrasts — Logical. FALSE and TRUE specify the two test functions as described in description part of this help file.

n.aver — If variances = "unequal", there is a possibility to use variances smoothed by appying moving average to the estimated sample variances. n.aver determines how many values on each side do contribute (incl. value itself).

mirror — The complement of the argument circular of `filter`. Another parameter for the moving average to estimate sample variances (see `n.aver`).

savefuns — Logical. If TRUE, then the functions from permutations are saved to the attribute simfuns.

test.equality — A character with possible values `mean` (default), `var` and `cov`. If `mean`, the functional ANOVA is performed to compare the means in the groups. If `var`, then the equality of variances of the curves in the groups is tested by performing the graphical functional ANOVA test on the functions

$$Z_{ij}(r) = T_{ij}(r) - \bar{T}_j(r).$$

If `cov`, then the equality of lag `cov.lag` covariance is tested by performing the fANOVA with

$$W_{ij}(r) = \sqrt{|V_{ij}(r)| \cdot sign(V_{ij}(r))},$$

where

$$V_{ij}(r) = (T_{ij}(r) - \bar{T}_j(r))((T_{ij}(r+s) - \bar{T}_j(r+s))).$$

See Mrkvicka et al. (2020) for more details.

cov.lag — The lag of the covariance for testing the equality of covariances, see `test.equality`.

... — Additional parameters to be passed to `global_envelope_test` (if `typeone = "fwer"`) or `fdr_envelope` (if `typeone = "fdr"`).

Details

This function can be used to perform one-way graphical functional ANOVA tests described in Mrkvička et al. (2020). Both 1d and 2d functions are allowed in curve sets.

The tests assume that there are J groups which contain $n_1, \dots, n_J$ functions $T_{ij}, i = \dots, J, j = 1, \dots, n_j$. The functions should be given in the argument `curve_set`, and the groups in the argument `groups`. The tests assume that $T_{ij}, i = 1, ..., n_j$ is an iid sample from a stochastic process with mean function μ_j and covariance function $\gamma_j(s,t)$ for s,t in R and j = 1,..., J.

To test the hypothesis

$$H_0 : \mu_1(r) \equiv \mu_2(r) \equiv \dots \equiv \mu_J(r),$$

you can use the test function

$$\mathbf{T} = (\overline{T}_1(\mathbf{r}), \overline{T}_2(\mathbf{r}), \dots, \overline{T}_J(\mathbf{r}))$$

where $\overline{T}_i(\mathbf{r})$ is a vector of mean values of functions in the group j. This test function is used when `contrasts = FALSE` (default).

The hypothesis can equivalently be written as

$$H_0 : \mu_i(r) - \mu_j(r) = 0, i = 1, \dots, J-1, j = 1, \dots, J.$$

and, alternatively, one can use the test function (vector) taken to consist of the differences of the group averages,

$$\mathbf{T}' = (\overline{T}_1(\mathbf{r}) - \overline{T}_2(\mathbf{r}), \overline{T}_1(\mathbf{r}) - \overline{T}_3(\mathbf{r}), \dots, \overline{T}_{J-1}(\mathbf{r}) - \overline{T}_J(\mathbf{r})).$$

The choice is available with the option `contrasts = TRUE`. This test corresponds to the post-hoc test done usually after an ANOVA test is significant, but it can be directed tested by means of the combined rank test (Mrkvička et al., 2017) with this test vector.

The test as such assumes that the variances are equal across the groups of functions. To deal with unequal variances, the differences are rescaled as the first step as follows

$$S_{ij}(r) = \frac{T_{ij}(r) - \overline{T}(r))}{\sqrt{Var(T_j(r))}} \sqrt{Var(T(r))} + \overline{T}(r))$$

where $\overline{T}(\mathbf{r})$ is the overall sample mean and $\sqrt{Var(T(r))}$ is the overall sample standard deviation. This scaling of the test functions can be obtained by giving the argument `variances = "unequal"`.

References

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2020) A one-way ANOVA test for functional data with graphical interpretation. Kybernetika 56 (3), 432-458. doi: 10.14736/kyb-2020-3-0432

Mrkvička, T., Myllymäki, M., and Hahn, U. (2017). Multiple Monte Carlo testing, with applications in spatial point processes. Statistics and Computing 27 (5): 1239-1255. doi:10.1007/s11222-016-9683-9

Myllymäki, M and Mrkvička, T. (2020). GET: Global envelopes in R. arXiv:1911.06583 [stat.ME]

See Also

frank.fanova

Examples

```
#-- NOx levels example (see for details Myllymaki and Mrkvicka, 2020)
if(require("fda.usc", quietly=TRUE)) {
  # Prepare data
  data("poblenou")
  fest <- poblenou$df$day.festive; week <- as.integer(poblenou$df$day.week)
  Type <- vector(length=length(fest))
  Type[fest == 1 | week >= 6] <- "Free"
  Type[fest == 0 & week %in% 1:4] <- "MonThu"
```

```
  Type[fest == 0 & week == 5] <- "Fri"
  Type <- factor(Type, levels = c("MonThu", "Fri", "Free"))

  # (log) Data as a curve_set
  cset <- create_curve_set(list(r = 0:23,
             obs = t(log(poblenou[['nox']][['data']]))))
  # Graphical functional ANOVA
  nsim <- 2999

  res.c <- graph.fanova(nsim = nsim, curve_set = cset,
                        groups = Type, variances = "unequal",
                        contrasts = TRUE)
  plot(res.c) + ggplot2::labs(x = "Hour", y = "Diff.")
}

#-- Centred government expenditure centralization ratios example
# This is an example analysis of the centred GEC in Mrkvicka et al.
data("cgec")

# Number of simulations
nsim <- 2499 # increase to reduce Monte Carlo error


# Test for unequal lag 1 covariances
res.cov1 <- graph.fanova(nsim = nsim, curve_set = cgec$cgec,
                         groups = cgec$group,
                         test.equality = "cov", cov.lag = 1)
plot(res.cov1)
# Add labels
plot(res.cov1, labels = paste("Group ", 1:3, sep="")) +
 ggplot2::xlab(substitute(paste(italic(i), " (", j, ")", sep=""), list(i="r", j="Year")))
# Test for equality of variances among groups
res.var <- graph.fanova(nsim = nsim, curve_set = cgec$cgec,
                        groups = cgec$group,
                        test.equality = "var")
plot(res.var)

# Test for equality of means assuming equality of variances
# a) using 'means'
res <- graph.fanova(nsim = nsim, curve_set = cgec$cgec,
                    groups = cgec$group,
                    variances = "equal", contrasts = FALSE)
plot(res)
# b) using 'contrasts'
res2 <- graph.fanova(nsim = nsim, curve_set = cgec$cgec,
                     groups = cgec$group,
                     variances = "equal", contrasts = TRUE)
plot(res2)

# Image set examples
data("imageset3")

res <- graph.fanova(nsim = 19, # Increase nsim for serious analysis!
```

```
                      curve_set = imageset3$image_set,
                      groups = imageset3$Group)
plot(res)
# Contrasts
res.c <- graph.fanova(nsim = 19, # Increase nsim for serious analysis!
                        curve_set = imageset3$image_set, groups = imageset3$Group,
                        contrasts = TRUE)
plot(res.c)
```

