

Package ‘GET’

July 12, 2019

Version 0.1-3

Date 2019-06-28

Title Global Envelopes

Encoding UTF-8

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

Imports caTools, fda.usc, ggplot2, graphics, grDevices, gridExtra,
gstat, methods, parallel, plyr, sp, spatstat, stats, utils,

Suggests testthat, mvtnorm, fda

Description Implementation of global envelopes with intrinsic graphical interpretation which 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 et al. (2017) <doi: 10.1111/rssb.12172>, Mrkvička et al. (2017) <doi: 10.1007/s11222-016-9683-9>, Mrkvička et al. (2016) <doi: 10.1016/j.jspasta.2016.04.005>, Mrkvička et al. (2018) <arXiv:1612.03608>, Mrkvička et al.

License GPL-3

RoxygenNote 6.1.1

NeedsCompilation no

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

Repository CRAN

Date/Publication 2019-07-12 16:10:08 UTC

R topics documented:

GET-package	3
adult_trees	7
central_region	8
central_region2d	12
cgec	13
combined_scaled_MAD_envelope	14
create_curve_set	16
create_image_set	17
crop_curves	18
deviation_test	18
envelope_to_curve_set	20
fallen_trees	21
fBoxplot	22
forder	22
frank.fanova	25
frank.fanova2d	26
frank.fglm	27
frank.fglm2d	29
GDptax	30
GET.composite	31
GET.necdf	37
GET.variogram	38
global_envelope_test	39
global_envelope_test2d	47
graph.fanova	49
graph.fanova2d	52
graph.fglm	53
graph.fglm2d	55
imageset1	57
imageset2	58
imageset3	59
is.curve_set	60
plot.combined_fboxplot	60
plot.combined_global_envelope	61
plot.combined_global_envelope2d	62
plot.curve_set	63
plot.fboxplot	63
plot.global_envelope	64
plot.global_envelope2d	65
plot.image_set	66
print.combined_global_envelope	67
print.combined_global_envelope2d	67
print.curve_set	68
print.deviation_test	68
print.global_envelope	69
print.global_envelope2d	69

print.image_set	70
qdir_envelope	70
rank_envelope	72
residual	74
rimov	75
saplings	76

Index	79
--------------	-----------

GET-package	<i>GET: Global Envelopes in R</i>
-------------	-----------------------------------

Description

The **GET** package provides global envelopes which 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).

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 library **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 models/functions.

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))),
```
 - 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 `envelope` 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,
```

- 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`

Also the adjusted tests can be based on several test functions.

- *One-way functional ANOVA:*

- Graphical functional ANOVA tests: `graph.fanova`
- Global rank envelope based on F-values: `frank.fanova`
- Image (2d function) counterparts: `graph.fanova2d`, `frank.fanova2d`

- *Functional general linear model (GLM):*

- Graphical functional GLM: `graph.fglm`
- Global rank envelope based on F-values: `frank.fglm`
- Image (2d function) counterparts: `graph.fglm2d`, `frank.fglm2d`

- Wrapper functions to perform global envelopes for specific purposes:

- Central regions for images (2d functions): `central_region2d`
- Global envelope tests for images (2d functions): `global_envelope_test2d`
- Graphical n sample test of correspondence of distribution functions: `GET.necdf`
- 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` (or `image_set`) object. Use `create_curve_set` (or `create_image_set`) to create a `curve_set` (or `image_set`) object from the functions $T_i(r)$, $i=1,\dots,s+1$. 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 ($T_2(r)$, ..., $T_{nsim+1}(r)$) in one way or another. Given the set of the functions $T_i(r)$, $i=1, \dots, nsim+1$, you can perform a test by [global_envelope_test](#).

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

- (Fit the model and) Create $nsim$ simulations from the (fitted) null model.
- Calculate the functions $T_1(r)$, $T_2(r)$, ..., $T_{nsim+1}(r)$.
- Use [create_curve_set](#) to create a `curve_set` object from the functions $T_i(r)$, $i=1, \dots, nsim+1$.
- Perform the test and plot the result


```
res <- global_envelope_test(curve_set) # curve_set is the 'curve_set'-object you created
plot(res)
```

2) The workflow utilizing [spatstat](#):

E.g. Say we have a point pattern, for which we would like to test a hypothesis, as a [ppp](#) object.

```
X <- spruces # an example pattern from spatstat
```

- Test complete spatial randomness (CSR):
 - Use [envelope](#) to create $nsim$ simulations under CSR and to calculate the functions you want. Important: use the option `'savefuns=TRUE'` and specify the number of simulations `nsim`. See the help documentation in [spatstat](#) for possible test functions (if `fun` not given, `Kest` is used, i.e. an estimator of the K function). Making 999 simulations of CSR and estimating K -function for each of them and data (the argument `simulate` specifies for `envelope` how to perform simulations under CSR):


```
env <- envelope(X, nsim=999, savefuns=TRUE,
```
 - Perform the test


```
res <- global_envelope_test(env)
```
 - Plot the result


```
plot(res)
```
- A goodness-of-fit of a parametric model (composite hypothesis case)
 - Fit the model to your data by means of the function [ppm](#) or [kppm](#). See the help documentation for possible models.
 - Use [GET.composite](#) to create $nsim$ simulations from the fitted model, to calculate the functions you want, and to make an adjusted global envelope test. See the example also in [saplings](#).
 - Plot the result


```
plot(res)
```

Functions for modifying sets of functions

It is possible to modify the curve set $T_1(r)$, $T_2(r)$, ..., $T_{nsim+1}(r)$ 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.