---

graph.flm *Graphical functional GLM*

Description

Non-parametric graphical tests of significance in functional general linear model (GLM)

Usage

```
graph.flm(
  nsim,
  formula.full,
  formula.reduced,
  typeone = c("fwer", "fdr"),
  curve_sets,
  factors = NULL,
  contrasts = FALSE,
  savefuns = FALSE,
  lm.args = NULL,
  GET.args = NULL,
  mc.cores = 1L,
  mc.args = NULL,
  cl = NULL,
  fast = TRUE
)
```

Arguments

nsim	The number of random permutations.
formula.full	The formula specifying the general linear model, see `formula` in `lm`.
formula.reduced	The formula of the reduced model with nuisance factors only. This model should be nested within the full model.
typeone	Character string indicating which type I error rate to control, either the familywise error rate ('fwer') or false discovery rate ('fdr'). Further arguments to the FWER or FDR envelope can be passed in argument `GET.args`. If 'fwer', the type of the envelope can be chosen by specifying the argument `type` in `GET.args`.

curve_sets	A named list of sets of curves giving the dependent variable (Y), and possibly additionally factors whose values vary across the argument values of the functions. The dimensions of the elements should match with each other. Note that factors that are fixed across the functions can be given in the argument `factors`. Also `fdata` objects allowed.
factors	A data frame of factors. An alternative way to specify factors when they are constant for all argument values of the functions. The number of rows of the data frame should be equal to the number of curves. Each column should specify the values of a factor.
contrasts	Logical. FALSE and TRUE specify the two test functions as described in description part of this help file.
savefuns	Logical. If TRUE, then the functions from permutations are saved to the attribute simfuns.
lm.args	A named list of additional arguments to be passed to `lm`. See details.
GET.args	A named list of additional arguments to be passed to `global_envelope_test`.
mc.cores	The number of cores to use, i.e. at most how many child processes will be run simultaneously. Must be at least one, and parallelization requires at least two cores. On a Windows computer mc.cores must be 1 (no parallelization). For details, see `mclapply`, for which the argument is passed. Parallelization can be used in generating simulations and in calculating the second stage tests.
mc.args	A named list of additional arguments to be passed to `mclapply`. Only relevant if `mc.cores` is more than 1.
cl	Allows parallelization through the use of `parLapply` (works also in Windows), see the argument `cl` there, and examples.
fast	Logical. See details.

Details

The function `graph.flm` performs the graphical functional GLM of Mrkvička et al. (2021), described also in Section 3.6 of Myllymäki and Mrkvička (2020) (type `vignette("GET")` in R). This is a nonparametric graphical test of significance of a covariate in functional GLM. The test is able to find not only if the factor of interest is significant, but also which functional domain is responsible for the potential rejection. In the case of functional multi-way main effect ANOVA or functional main effect ANCOVA models, the test is able to find which groups differ (and where they differ). In the case of functional factorial ANOVA or functional factorial ANCOVA models, the test is able to find which combination of levels (which interactions) differ (and where they differ). The described tests are global envelope tests applied in the context of GLMs. The Freedman-Lane algorithm (Freedman and Lane, 1983) is applied to permute the functions (to obtain the simulations under the null hypothesis of "no effects"); consequently, the test approximately achieves the desired significance level.

The specification of the full and reduced formulas is important. The reduced model should be nested within the full model. The full model should include in addition to the reduced model the interesting factors whose effects are under investigation. The implementation to find the coefficients of the interesting factors is based on `dummy.coef` and the restrictions there apply.

The regression coefficients serve as test functions in the graphical functional GLM. For a continuous interesting factor, the test function is its regression coefficient across the functional domain. For a discrete factor, there are two possibilities that are controlled by the arguments `contrasts`. If `contrasts = FALSE`, then the test statistic is the function/long vector where the coefficients related to all levels of the factor are joined together. If `contrasts = TRUE`, then the differences between the same coefficients are considered instead. Given the coefficients in a specific order that is obtained through the use of `lm` and `dummy.coef`, the differences are taken for couples i and j where i < j and reducing j from i (e.g. for three groups 1,2,3, the constrasts are 1-2, 1-3, 2-3).

There are different versions of the implementation depending on the application. Given that the argument `fast` is TRUE, then

- If all the covariates are constant across the functions, i.e. they can be provided in the argument `factors`, then a linear model is fitted separately by least-squares estimation to the data at each argument value of the functions fitting a multiple linear model by `lm`. The possible extra arguments passed in `lm.args` to `lm` must be of the form that `lm` accepts for fitting a multiple linear model. In the basic case, no extra arguments are needed.
- If some of the covariates vary across the space and there are user specified extra arguments given in `lm.args`, then the implementation fits a linear model at each argument value of the functions using `lm`, which can be rather slow. The arguments `lm.args` are passed to `lm` for fitting each linear model.

By setting `fast = FALSE`, it is possible to use the slow version for any case. Usually this is not desired.

Value

A `global_envelope` or `combined_global_envelope` object, which can be printed and plotted directly.

References

Mrkvička, T., Roskovec, T. and Rost, M. (2021) A nonparametric graphical tests of significance in functional GLM. Methodology and Computing in Applied Probability 23, 593-612. doi: 10.1007/s11009-019-09756-y

Myllymäki, M and Mrkvička, T. (2020). GET: Global envelopes in R. arXiv:1911.06583 [stat.ME]

Freedman, D., & Lane, D. (1983) A nonstochastic interpretation of reported significance levels. Journal of Business & Economic Statistics, 1(4), 292-298. doi:10.2307/1391660

Examples

```
data("rimov")
res <- graph.flm(nsim=19, # Increase the number of simulations for serious analysis!
  formula.full = Y~Year,
  formula.reduced = Y~1,
  curve_sets = list(Y=rimov), factors = data.frame(Year = 1979:2014))
plot(res)

# Test if there is a change in the slope in 1994,
# i.e. the full model is T = a + b*year + c*year:group,
res <- graph.flm(nsim = 19, # Increase the number of simulations for serious analysis!
  formula.full = Y ~ Year + Year:Group,
  formula.reduced = Y ~ Year,
```

```
  curve_sets = list(Y=rimov),
  factors = data.frame(Year = 1979:2014,
                       Group = factor(c(rep(1,times=24), rep(2,times=12)),
                                      levels=1:2)),
  contrasts = FALSE)
plot(res)

# An example of testing the joint effect of a discrete and a continuous variable

nsim <- 999
data("GDPtax")
factors.df <- data.frame(Group = GDPtax$Group, Tax = GDPtax$Profittax)
res.tax_within_group <- graph.flm(nsim = nsim,
  formula.full = Y~Group+Tax+Group:Tax,
  formula.reduced = Y~Group+Tax,
  curve_sets = list(Y=GDPtax$GDP),
  factors = factors.df)
plot(res.tax_within_group)

# Image data examples

data("abide_9002_23")
iset <- abide_9002_23$curve_set


# Testing the discrete factor 'group' with contrasts
# (Use contrasts = FALSE for 'means'; and for continuous factors)
res <- graph.flm(nsim = 19, # Increase nsim for serious analysis!
  formula.full = Y ~ Group + Sex + Age,
  formula.reduced = Y ~ Sex + Age,
  curve_sets = list(Y = iset),
  factors = abide_9002_23[['factors']],
  contrasts = TRUE,
  GET.args = list(type = "area"))
plot(res)
```

`imageset3` *A simulated set of images*

Description

A simulated set of images with a categorical factor (with three levels)

Usage

```
data("imageset3")
```

Format

A list of the `image_set` containing the simulated images, and the discrete group factor in the list component `Group`.

Details

We considered a categorical factor `Group` obtaining the values 0, 1 or 2 according to the group to which the image belongs to (10 images in each of the three groups). The images were simulated in the square window [-1,1]^2 from the general linear model (GLM)

$$Y(r) = \exp(-10 \cdot ||r||) \cdot (1 + \mathbf{1}(g = 2)) + \epsilon(r),$$

where ||r|| denotes the Euclidean distance of the pixel to the origin, g is the group and the error stems from an inhomogeneous distribution over I with the normal and bimodal errors in the middle and periphery of the image:

$$\epsilon(r) = \mathbf{1}(\|r\| \leq 0.5)G(r) + \mathbf{1}(\|r\| > 0.5)\frac{1}{2}G(r)^{1/5},$$

where G(r) is a Gaussian random field with the exponential correlation structure with scale parameter 0.15 and standard deviation 0.2. Consequently, the first two groups (0,1) have the same mean, while a bigger bump appears in the third group (2) in the middle of the image.

References

Mrkvička, T., Myllymäki, M., Kuronen, M. and Narisetty, N. N. (2022) New methods for multiple testing in permutation inference for the general linear model. Statistics in Medicine 41(2), 276-297. doi: 10.1002/sim.9236

See Also

`graph.fanova`, `frank.fanova`

Examples

```
data("imageset3")
plot(imageset3$image_set, idx=c(1:5, 11:15, 21:25), ncol=5)

# Colors can be changed as follows:
plot(imageset3$image_set, idx=c(1:5, 11:15, 21:25), ncol=5) +
  ggplot2::scale_fill_gradient(low="black", high="white")
```

`is.curve_set` *Check class.*

Description

Check class.

Usage

```
is.curve_set(x)
```

Arguments

x	An object to be checked.

`partial_forder` *Functional ordering in parts*

Description

If the functional data doesn't comfortably fit in memory it is possible to compute functional ordering by splitting the domain of the data (voxels in a brain image), using `partial_forder` on each part and finally combining the results with `combine_forder`.

Usage

```
partial_forder(
  curve_set,
  measure = c("erl", "rank", "cont", "area"),
  alternative = c("two.sided", "less", "greater")
)

combine_forder(ls)
```

Arguments

curve_set	A `curve_set` object, usually a part of a larger `curve_set`.
measure	The measure to use to order the functions from the most extreme to the least extreme one. Must be one of the following: 'rank', 'erl', 'cont', 'area', 'max', 'int', 'int2'. Default is 'erl'.
alternative	A character string specifying the alternative hypothesis. Must be one of the following: "two.sided" (default), "less" or "greater". The last two options only available for types `'rank'`, `'erl'`, `'cont'` and `'area'`.
ls	List of objects returned by partial_forder

Value

See forder

See Also

forder

Examples

```
data("abide_9002_23")
res <- lapply(list(1:100, 101:200, 201:261), function(part) {
  set.seed(123) # When using partial_forder, all parts must use the same seed.
  fset <- frank.flm(nsim=99, formula.full = Y ~ Group + Sex + Age,
                    formula.reduced = Y ~ Group + Sex,
                    curve_sets = list(Y = abide_9002_23$curve_set[part,]),
                    factors = abide_9002_23$factors, savefuns = "return")
  partial_forder(fset, measure="erl")
})
combine_forder(res)
```

---

plot.combined_fboxplot *Plot method for the class 'combined_fboxplot'*

Description

Plot method for the class 'combined_fboxplot'

Usage

```
## S3 method for class 'combined_fboxplot'
plot(
  x,
  labels,
  scales = "free",
  ncol = 2 + 1 * (length(x) == 3),
  digits = 3,
  outliers = TRUE,
  ...
)
```

Arguments

x	an 'combined_fboxplot' object
labels	A character vector of suitable length. If dotplot = TRUE (for the level 2 test), then labels for the tests at x-axis. Otherwise labels for the separate plots.

scales	See facet_wrap. Use scales = "free" when the scales of the functions in the global envelope vary. scales = "fixed" is a good choice, when you want the same y-axis for all components. A sensible default based on r-values exists.
ncol	The maximum number of columns for the figures. Default 2 or 3, if the length of x equals 3. (Relates to the number of curve_sets that have been combined.)
digits	The number of digits used for printing the p-value or p-interval in the default main.
outliers	Logical. If TRUE, then the functions outside the functional boxplot are drawn.
...	Ignored.