The function [envelope_to_curve_set](#) can be used to create a `curve_set` object from the object returned by [envelope](#). An `envelope` object can also directly be given to the functions mentioned above in this section.

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

- [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
- [imageset1](#): a simulated set of images (see [graph.fanova2d](#), [frank.fanova2d](#))
- [imageset2](#): a simulated set of images (see [graph.fglm2d](#), [frank.fglm2d](#))
- [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 simulations

Note that the recommended minimum number of simulations for the rank envelope test based on a single function is $n_{sim}=2499$, while for the "erl", "cont", "area", "qdir" and "st" global envelope tests and deviation tests, a lower number of simulations can be used, although the Monte Carlo error is obviously larger with a small number of simulations. For increasing 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.

Acknowledgements

Pavel Grabarnik, Ute Hahn, Mikko Kuronen, Michael Rost and Henri Seijo have made contributions and suggestions of code.

Author(s)

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

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

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., 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>

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2018) A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME] (<http://arxiv.org/abs/1612.03608>)

Mrkvička, T., Myllymäki, M. and Narisetty, N. N. (2019) New methods for multiple testing in permutation inference for the general linear model. arXiv:1906.09004 [stat.ME]

Mrkvička, T., Roskovec, T. and Rost, M. (2019) A nonparametric graphical tests of significance in functional GLM. arXiv:1902.04926 [stat.ME]

adult_trees

Adult trees data set

Description

Adult trees data set

Usage

```
data(adult_trees)
```

Format

An object of class `ppp.object` representing the point pattern of tree locations.

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.

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((1 -
  coverage)/2, 1 - (1 - coverage)/2), 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.
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.
nstep	1 or 2 for how to construct 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 forder .

Details

Given a curve_set (see [create_curve_set](#) for how to create such an object) or an [envelope](#) object, the function central_region constructs a central region, i.e. a global envelope, from the given set of functions (or vectors). There are two options for the functions that the curve_set can contain:

- If the component obs of the curve_set is a matrix, then it is assumed that all the functions are data/observed. In this case, the component sim_m of the curve_set (which can be then NULL) is ignored and the central region constructed from the functions given in obs.

- If the component `obs` is a vector, then `sim_m` should be provided as well and it is assumed to contain simulated functions (obtained, e.g., from some model or by permutation). The central region is calculated from all the functions.

Thus the `curve_set` contains functions (or vectors) $T_1(r), \dots, T_s(r)$. In the case of one observed function only, the data function is considered to be $T_1(r)$.

Generally an envelope is a band bounded by the vectors (or functions) T_{low} and T_{hi} . A $100(1-\alpha)\%$ or $100*\text{coverage}\%$ 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 [forder](#).

- `'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 constructed 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 constructed 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.

For each curve in the `curve_set`, both the data curve and the simulations, an above mention measure k is determined. The measure values k_1, k_2, \dots, k_s are returned in the attribute `'k'` (in a case of one observed curve only, `k[1]` is its value). Based on the chosen measure, the central region, i.e. the global envelope, is constructed on the chosen interval of argument values (the functions in the `curve_set` are assumed to be given on this interval only).

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 (2019). 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.

Value

Either an object of class "global_envelope" and "fv" (see [fv.object](#)), or an "combined_global_envelope" for the case that `curve_sets` is a list of objects. The objects can be printed and plotted directly.

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
- `obs` = the data function, if there is only one data function. Otherwise not existing.
- `lo` = the lower envelope based on the simulated functions
- `hi` = the upper envelope based on the simulated functions
- `central` = If the `curve_set` (or envelope object) contains a component 'theo', then this function is used as the central curve and returned in this component. Otherwise, the `central_curve` is the mean of the test functions $T_i(r)$, $i=2, \dots, s+1$. Used for visualization only.

Additionally, the return value has attributes

- `method` = The name of the envelope test used for plotting purposes ("Global envelope")
- `alternative` = The alternative specified in the function call.
- `ties` = As the argument `ties`.
- `k_alpha` = The value of `k` corresponding to the $100(1-\alpha)\%$ global envelope.
- `k` = The values of the chosen measure for all the functions. If there is only one observed function, then `k[1]` will give the value of the measure for this.
- `call` = The call of the function.

and a bunch of attributes for the "fv" object type, see [fv](#). Attributes of an object `res` can be obtained using the function `attr`, e.g. `attr(res, "k")` for the values of the ordering measure.

The "combined_global_envelope" is a list of "global_envelope" objects corresponding to the components of `curve_sets`. The second level envelope on which the envelope construction is based on is saved in the attribute "level2_ge".

References

- Mrkvička, T., Hahn, U. and Myllymäki, M. (2018). A one-way ANOVA test for functional data with graphical interpretation. [arXiv:1612.03608 \[stat.ME\]](#)
- Mrkvička, T., Myllymäki, M. and Narisetty, N. N. (2019) New methods for multiple testing in permutation inference for the general linear model. [arXiv:1906.09004 \[stat.ME\]](#)
- 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. Global envelopes in R.
 Ripley, B.D. (1981). Spatial statistics. Wiley, New Jersey.

See Also

[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, ylab="height")
  cr <- central_region(curve_set, coverage=0.50, type="erl")
  plot(cr, main="50% central region")
}

## 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=t(m)))
cr <- central_region(boot.cset, coverage=0.95, type="erl")
CB.lo <- ftheta+s0*cr$lo
CB.hi <- ftheta+s0*cr$hi

# Plotting the result
plot(d, ylab="f(x)", xaxt="n", xlab="x", main="95% central region")
axis(1, at=(0:5)*20, labels=(0:5)/5)
lines(ftheta)
lines(CB.lo, lty=2)
lines(CB.hi, lty=2)

```

central_region2d *2D central region / global envelope*

Description

Provides central regions or global envelopes or confidence bands in 2D

Usage

```
central_region2d(image_sets, ...)
```

Arguments

`image_sets` An image set, i.e. a set of 2d functions (or a list of them). See [create_image_set](#).
`...` Additional parameters to be passed to [central_region](#).

Value

An object of class "global_envelope2d" (and "list"), which can be printed and plotted directly.

Essentially a data frame containing columns

- `r` = a list of vectors of values of x- and y-coordinates at which the test was made
- `obs` = the data function (matrix), if there is only one data function. Otherwise not existing.
- `lo` = the lower envelope (matrix) based on the simulated functions
- `hi` = the upper envelope (matrix) based on the simulated functions
- `central` = the central curve, mean or median (default) of the test functions $T_i(r)$, $i=2, \dots, s+1$.
Used for visualization only.

Additionally, the return value has attributes `method`, `type`, `alternative`, `k_alpha`, `alpha`, `k`, and `call`, see more detailed description in [central_region](#).

References

- 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: 10.1016/j.spasta.2016.04.005
- 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., Hahn, U. and Myllymäki, M. (2018). A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME]

cgec

Centred government expenditure centralization ratios

Description

Centred government expenditure centralization (GEC) ratios

Usage

`data(cgec)`

Format

A `curve_set` object with components `r` and `obs` containing the years of observations and the curves, i.e. the observed values of centred GEC in those years. Each column of `obs` contains the centred GEC for the years for a particular country (seen as column names). The grouping is given in the attribute `group`.

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 and 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\(26/10/2018\)](https://ec.europa.eu/eurostat/data/database(26/10/2018)).

Mrkvička, T., Hahn, U. and Myllymäki, M. A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME] (<http://arxiv.org/abs/1612.03608>)

See Also

[graph.fanova](#)

Examples

```
data(cgec)
# Plot data in groups
subs <- function(group, ...) {
  cset <- cgec
  cset$obs <- cgec$obs[, attr(cset, "group") == group]
  plot(cset, ...)
}
for(i in 1:3) subs(i, main=paste("Group ", i, sep=""), ylab="Centred GEC")
```

combined_scaled_MAD_envelope

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 the combined test can be found in Mrkvicka et al. (2017)

Usage

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

Arguments

curve_sets	A list of objects of type 'curve_set' or envelope .
type	Either "qdir" for the direction quantile envelope test (type 'qdir' in global_envelope_test) or "st" for the studentized envelope test (type 'st' global_envelope_test).
alpha	The significance level. The 100(1-alpha)% global envelope will be calculated.
probs	A two-element vector containing the lower and upper quantiles for the envelope, in that order and on the interval [0, 1]. The default values are 0.025 and 0.975.
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.
...	Additional parameters to be passed to central_region .

References

Mrkvicka, 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

Examples

```
if(require("spatstat", 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 <- saplings

  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
  system.time( env_L <- envelope(X, nsim=nsim,
    simulate=expression(runifpoint(X$n, win=X$window)),
    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
  system.time( env_F <- envelope(X, nsim=nsim,
    simulate=simulations,
```

```

        fun="Fest", correction="Kaplan", r=rJ,
        savefuns=TRUE) )
system.time( env_G <- envelope(X, nsim=nsim,
        simulate=simulations,
        fun="Gest", correction="km", r=rJ,
        savefuns=TRUE) )
system.time( 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(curve_sets=list(curve_set_L, curve_set_F,
        curve_set_G, curve_set_J),
        type = "qdir")
plot(res, plot_style="ggplot2",
        labels=c("L(r)-r", "F(r)", "G(r)", "J(r)"),
        base_size=12)
}

```

create_curve_set *Create a curve set out of a list in the right form.*

Description

Create a curve set out of a list in the right form.

Usage

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

Arguments

curve_set A list containing elements `r`, `obs`, and optionally the elements `sim_m` and `theo`. `r` must be a vector describing the radius vector, the argument values where functions have been observed (or simulated). `obs` must be either 1) a vector containing the data function, in which case `obs` must have same length as `r`, or 2) a matrix containing the `s` data functions, in which case it is assumed that each column corresponds to a data function, and the number of rows must match the length or `r`. 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 `r`. If `obs` is a matrix, `sim_m` is ignored.

If included, theo corresponds to the theoretical function (e.g., under the null hypothesis). If present, its length must match the length of r.

... Do not use. (For internal use only.)

Value

The given list adorned with the proper class name.

create_image_set *Create an image set out of a list in the right form.*

Description

Create an image set out of a list in the right form containing the values of the 2d functions. Only 2d functions in a rectangular windows are currently supported; the values are provided in matrices (arrays).

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. r must be a list of two vectors describing the argument values where images have been observed (or simulated). r[[1]] should give the argument values for x-coordinate (first dimension of the 2d functions). r[[2]] should give the argument values for y-coordinate (second dimension of the 2d functions). 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 adorned with the proper class name.

crop_curves	<i>Crop the curves to a certain interval.</i>
-------------	---

Description

Crop the curves to a certain interval in preparation for a deviation test.

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. If an envelope object is given, it must contain the summary functions from the simulated patterns which can be achieved by setting <code>savefuncs = TRUE</code> when calling envelope .
r_min	The minimum radius to include.
r_max	The maximum radius to include.

Value

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

deviation_test	<i>Deviation test</i>
----------------	-----------------------

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.

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 <code>residual()</code> .
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'.
save devs	Logical. Should the global rank values $k_i, i=1, \dots, \text{nsim}+1$ 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, $T_2(r), \dots, T_{\text{nsim}+1}(r)$, 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, i.e. `curve_set$is_residual` is FALSE (or does not exist), 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 $T_j(r), j=2, \dots, \text{nsim}+1$.
- 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 $u_i, i=1, \dots, \text{nsim}+1$, 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 $T_i(r)$ occurring 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 , $i=1,\dots,nsim+1$ (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

Examples