---

plot.combined_global_envelope
Plot method for the class 'combined_global_envelope'

Description

This function provides plots for combined global envelopes.

Usage

```
## S3 method for class 'combined_global_envelope'
plot(
  x,
  labels,
  scales,
  sign.col = "red",
  ncol = 2 + 1 * (length(x) == 3),
  digits = 3,
  level = 1,
  ...
)
```

Arguments

x	An 'combined_global_envelope' object
labels	A character vector of suitable length. If dotplot = TRUE (for the level 2 test), then labels for the tests at x-axis. Otherwise labels for the separate plots.
scales	See facet_wrap. Use scales = "free" when the scales of the functions in the global envelope vary. scales = "fixed" is a good choice, when you want the same y-axis for all components. A sensible default based on r-values exists.
sign.col	The color for the observed curve when outside the global envelope (significant regions). Default to "red". Setting the color to NULL corresponds to no coloring. If the object contains several envelopes, the coloring is done for the widest one.
ncol	The maximum number of columns for the figures. Default 2 or 3, if the length of x equals 3. (Relates to the number of curve_sets that have been combined.)

digits	The number of digits used for printing the p-value or p-interval in the default main.
level	1 or 2. In the case of two-step combined tests (with several test functions), two different plots are available: 1 for plotting the combined global envelopes (default and most often wanted) or 2 for plotting the second level test result.
...	Ignored.

Details

Plotting method for the class 'combined_global_envelope', i.e. combined envelopes for 1d functions.

See Also

central_region

plot.combined_global_envelope2d

Plotting function for combined 2d global envelopes

Description

If fixedscales is FALSE (or 0) all images will have separate scale. If fixedscales is TRUE (or 1) each x[[i]] will have a common scale. If fixedscales is 2 all images will have common scale.

If more than one envelope has been calculated (corresponding to several coverage/alpha), only the largest one is plotted.

Usage

```
## S3 method for class 'combined_global_envelope2d'
plot(
  x,
  fixedscales = 2,
  labels,
  what = c("obs", "lo", "hi", "lo.sign", "hi.sign"),
  sign.col = "red",
  transparency = 155/255,
  digits = 3,
  ...
)
```

Arguments

x	A 'global_envelope' object for two-dimensional functions
fixedscales	0, 1 or 2. See details.

`labels` A character vector of suitable length giving the labels for the separate plots. Default exists. This parameter allows replacing the default.

`what` Character vector specifying what information should be plotted for 2d functions. A combination of: Observed (`"obs"`), upper envelope (`"hi"`), lower envelope (`"lo"`), observed with significantly higher values highlighted (`"hi.sign"`), observed with significantly lower values highlighted (`"lo.sign"`).

`sign.col` The color for the observed curve when outside the global envelope (significant regions). Default to "red". Setting the color to `NULL` corresponds to no coloring. If the object contains several envelopes, the coloring is done for the widest one.

`transparency` A number between 0 and 1 (default 155/255, 60 Similar to alpha of `rgb`. Used in plotting the significant regions for 2d functions.

`digits` The number of digits used for printing the p-value or p-interval in the default main.

`...` Ignored.

Examples

```
data("abide_9002_23")
iset <- subset(abide_9002_23[['curve_set']], 1:50)
factors <- abide_9002_23[['factors']][1:50,]

res <- graph.flm(nsim = 19, # Increase nsim for serious analysis!
  formula.full = Y ~ Group + Sex + Age,
  formula.reduced = Y ~ Sex + Age,
  curve_sets = list(Y=iset), factors = factors,
  contrasts = FALSE, GET.args = list(type="area"))
plot(res)
plot(res, what=c("obs", "hi"))

plot(res, what=c("hi", "lo"), fixedscales=1)

plot(res, what=c("obs", "lo", "hi"), fixedscales=FALSE)

if(requireNamespace("gridExtra", quietly=TRUE)) {
  # Edit style of "fixedscales = 2" plots
  plot(res, what=c("obs", "hi")) + ggplot2::theme_minimal()
  plot(res, what=c("obs", "hi")) + ggplot2::theme_bw()

  # Edit style (e.g. theme) of "fixedscales = 1 or 0" plots
  gs <- lapply(res, function(x, what) { plot(x, what=what) +
     ggplot2::ggtitle("") }, what=c("obs", "hi"))
  gridExtra::grid.arrange(grobs=gs, ncol=1, top="My main")

  gs <- outer(res, c("obs", "hi"), FUN=Vectorize(function(res, what)
    list(plot(res, what=what) + ggplot2::ggtitle("") +
      ggplot2::theme(axis.ticks=ggplot2::element_blank(),
      axis.text=ggplot2::element_blank(), axis.title=ggplot2::element_blank()))))
  gridExtra::grid.arrange(grobs=t(gs))
}
```

`plot.curve_set` *Plot method for the class 'curve_set'*

Description

Plot method for the class 'curve_set'

Usage

```
## S3 method for class 'curve_set'
plot(x, idx, col_idx, idx_name = "", col = "grey70", ...)
```

Arguments

`x`	An `curve_set` object.
`idx`	Indices of functions to highlight with color `col_idx`. Default to the observed function, if there is just one. The legend of curves' colours is shown if indices are given or `x` contains one observed function. See examples to remove the legend if desired.
`col_idx`	A color for the curves to highlight, or a vector of the same length as `idx` containing the colors for the highlighted functions. Default exists.
`idx_name`	A variable name to be printed with the highlighted functions' idx. Default to empty.
`col`	The basic color for the curves (which are not highlighted).
`...`	Ignored.