```
## Testing complete spatial randomness (CSR)
#-----
if(require("spatstat", 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')
}
```

envelope_to_curve_set *Turn an [envelope](#) object into a [curve_set](#) object.*

Description

Turn an [envelope](#) object into a [curve_set](#) object.

Usage

```
envelope_to_curve_set(env, ...)
```

Arguments

env	An envelope object. The envelope() functions must have been called with savefuns = TRUE.
...	Do not use. (For internal use only.)

Value

A corresponding [curve_set](#) object.

`fallen_trees`*Fallen trees*

Description

Fallen trees

Usage`data(fallen_trees)`**Format**

A `ppp.object` object with locations and heights (=marks) of 232 trees in a window with polygonal boundary.

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)
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, ...)
```

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.
...	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)) {
  curve_set <- create_curve_set(list(r=as.numeric(row.names(fda::growth$hgtf)),
                                     obs=fda::growth$hgtf))
  plot(curve_set, ylab="height")
  bp <- fBoxplot(curve_set, coverage=0.50, type="area", factor=1)
  plot(bp)
}
```

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, r_min = NULL, r_max = NULL, measure = "er1",
       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.
r_min	The minimum radius to include.
r_max	The maximum radius to include.
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, which contains both the data curve (or function or vector) $T_1(r)$ and the simulated curves $T_2(r), \dots, T_{s+1}(r)$, 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, for details see Mrkvička et al. (2017, page 1241) or Mrkvička et al. (2018, page 6).
- '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

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

where $\mathbf{R}_j \prec \mathbf{R}_i$ 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 .

- '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).

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., Hahn, U. and Myllymäki, M. (2018). A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME]

Mrkvička, T., Myllymäki, M. and Narisetty, N. N. (2019) New methods for multiple testing in permutation inference for the general linear model. arXiv:1906.09004 [stat.ME]

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

```
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, ylab="height")
  forder(curve_set, measure = "max", scaling="qdir")
  forder(curve_set, measure = "rank")
  forder(curve_set, measure = "erl")
  forder(curve_set, measure = "area")
}
```

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, variances = "equal",
             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).
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

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

where

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

See Mrkvicka et al. (2018) 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 .

Details

The test assumes 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 `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

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

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

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

, 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. (2016).

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

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. (2018) A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME] (<http://arxiv.org/abs/1612.03608>)

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, ylab="F-statistic")
```

frank.fanova2d	<i>Rank envelope F-test for images</i>
----------------	--

Description

Rank envelope F-test for images

Usage

```
frank.fanova2d(nsim, image_set, groups, ...)
```

Arguments

<code>nsim</code>	The number of random permutations.
<code>image_set</code>	A set of images containing the data, see create_image_set .
<code>groups</code>	The original groups (a factor vector representing the assignment to groups).
<code>...</code>	Additional parameters to be passed to frank.fanova .

Details

A one-way functional ANOVA for images (2d functions), based on the rank envelope applied to F values. This function can be used to perform F-rank one-way functional ANOVA tests described in Mrkvička et al. (2018). The function transforms images to vectors (1d), calls `frank.fanova` and transform results back to images (2d).

References

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2018) A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME] (<http://arxiv.org/abs/1612.03608>)

Examples

```
data("imageset1")
res <- frank.fanova2d(nsim = 19, # Increase nsim for serious analysis!
                     image_set = imageset1$image_set,
                     groups = imageset1$Group)
plot(res)
plot(res, fixedscales=FALSE, contours=FALSE)
```

frank.fglm

F rank functional GLM

Description

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

Usage

```
frank.fglm(nsim, formula.full, formula.reduced, curve_sets,
           factors = NULL, ..., GET.args = NULL, mc.cores = 1,
           mc.args = NULL)
```