See Also

`create_curve_set`

Examples

```
cset <- create_curve_set(list(r = 1:10, obs = matrix(runif(10*5), ncol=5)))
plot(cset)
# Highlight some functions
plot(cset, idx=c(1,3))
plot(cset, idx=c(1,3), col_idx=c("black", "red"))
# Change legend
plot(cset, idx=c(1,3), col_idx=c("black", "red"), idx_name="Special functions")
plot(cset, idx=c(1,3)) + ggplot2::theme(legend.position="bottom")
# Add labels
plot(cset, idx=c(1,3)) + ggplot2::labs(x="x", y="Value")
# and title
plot(cset) + ggplot2::labs(title="Example curves", x="x", y="Value")
# A curve_set with one observed function (other simulated)
if(requireNamespace("mvtnorm", quietly=TRUE)) {
  cset <- create_curve_set(list(obs = c(-1.6, 1.6),
```

```
            sim_m = t(mvtnorm::rmvnorm(200, c(0,0), matrix(c(1,0.5,0.5,1), 2, 2)))))
  plot(cset)
  # Remove legend
  plot(cset) + ggplot2::theme(legend.position="none")
}
```

`plot.curve_set2d` *Plot method for the class 'curve_set2d'*

Description

Plot method for the class 'curve_set2d', i.e. two-dimensional functions

Usage

```
## S3 method for class 'curve_set2d'
plot(x, idx = 1, ncol = 2 + 1 * (length(idx) == 3), ...)
```

Arguments

`x`	An `curve_set2d` object
`idx`	Indices of 2d functions to plot.
`ncol`	The maximum number of columns for the figures. Default 2 or 3, if the length of x equals 3. (Relates to the number of curve_sets that have been combined.)
`...`	Ignored.

Examples

```
data("abide_9002_23")
plot(abide_9002_23$curve_set, idx=c(1, 27))
```

`plot.fboxplot` *Plot method for the class 'fboxplot'*

Description

Plot method for the class 'fboxplot'

Usage

```
## S3 method for class 'fboxplot'
plot(x, digits = 3, outliers = TRUE, ...)
```

Arguments

x	an 'fboxplot' object
digits	The number of digits used for printing the p-value or p-interval in the default main.
outliers	Logical. If TRUE, then the functions outside the functional boxplot are drawn.
...	Ignored.

Examples

```
if(requireNamespace("fda", quietly=TRUE)) {
  years <- paste(1:18)
  curves <- fda::growth[['hgtf']][years,]
  # Heights
  cset1 <- create_curve_set(list(r = as.numeric(years),
                                 obs = curves))
  bp <- fBoxplot(cset1, coverage=0.50, type="area", factor=1)
  plot(bp)
  plot(bp) + ggplot2::theme(legend.position="bottom")
  plot(bp) + ggplot2::theme(legend.position="none")
  plot(bp, plot_outliers=FALSE)
}
```

---

plot.fclust *Plot method for the class 'fclust'*

Description

Plot method for the 'fclust' objects returned by fclustering.

Usage

```
## S3 method for class 'fclust'
plot(x, plotstyle = c("marginal", "joined"), coverage = 0.5, nstep, ncol, ...)
```

Arguments

x	An 'fclust' object.
plotstyle	The resulting central regions of clusters can be plotted by sorting the appropriate curve_set only 'marginal' or by sorting the joined list of curve_set objects 'joined'. If 'joined' is used the shown central regions corresponds to the joined ordering used to cluster the functional data. If 'marginal' is used the shown central regions do not correspond to the joined ordering used to cluster the functional data, but better express the shape of cluster with respect to given curve_set.
coverage	The coverage of central regions to be used to show the clusters.

`nstep` 1 or 2 for how to contruct a combined (joined) global envelope if there are more than one sets of curves. Default to 1, if the numbers of points where the curves are observed (r) are the same in each set, and 2 otherwise.

`ncol` The number of columns in the graphical output, when there is just one set of curves that has been ordered. If not given, `c(1, k+1)` is used, which gives all plots in one row. For more sets of curves, the rows are fixed to correspond to the sets (one row for each set).

`...` Ignored.

Details

The clusters are shown respectively for each `curve_set`. Thus for each `curve_set` the panel with all the medoids is shown followed by all clusters represented by central region, medoid and all curves belonging to it.

For all sources, the function plots the deepest curves for all clusters and the deepest curve of each cluster together with the desired central region and all the curves of the group.

References

Dai, W., Athanasiadis, S., Mrkvička, T. (2021) A new functional clustering method with combined dissimilarity sources and graphical interpretation. Intech open, London, UK. DOI: 10.5772/intechopen.100124

`plot.global_envelope` *Plot method for the class 'global_envelope'*

Description

Plot method for the class 'global_envelope'

Usage

```
## S3 method for class 'global_envelope'
plot(
  x,
  dotplot = length(x$r) < 10,
  sign.col = "red",
  labels = NULL,
  digits = 3,
  ...
)
```

Arguments

x	An 'global_envelope' object
dotplot	Logical. If TRUE, then instead of envelopes a dot plot is done. Suitable for low dimensional test vectors. Default: TRUE if the dimension is less than 10, FALSE otherwise.
sign.col	The color for the observed curve when outside the global envelope (significant regions). Default to "red". Setting the color to NULL corresponds to no coloring. If the object contains several envelopes, the coloring is done for the widest one.
labels	A character vector of suitable length. If dotplot = TRUE, then labels for the tests at x-axis.
digits	The number of digits used for printing the p-value or p-interval in the default main.
...	Ignored.

See Also

central_region, global_envelope_test

Examples

```
if(require("spatstat.explore", quietly=TRUE)) {
  X <- unmark(spruces)
  nsim <- 1999 # Number of simulations

  env <- envelope(X, fun="Kest", nsim=nsim,
                  savefuns=TRUE, # save the functions
                  correction="translate", # edge correction for K
                  simulate=expression(runifpoint(ex=X))) # Simulate CSR
  res <- global_envelope_test(env, type="erl")

  # Default plot
  plot(res)
  # Plots can be edited, e.g.
  # Remove legend
  plot(res) + ggplot2::theme(legend.position="none")
  # Change its position
  plot(res) + ggplot2::theme(legend.position="right")
  # Change the outside color
  plot(res, sign.col="#5DC863FF")
  plot(res, sign.col=NULL)
  # Change default title and x- and y-labels
  plot(res) + ggplot2::labs(title="95% global envelope", x="x", y="f(x)")

  # Prior to the plot, you can set your preferred ggplot theme by theme_set
  old <- ggplot2::theme_set(ggplot2::theme_bw())
  plot(res)

  # Do other edits, e.g. turn off expansion with the default limits
  plot(res) + ggplot2::coord_cartesian(expand=FALSE)
```

```
  # Go back to the old theme
  ggplot2::theme_set(old)

  # If you are working with the R package spatstat and its envelope-function,
  # you can obtain global envelope plots in the style of spatstat using plot.fv:
  plot.fv(res)
}
```

---

plot.global_envelope2d

Plotting function for 2d global envelopes

Description

If more than one envelope has been calculated (corresponding to several coverage/alpha), only the largest one is plotted.

Usage

```
## S3 method for class 'global_envelope2d'
plot(
  x,
  fixedscales = TRUE,
  what = c("obs", "lo", "hi", "lo.sign", "hi.sign"),
  sign.col = "red",
  transparency = 155/255,
  digits = 3,
  ...
)
```

Arguments

x	A 'global_envelope' object for two-dimensional functions
fixedscales	Logical. TRUE for the same scales for all images.
what	Character vector specifying what information should be plotted for 2d functions. A combination of: Observed ("obs"), upper envelope ("hi"), lower envelope ("lo"), observed with significantly higher values highlighted ("hi.sign"), observed with significantly lower values highlighted ("lo.sign").
sign.col	The color for the observed curve when outside the global envelope (significant regions). Default to "red". Setting the color to NULL corresponds to no coloring. If the object contains several envelopes, the coloring is done for the widest one.
transparency	A number between 0 and 1 (default 155/255, 60 Similar to alpha of rgb. Used in plotting the significant regions for 2d functions.
digits	The number of digits used for printing the p-value or p-interval in the default main.
...	Ignored.

popgrowthmillion *Population growth*

Description

Population growth

Usage

```
data("popgrowthmillion")
```

Format

A matrix, where each row corresponds to a year and each column to a country. Column names correspond to the countries, and row names to the years.

Details

This dataset includes population growth, i.e. population at the end of the year divided by population at the beginning of the year, in 134 countries in years from 1950 to 2015. The dataset includes only countries over million inhabitants in 1950. The data were extracted from the supplement of Nagy et al. (2017) distributed under the GPL-2 license.

References

Nagy, S., I. Gijbels, and D. Hlubinka (2017). Depth-based recognition of shape outlying functions. Journal of Computational and Graphical Statistics 26 (4), 883-893.

print.combined_fboxplot *Print method for the class 'combined_fboxplot'*

Description

Print method for the class 'combined_fboxplot'

Usage

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

Arguments

x	an 'combined_fboxplot' object
...	Ignored.

print.combined_global_envelope

Print method for the class 'combined_global_envelope'

Description

Print method for the class 'combined_global_envelope'

Usage

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

Arguments

x	A 'combined_global_envelope' object
...	Ignored.

print.curve_set

Print method for the class 'curve_set'

Description

Print method for the class 'curve_set'

Usage

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

Arguments

x	an 'curve_set' object
...	Passed to str.

print.deviation_test *Print method for the class 'deviation_test'*

Description

Print method for the class 'deviation_test'

Usage

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

Arguments

x	an 'deviation_test' object
...	Ignored.

print.fboxplot *Print method for the class 'fboxplot'*

Description

Print method for the class 'fboxplot'

Usage

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

Arguments

x	an 'fboxplot' object
...	Ignored.

`print.fclust` *Print method for the class 'fclust'*

Description

Print method for the 'fclust' objects returned by `fclustering`.

Usage

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

Arguments

`x`	A object of class 'fclust'.
`...`	Ignored.

`print.fdr_envelope` *Print method for the class 'fdr_envelope'*

Description

Print method for the class 'fdr_envelope'

Usage

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

Arguments

`x`	An 'fdr_envelope' object
`...`	Ignored.

print.GET_contingency *Print method for the class 'GET_contingency'*

Description

Print method for the class 'GET_contingency'

Usage

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

Arguments

`x`	A 'GET_contingency' object
`...`	Ignored.

print.global_envelope *Print method for the class 'global_envelope'*

Description

Print method for the class 'global_envelope'

Usage

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

Arguments

`x`	A 'global_envelope' object.
`...`	Ignored.

qdir_envelope *Global scaled maximum absolute difference (MAD) envelope tests*

Description

Performs the global scaled MAD envelope tests, either directional quantile or studentised, or the unscaled MAD envelope test. These tests correspond to calling the function `global_envelope_test` with `type="qdir"`, `type = "st"` and `type="unscaled"`, respectively. The functions `qdir_envelope`, `st_envelope` and `unscaled_envelope` have been kept for historical reasons; preferably use `global_envelope_test` with the suitable `type` argument.

Usage

```
qdir_envelope(curve_set, ...)

st_envelope(curve_set, ...)

unscaled_envelope(curve_set, ...)
```

Arguments

`curve_set`	A curve_set (see `create_curve_set`) or an `envelope` object of **spatstat**. If an envelope object is given, it must contain the summary functions from the simulated patterns which can be achieved by setting savefuns = TRUE when calling the function of **spatstat**.
`...`	Additional parameters to be passed to `global_envelope_test`.

Details

The directional quantile envelope test (Myllymäki et al., 2015, 2017) takes into account the unequal variances of the test function T(r) for different distances r and is also protected against asymmetry of T(r).

The studentised envelope test (Myllymäki et al., 2015, 2017) takes into account the unequal variances of the test function T(r) for different distances r.

The unscaled envelope test (Ripley, 1981) corresponds to the classical maximum deviation test without scaling, and leads to envelopes with constant width over the distances r. Thus, it suffers from unequal variance of T(r) over the distances r and from the asymmetry of distribution of T(r). We recommend to use the other global envelope tests available, see `global_envelope_test` for full list of alternatives.

Value

An object of class `global_envelope` of `combined_global_envelope` which can be printed and plotted directly. See `global_envelope_test` for more details.

References

Myllymäki, M., Grabarnik, P., Seijo, H. and Stoyan. D. (2015). Deviation test construction and power comparison for marked spatial point patterns. Spatial Statistics 11: 19-34. doi: 10.1016/j.spasta.2014.11.004

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017). Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 79: 381–404. doi: 10.1111/rssb.12172

Ripley, B.D. (1981). Spatial statistics. Wiley, New Jersey.