Arguments

<code>nsim</code>	The number of random permutations.
<code>formula.full</code>	The formula specifying the general linear model, see formula in lm .
<code>formula.reduced</code>	The formula of the reduced model with nuisance factors only.
<code>curve_sets</code>	A named list of sets of curves giving the dependent variable (Y), and possibly additionally all the factors. The dimensions of the elements should match with each other, i.e. the factor values should be given for each argument value and each function. If factors are given in the argument <code>factors</code> , then can also be just the curve set representing Y. Also fdata objects allowed.


```
plot(res.tax_within_group)
```

frank.fglm2d *F rank functional GLM for images*

Description

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

Usage

```
frank.fglm2d(nsim, formula.full, formula.reduced, image_sets,  
          factors = NULL, ...)
```

Arguments

<code>nsim</code>	The number of random permutations.
<code>formula.full</code>	The formula specifying the general linear model, see formula in lm .
<code>formula.reduced</code>	The formula of the reduced model with nuisance factors only.
<code>image_sets</code>	A named list of sets of images giving the dependent variable (Y), and possibly additionally all the factors. The dimensions of the elements should match with each other, i.e. the factor values should be given for each argument value and each function.
<code>factors</code>	A data frame of factors. An alternative way to specify factors when they are constant for all argument values. 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.
<code>...</code>	Additional parameters to be passed to frank.fglm .

Value

A `global_envelope2d` object, which can be printed and plotted directly.

References

Mrkvička, T., Myllymäki, M. and Narisetty, N. N. (2019) New methods for multiple testing in permutation inference for the general linear model. [arXiv:1906.09004 \[stat.ME\]](#)

See Also

[frank.fglm](#), [graph.fglm2d](#)

Examples

```

data("imageset2")
# Testing discrete factor group
res.g <- frank.fglm2d(nsim = 19, # Increase nsim for serious analysis!
  formula.full = Y ~ group + z,
  formula.reduced = Y ~ z,
  image_sets = list(Y = imageset2$image_set),
  factors = data.frame(group = imageset2$Group,
    z = imageset2$z))

plot(res.g)

# Testing continuous factor z
res.z <- frank.fglm2d(nsim = 19, # Increase nsim for serious analysis!
  formula.full = Y ~ group + z,
  formula.reduced = Y ~ group,
  image_sets = list(Y = imageset2$image_set),
  factors = data.frame(group = imageset2$Group,
    z = imageset2$z))

plot(res.z)

```

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 `Profit tax`.

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/PayingTaxes/>)) 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.fglm](#)

Examples

```
data(GDPtax)
GDPcset <- GDPtax$GDP
# Plot data in groups
subs <- function(group, ...) {
  cset <- GDPcset
  cset$obs <- GDPcset$obs[, GDPtax$Group == group]
  plot(cset, ...)
}
for(i in 1:4) subs(i, main=paste("Group ", i, sep=""), ylab="GDP")
```

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 = "er1", 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

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

<code>take_residual</code>	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.
<code>save.cons.envelope</code>	Logical flag indicating whether to save the unadjusted envelope test results.
<code>savefuns</code>	Logical flag indicating whether to save all the simulated function values. Similar to the same argument of envelope .
<code>verbose</code>	Logical flag indicating whether to print progress reports during the simulations. Similar to the same argument of envelope .
<code>MrkvickaEtal2017</code>	Logical. If TRUE, type is "st" or "qdir" and several test functions are used, then the combined scaled MAD envelope presented in Mrkvicka 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 kept for historical reasons.
<code>mc.cores</code>	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 <code>mc.cores</code> 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 including the observed function and simulations from the tested model. `X.ls` should be a list of `curve_set` or [envelope](#) 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 library [spatstat](#).
 - If `X` is a fitted (point process) model object of the R library [spatstat](#), then the simulations from this model are generated and summary functions for testing calculated by [envelope](#). 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, then `envelope` 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).

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

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

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.

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>

See Also

`global_envelope_test`, `plot.global_envelope`, `saplings`

Examples

```
if(require("spatstat", quietly=TRUE)) {
  data(saplings)

  # 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)

# Option 2: Generate the simulations "by yourself"
# Writing with for-loops (instead of envelope calls)
# Serves as an example for non-spatstat models.
#-----
# 1. Fit the model
M1 <- kppm(saplings~1, clusters = "MatClust", statistic="K")
# Generate nsim simulations by the given function using the fitted model
# 2. Simulate a sample from the fitted null model and
# compute the test vectors for data and each simulation
nsim <- nsimsub <- 19 # Number of simulations
# Observed function
obs.L <- Lest(saplings, correction = "translate", r = r)

```

```

obs <- obs.L[['trans']] - r
# Simulated functions
sim <- matrix(nrow = length(r), ncol = nsimsub)
for(i in 1:nsimsub) {
  sim.X <- simulate(M1)[[1]]
  sim[, i] <- Lest(sim.X, correction = "translate", r = r)[['trans']] - r
}
# Create a curve_set
X <- create_curve_set(list(r = r, obs = obs, sim_m = sim))
# 3. Simulate another sample from the fitted null model.
sims2 <- simulate(M1, nsim=nsim) # list of simulations
# 4. Fit the null model to each of the patterns,
# simulate a sample from the null model,
# and compute the test vectors for all.
X.ls <- list()
for(rep in 1:nsim) {
  M2 <- kppm(sims2[[rep]], clusters = "LGCP", statistic="K")
  obs2 <- Lest(sims2[[rep]], correction = "translate", r = r)[['trans']] - r
  sim2 <- matrix(nrow = length(r), ncol = nsimsub)
  for(i in 1:nsimsub) {
    sim2.X <- simulate(M2)[[1]]
    sim2[, i] <- Lest(sim2.X, correction = "translate", r = r)[['trans']] - r
  }
  X.ls[[rep]] <- create_curve_set(list(r = r, obs = obs2, sim_m = sim2))
}
# 5. Perform the adjusted test
res <- GET.composite(X=X, X.ls=X.ls, type="area",
                    r_min=rmin, r_max=rmax)
plot(res)

# Option 3: Provide fitfun and simfun functions
#-----
# fitfun = a function to fit the model
# simfun = a function to make a simulation from the model
# calcfun = a function to calculate the test vector from data/simulation
fitf <- function(X) {
  kppm(X, clusters = "MatClust", statistic="K")
}
simf <- function(M) {
  simulate(M, nsim=1)[[1]]
}
calcf <- function(X) {
  rmin <- 0.3; rmax <- 10; rstep <- (rmax-rmin)/500
  r <- seq(0, rmax, by=rstep)
  L <- Lest(X, correction = "translate", r = r)[['trans']] - r
  L[r >= rmin & r <= rmax]
}
res <- GET.composite(X = saplings, nsim=nsim, fitfun = fitf, simfun=simf, calcfun=calcf,
                    type="area")
plot(res, xlab="Index")
# Note: r-values are not passed; r in the x-axis is from 1 to length of r.
}

```

 GET.necdf

Graphical n sample test of correspondence of distribution functions

Description

Compare the distributions of two (or more) groups.

Usage

```
GET.necdf(x, r = seq(min(unlist((lapply(x, min)))), max(unlist((lapply(x,
  max)))), length = 100), summaryfun = c("means", "contrasts"), nsim,
  ...)
```

Arguments

x	A list (of length n) of values in the n groups.
r	The sequence of argument values at which the distribution functions are compared. The default is 100 equally spaced values between the minimum and maximum over all groups.
summaryfun	Possible values are "means" and "contrasts". See graph.fanova for more information.
nsim	The number of random permutations.
...	Additional parameters to be passed to global_envelope_test .

Details

The default number of permutation, if nsim is not specified, is $n \cdot 1000 - 1$ for the case "means" and $(n \cdot (n-1)/2) \cdot 1000 - 1$ for the case "contrasts", 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=grey(0.7), add=TRUE)
  # Create a list of the data
  fm.list <- list(Girls=f.a, Boys=m.a)

  res_m <- GET.necdf(fm.list, summaryfun="means")
  plot(res_m)
  res_c <- GET.necdf(fm.list, summaryfun="contrasts")
  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=grey(0.7), add=TRUE)
# Create a list of the data
fm.list <- list(Girls=f.a, Boys=m.a)

res_m <- GET.necdf(fm.list, summaryfun="means")
plot(res_m)
res_c <- GET.necdf(fm.list, summaryfun="contrasts")
plot(res_c)

}

```

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

<code>object</code>	An object of class <code>gstat</code> or a <code>variogram.formula</code> . In the first case, direct (residual) variograms are calculated for the variable defined in <code>object</code> . 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. <code>z~1</code> . See <code>variogram</code> .
<code>nsim</code>	The number of permutations.
<code>data</code>	A data frame where the names in formula are to be found. If <code>NULL</code> , the data are assumed to be found in the <code>object</code> .
<code>...</code>	Additional parameters to be passed to <code>variogram</code> .
<code>GET.args</code>	A named list of additional arguments to be passed to <code>global_envelope_test</code> .
<code>savefuns</code>	Logical. If <code>TRUE</code> , then the functions from permutations are saved to the attribute <code>simfuns</code> .

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 library.
  # 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, base_size=10)

  # 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)
}

```

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 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 <code>savefuns = TRUE</code> when calling envelope . 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.
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 <code>type = 'rank'</code> . 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 <code>theo</code> 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.
nstep	1 or 2 for how to construct 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 forder .

Details

Given a `curve_set` (see [create_curve_set](#) for how to create such an object) or an `envelope` object, which contains both the data curve (or function or vector) $T_1(r)$ (in the component `obs`) and the simulated curves $T_2(r), \dots, T_{s+1}(r)$ (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 constant width and that corresponds to the classical maximum deviation test (Ripley, 1981).

See [forder](#) and [central_region](#) and the references for more detailed description of the measures and the corresponding envelopes.

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.

Value

Either an object of class `"global_envelope"` and `"fv"` (see [fv.object](#)) or `"combined_global_envelope"` for combined tests. The objects can be printed and plotted directly.

The "global_envelope" is essentially a data frame containing columns

- `r` = the vector of values of the argument `r` at which the test was made
- `obs` = values of the data function
- `lo` = the lower envelope based on the simulated functions
- `hi` = the upper envelope based on the simulated functions
- `central` = If the `curve_set` (or envelope object) contains a component 'theo', then this function is used as the central curve and returned in this component. Otherwise, the `central_curve` is the mean of the test functions $T_i(r)$, $i=2, \dots, s+1$. Used for visualization only.

Moreover, the return value has the same attributes as the 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 corresponding to the components of `curve_sets`. The second level envelope on which the envelope construction is based on is saved in the attribute "level2_ge".

Ranking of the curves

The options for measures to order the functions from the most extreme one to the least extreme one are given by the argument `type`: 'rank', 'erl', 'cont', 'area', 'qdir', 'st', 'unscaled'. 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)

See more detailed description of the envelopes and measures in [central_region](#) and [forder](#).