See Also

global_envelope_test

Examples

```
# See more examples in ?global_envelope_test
## Testing complete spatial randomness (CSR)
#-------------------------------------------
if(require("spatstat.explore", quietly=TRUE)) {
  X <- spruces
  nsim <- 999 # Number of simulations

  ## Test for complete spatial randomness (CSR)
 # Generate nsim simulations under CSR, calculate centred L-function for the data and simulations
  env <- envelope(X, fun="Lest", nsim=nsim, savefuns=TRUE,
                  correction="translate", transform=expression(.-r),
                  simulate=expression(runifpoint(ex=X)))
  res_qdir <- qdir_envelope(env) # The directional quantile envelope test
  plot(res_qdir)

  ## Advanced use:
  # Create a curve set, choosing the interval of distances [r_min, r_max]
  curve_set <- crop_curves(env, r_min=1, r_max=8)
  # The directional quantile envelope test
  res_qdir <- qdir_envelope(curve_set); plot(res_qdir)
  # The studentised envelope test
  res_st <- st_envelope(curve_set); plot(res_st)
  # The unscaled envelope test
  res_unscaled <- unscaled_envelope(curve_set); plot(res_unscaled)
}
```

---

rank_envelope *The rank envelope test*

Description

The rank envelope test, p-values and global envelopes. The test corresponds to the global envelope test that can be carriet out by global_envelope_test by specifying the type for which the options "rank", "erl", "cont" and "area" are available. The last three are modifications of the first one to treat the ties in the extreme rank ordering used in "rank". This function is kept for historical reasons.

Usage

```
rank_envelope(curve_set, type = "rank", ...)
```

Arguments

curve_set	A curve_set (see create_curve_set) or an envelope object of **spatstat**. If an envelope object is given, it must contain the summary functions from the simulated patterns which can be achieved by setting savefuns = TRUE when calling the function of **spatstat**.
type	The type of the global envelope with current options for "rank", "erl", "cont" and "area". If "rank", the global rank envelope accompanied by the p-interval is given (Myllymäki et al., 2017). If "erl", the global rank envelope based on extreme rank lengths accompanied by the extreme rank length p-value is given (Myllymäki et al., 2017, Mrkvička et al., 2018). See details and additional sections thereafter.
...	Additional parameters to be passed to global_envelope_test.

Details

The "rank" envelope test is a completely non-parametric test, which provides the 100(1-alpha)% global envelope for the chosen test function T(r) on the chosen interval of distances and associated p-values. The other three types are solutions to break the ties in the extreme ranks on which the "rank" envelope test is based on.

Note: The method to break ties for the global type = "rank" envelope (Myllymäki et al., 2017) can be done by the argument ties with default to ties = "erl" corresponding to the extreme rank length breaking of ties. In this case the global envelope corresponds to the extreme rank measure. If instead choosing type to be "erl", "cont" or "area", then the global envelope corresponds to these measures.

Value

An object of class global_envelope of combined_global_envelope which can be printed and plotted directly. See global_envelope_test for more details.

Number of simulations

The global "erl", "cont", "area" envelope tests allow in principle a lower number of simulations to be used than the global "rank" test based on extreme ranks. However, if feasible, we recommend some thousands of simulations in any case to achieve a good power and repeatability of the test. For the global "rank" envelope test, Myllymäki et al. (2017) recommended to use at least 2500 simulations for testing at the significance level alpha = 0.05 for single function tests, experimented with summary functions for point processes.

References

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017). Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 79: 381–404. doi: 10.1111/rssb.12172

Mrkvička, T., Myllymäki, M. and Hahn, U. (2017). Multiple Monte Carlo testing, with applications in spatial point processes. Statistics & Computing 27 (5): 1239-1255. doi: 10.1007/s11222-016-9683-9

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2020) A one-way ANOVA test for functional data with graphical interpretation. Kybernetika 56 (3), 432-458. doi: 10.14736/kyb-2020-3-0432

See Also

global_envelope_test

Examples

```
# See ?global_envelope_test for more examples

## Testing complete spatial randomness (CSR)
#-------------------------------------------
if(require("spatstat.explore", quietly=TRUE)) {
  X <- unmark(spruces)
  nsim <- 2499 # Number of simulations

 # Generate nsim simulations under CSR, calculate centred L-function for the data and simulations
  env <- envelope(X, fun="Lest", nsim=nsim, savefuns=TRUE,
                  correction="translate", transform=expression(.-r),
                  simulate=expression(runifpoint(ex=X)))
  # The rank envelope test
  res <- rank_envelope(env)
  # Plot the result.
  plot(res)

  ## Advanced use:
 # Choose the interval of distances [r_min, r_max] (at the same time create a curve_set from 'env')
  curve_set <- crop_curves(env, r_min=1, r_max=7)
  # Do the rank envelope test
  res <- rank_envelope(curve_set); plot(res)
}
```

---

residual — *Residual form of the functions*

Description

Subtract the theoretical function S_H_0 or the mean of the functions in the curve set. If the curve_set object contains already residuals T_i(r) - T_0(r), use_theo ignored and the same object returned.

Usage

```
residual(curve_set, use_theo = TRUE)
```

Arguments

`curve_set`	A curve_set (see `create_curve_set`) or an `envelope` object of **spatstat**. If an envelope object is given, it must contain the summary functions from the simulated patterns which can be achieved by setting savefuns = TRUE when calling the `envelope` function.
`use_theo`	Whether to use the theoretical summary function or the mean of the functions in the curve_set.

Details

The mean of the functions in the `curve_set` is the mean of all functions. If `use_theo = TRUE`, but the component `theo` does not exist in the `curve_set`, the mean of the functions is used silently.

Value

A curve set object containing residual summary functions. theo is no longer included.

`rimov` *Year temperature curves*

Description

Year temperature curves

Usage

```
data("rimov")
```

Format

A `curve_set` object with water temperatures in 365 days of the 36 years. The component `curve_set[['r']]` is a vector of days (from 1 to 365), whereas `curve_set[['obs']]` contains the water temperatures such that each column gives year temperatures in a year.

Details

The water temperature data sampled at the water level of Rimov reservoir in Czech republic every day for the 36 years between 1979 and 2014.

References

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2020) A one-way ANOVA test for functional data with graphical interpretation. Kybernetika 56 (3), 432-458. doi: 10.14736/kyb-2020-3-0432

See Also

graph.fanova

Examples

```
data("rimov")
groups <- factor(c(rep(1, times=12), rep(2, times=12), rep(3, times=12)))
for(i in 1:3)
  assign(paste0("p", i), plot(subset(rimov, groups==i)) +
    ggplot2::labs(title=paste("Group ", i, sep=""), y="Temperature"))
p3
if(require("patchwork", quietly=TRUE))
  p1 + p2 + p3
# See example analysis in ?graph.fanova
```

`saplings` *Saplings data set*

Description

Saplings data set

Usage

```
data("saplings")
```

Format

A `data.frame` containing the locations (x- and y-coordinates) of 123 trees in an area of 75 m x 75 m.

Details

A pattern of small trees (height <= 15 m) originating from an uneven aged multi-species broadleaf nonmanaged forest in Kaluzhskie Zaseki, Russia.

The pattern is a sample part of data collected over 10 ha plot as a part of a research program headed by project leader Prof. O.V. Smirnova.

References

Grabarnik, P. and Chiu, S. N. (2002) Goodness-of-fit test for complete spatial randomness against mixtures of regular and clustered spatial point processes. *Biometrika*, **89**, 411–421.

van Lieshout, M.-C. (2010) Spatial point process theory. In Handbook of Spatial Statistics (eds. A. E. Gelfand, P. J. Diggle, M. Fuentes and P. Guttorp), Handbooks of Modern Statistical Methods. Boca Raton: CRC Press.

Myllymäki, M., Mrkvička, T., Grabarnik, P., Seijo, H. and Hahn, U. (2017). Global envelope tests for spatial point patterns. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 79: 381-404. doi: 10.1111/rssb.12172

See Also

adult_trees

Examples

```
# This is an example analysis of the saplings data set
#=====================================================
# Example of Myllymaki et al. (2017, Supplement S4).
if(require("spatstat.explore", quietly=TRUE)) {
  data("saplings")
  saplings <- as.ppp(saplings, W=square(75))

  # First choose the r-distances for L (r) and J (rJ) functions, respectively.
  nr <- 500
  rmin <- 0.3; rminJ <- 0.3
  rmax <- 10; rmaxJ <- 6
  rstep <- (rmax-rmin)/nr; rstepJ <- (rmaxJ-rminJ)/nr
  r <- seq(0, rmax, by=rstep)
  rJ <- seq(0, rmaxJ, by=rstepJ)

  #-- CSR test --# (a simple hypothesis)
  #--------------#
  # First, a CSR test using the L(r)-r function:
  # Note: CSR is simulated by fixing the number of points and generating nsim simulations
  # from the binomial process, i.e. we deal with a simple hypothesis.
  nsim <- 999 # Number of simulations

  env <- envelope(saplings, nsim=nsim,
   simulate=expression(runifpoint(ex=saplings)), # Simulate CSR
  fun="Lest", correction="translate", # T(r) = estimator of L with translational edge correction
   transform=expression(.-r),          # Take the L(r)-r function instead of L(r)
   r=r,                                # Specify the distance vector
   savefuns=TRUE)                      # Save the estimated functions
  # Crop the curves to the interval of distances [rmin, rmax]
  # (at the same time create a curve_set from 'env')
  curve_set <- crop_curves(env, r_min=rmin, r_max=rmax)
  # Perform a global envelope test
 res <- global_envelope_test(curve_set, type="erl") # type="rank" and larger nsim was used in S4.
  # Plot the result.
  plot(res) + ggplot2::ylab(expression(italic(hat(L)(r)-r)))

  # -> The CSR hypothesis is clearly rejected and the rank envelope indicates clear
  # clustering of saplings. Next we explore the Matern cluster process as a null model.
}

if(require("spatstat.model", quietly=TRUE)) {
  #-- Testing the Matern cluster process --# (a composite hypothesis)
  #----------------------------------------#
 # Fit the Matern cluster process to the pattern (using minimum contrast estimation with the pair
  # correction function)
  fitted_model <- kppm(saplings~1, clusters="MatClust", statistic="pcf")
  summary(fitted_model)
```

```
  nsim <- 19 # 19 just for experimenting with the code!!
  #nsim <- 499 # 499 is ok for type = 'qdir' (takes > 1 h)

  # Make the adjusted directional quantile global envelope test using the L(r)-r function
  # (For the rank envelope test, choose type = "rank" instead and increase nsim.)
  adjenvL <- GET.composite(X=fitted_model,
                    fun="Lest", correction="translate",
                    transform=expression(.-r), r=r,
                    type="qdir", nsim=nsim, nsimsub=nsim,
                    r_min=rmin, r_max=rmax)
  # Plot the test result
  plot(adjenvL) + ggplot2::ylab(expression(italic(L(r)-r)))

  # From the test with the L(r)-r function, it appears that the Matern cluster model would be
  # a reasonable model for the saplings pattern.
  # To further explore the goodness-of-fit of the Matern cluster process, test the
  # model with the J function:
  # This takes quite some time if nsim is reasonably large.
  adjenvJ <- GET.composite(X=fitted_model,
                    fun="Jest", correction="none", r=rJ,
                    type="qdir", nsim=nsim, nsimsub=nsim,
                    r_min=rminJ, r_max=rmaxJ)
  # Plot the test result
  plot(adjenvJ) + ggplot2::ylab(expression(italic(J(r))))
  # -> the Matern cluster process not adequate for the saplings data

  # Test with the two test functions jointly
  adjenvLJ <- GET.composite(X=fitted_model,
                    testfuns=list(L=list(fun="Lest", correction="translate",
                                     transform=expression(.-r), r=r),
                                  J=list(fun="Jest", correction="none", r=rJ)),
                    type="erl", nsim=nsim, nsimsub=nsim,
                    r_min=c(rmin, rminJ), r_max=c(rmax, rmaxJ),
                    save.cons.envelope=TRUE)
  plot(adjenvLJ)
}
```

subset.curve_set *Subsetting curve sets*

Description

Return subsets of curve sets which meet conditions.

Usage

```
## S3 method for class 'curve_set'
subset(x, subset, ...)
```

Arguments

`x`	A `curve_set` object.
`subset`	A logical expression indicating curves to keep.
`...`	Ignored.

Examples

```
if(require("fda.usc", quietly=TRUE)) {
  # Prepare data
  data("poblenou")
  Free <- poblenou$df$day.festive == 1 |
    as.integer(poblenou$df$day.week) >= 6
  MonThu <- poblenou$df$day.festive == 0 & poblenou$df$day.week %in% 1:4
  Friday <- poblenou$df$day.festive == 0 & poblenou$df$day.week == 5

  # Data as a curve_set
  cset <- create_curve_set(list(r=0:23,
             obs=t(poblenou[['nox']][['data']])))
  plot(subset(cset, MonThu))
  plot(subset(cset, Friday))
  plot(subset(cset, Free))
}
```

Index