Global envelope

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

P-values

In the case `type="rank"`, based on the extreme ranks k_i , $i=1, \dots, s+1$, 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"), but another option can be used by specifying `ties` argument.

If the case `type = "erl"`, the (single) p-value based on the extreme rank length ordering of the functions is calculated and returned in the attribute `p`. The same is done for other measures, the p-value always being correspondent to the chosen measure.

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. 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.

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 (2019). 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. (2018). A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME]
- Mrkvička, T., Myllymäki, M. and Narisetty, N. N. (2019) New methods for multiple testing in permutation inference for the general linear model. arXiv:1906.09004 [stat.ME]
- 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., Mrkvička, T. Global envelopes in R.
- Ripley, B.D. (1981). *Spatial statistics*. Wiley, New Jersey.

See Also

[plot.global_envelope](#), [central_region](#), [global_envelope_test2d](#), [GET.composite](#)

Examples

```

if(require("spatstat", quietly=TRUE)) {
  ## Testing complete spatial randomness (CSR)
  #-----
  nsim <- 1999 # Number of simulations

  pp <- unmark(spruces)
  # Generate nsim simulations under CSR, calculate L-function for the data and simulations
  env <- envelope(pp, fun="Lest", nsim=nsim,
                 savefuns=TRUE, # save the functions
                 correction="translate", # edge correction for L
                 simulate=expression(runifpoint(ex=pp))) # Simulate CSR
  # The rank envelope test (extreme rank length (ERL) breaking of ties)
  res <- global_envelope_test(env)
  # Plot the result.
  # - The central curve is now obtained from env[['theo']], which is the
  # value of the L-function under the null hypothesis (L(r) = r).
  # - Three different plot styles are provided:
  # a) ggplot2 style (requires ggplot2)
  plot(res, plot_style="ggplot2")
  # b) spatstat's style (requires spatstat)
  plot(res, plot_style="fv")
  # c) a basic style
  plot(res, plot_style="basic")

  ## 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)
  # For better visualisation, take the L(r)-r function
  curve_set <- residual(curve_set, use_theo=TRUE)
  # Do the rank envelope test (erl)
  res <- global_envelope_test(curve_set); plot(res, ylab=expression(italic(L(r)-r)))

  ## Random labeling test
  #-----
  mpp <- spruces
  # 1) Perform simulations under the random labelling hypothesis and calculate
  # the test function T(r) for the data pattern (mpp) and each simulation.
  # The command below specifies that the test function is  $T(r) = \hat{L}_{mm}(r)$ ,
  # which is an estimator of the mark-weighted L function,  $L_{mm}(r)$ ,
  # with translational edge correction.
  nsim <- 1999 # Number of simulations
  env <- envelope(mpp, fun=Kmark, nsim = nsim, f=function(m1, m2) { m1*m2 },
                 correction="translate", returnL=TRUE,
                 simulate=expression(rlabel(mpp, permute=TRUE)), # Permute the marks
                 savefuns=TRUE) # Save the functions

  # 2)

```

```

# Crop curves to desired r-interval
curve_set <- crop_curves(env, r_min=1.5, r_max=9.5)
# Center the functions, i.e. take  $\hat{L}_{mm}(r)-T_0(r)$ .
# Below  $T_0(r) = \hat{L}(r)$  is the mean of simulated functions.
# (This is only for visualization, does not affect the test result.)
curve_set <- residual(curve_set)
# 3) Do the rank envelope test
res <- global_envelope_test(curve_set)
# 4) Plot the test result
plot(res, ylab=expression(italic(L[m](r)-L(r))))

## Goodness-of-fit test (typically conservative, see ?GET.composite for adjusted tests)
#-----
pp <- unmark(spruces)
# Minimum distance between points in the pattern
min(nndist(pp))
# Fit a model
fittedmodel <- ppm(pp, interaction=Hardcore(hc=1)) # Hardcore process

# Simulating Gibbs process by 'envelope' is slow, because it uses the MCMC algorithm
#env <- envelope(fittedmodel, fun="Jest", nsim=999, savefuns=TRUE,
#               correction="none", r=seq(0, 4, length=500))

# Using direct algorithm can be faster, because the perfect simulation is used here.
simulations <- NULL
nsim <- 999 # Number of simulations
for(j in 1:nsim) {
  simulations[[j]] <- rHardcore(beta=exp(fittedmodel$coef[1]),
                               R=fittedmodel$interaction$par$hc,
                               W=pp$window)
  if(j%10==0) cat(j, "...", sep="")
}
env <- envelope(pp, simulate=simulations, fun="Jest", nsim=length(simulations),
               savefuns=TRUE, correction="none", r=seq(0, 4, length=500))
curve_set <- crop_curves(env, r_min=1, r_max=3.5)
res <- global_envelope_test(curve_set, type="er1"); plot(res, ylab=expression(italic(J(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 <- saplings

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(curve_set_L, curve_set_F,
  curve_set_G, curve_set_J))
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)
}

```

global_envelope_test2d

2D global envelope test

Description

Provides global envelope tests based on 2D functions

Usage

```
global_envelope_test2d(image_sets, ...)
```

Arguments

`image_sets` An image set, i.e. a set of 2d functions (or a list of them). See [create_image_set](#).
`...` Additional parameters to be passed to [global_envelope_test](#).

Value

An object of class "global_envelope2d" (and "list"), which can be printed and plotted directly.
Essentially a data frame containing columns

- `r` = a list of vectors of values of x- and y-coordinates at which the test was made
- `obs` = the data function (matrix), if there is only one data function. Otherwise not existing.
- `lo` = the lower envelope (matrix) based on the simulated functions
- `hi` = the upper envelope (matrix) based on the simulated functions
- `central` = the central curve, mean (default) or median of the test functions $T_i(r)$, $i=2, \dots, s+1$.
Used for visualization only.

Additionally, the return value has attributes `method`, `type`, `alternative`, `k_alpha`, `alpha`, `k`, `p` (and `p_interval` and `ties` if type is 'rank') and `call`, see more detailed description in [global_envelope_test](#).

References

- 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: 10.1016/j.spasta.2016.04.005
- 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., Hahn, U. and Myllymäki, M. (2018). A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME]

Examples

```

# Example of spatial point pattern residuals
#-----
if(require("spatstat", quietly=TRUE)) {
  data(cells)
  X <- cells
  # Fit the hard-core process
  model <- ppm(X, interaction=Hardcore())
  summary(model)
  HD <- 0.08168525 # Hard-core process
  # Choose a bandwidth by Scott's rule of thumb
  ds <- bw.scott(X); ds
  # Calculate raw residuals of the fitted model
  u <- diagnose.ppm(model, type="raw", rbord = HD, which = "smooth",
                    sigma=ds, plot.it=FALSE)

  obs <- u$smooth$Z$v
  # Generate simulations from the hard-core null model
  nsim <- 499 # Number of simulations; increase for serious analysis!
  simulations <- NULL
  ext.factor <- max(X$window$xrange[2]-X$window$xrange[1],
                  X$window$yrange[2]-X$window$yrange[1]) / 10
  win.extend <- owin(c(X$window$xrange[1]-ext.factor, X$window$xrange[2]+ext.factor),
                   c(X$window$yrange[1]-ext.factor, X$window$yrange[2]+ext.factor))
  mod02 <- list(cif="hardcore", par=list(beta=exp(model$fitin$coefs[1]),hc=HD), w=win.extend)
  # starting point pattern in an extended window
  x.start <- runifpoint(X$n, win=win.extend)
  # simulations
  for(sss in 1:nsim){
    uppp <- rmh(model=mod02, start=list(x.start=x.start), control=list(p=1,nrep=1e5,nverb=5000))
    f <- uppp$x > X$window$xrange[1] & uppp$x < X$window$xrange[2] &
        uppp$y > X$window$yrange[1] & uppp$y < X$window$yrange[2]
    simulations[[sss]] <- ppp(uppp$x[f], uppp$y[f], window=X$window)
  }
  # Calculate the raw residuals for simulations
  sim <- array(0, c(u$smooth$Z$dim, nsim))
  for(i in 1:length(simulations)) {
    model=ppm(simulations[[i]],interaction=Hardcore(HD));
    u_sim <- diagnose.ppm(model, type="raw", rbord = HD, which = "smooth", sigma=ds, plot.it=FALSE)
    sim[, ,i] <- u_sim$smooth$Z$v
    if((i %% 100)==0) cat(i, ' ')
  }
  # Construct the global envelope test for the (2D) raw residuals
  iset <- create_image_set(list(obs=obs, sim_m=sim))
  res <- global_envelope_test2d(iset, type="area")
  plot(res)
  plot(res, contours=FALSE) + ggplot2::scale_fill_gradient(low="black", high="white")
  plot(res, fixedscales=FALSE)
}

```

graph.fanova	<i>One-way graphical functional ANOVA</i>
--------------	---

Description

One-way ANOVA tests for functional data with graphical interpretation

Usage

```
graph.fanova(nsim, curve_set, groups, variances = "equal",
             summaryfun = c("means", "contrasts"), 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).
variances	Either "equal" or "unequal". If "unequal", then correction for unequal variances as explained in details will be done.
summaryfun	Possible values are "means" and "contrasts". See description for their meaning.
n.aver	If variances = "unequal", there is a possibility to use variances smoothed by applying 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 .
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 <div style="text-align: center; margin: 10px 0;"> $Z_{ij}(r) = T_{ij}(r) - \bar{T}_j(r).$ </div> If cov, then the equality of lag cov.lag covariance is tested by performing the fANOVA with <div style="text-align: center; margin: 10px 0;"> $W_{ij}(r) = \sqrt{ V_{ij}(r) \cdot \text{sign}(V_{ij}(r))},$ </div> where <div style="text-align: center; margin: 10px 0;"> $V_{ij}(r) = (T_{ij}(r) - \bar{T}_j(r))(T_{ij}(r + s) - \bar{T}_j(r + s)).$ </div> See Mrkvicka et al. (2018) 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 .

Details

This functions can be used to perform one-way graphical functional ANOVA tests described in Mrkvička et al. (2016).

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 $T_{ij}, i = 1, \dots, n_j$ is an iid sample from a stochastic process with mean function μ_j and covariance function $\gamma_j(s, t)$ for s, t in \mathbf{R} and $j = 1, \dots, J$.

If you want to test the hypothesis

$$H_0 : \mu_j(r) \equiv 0, j = 1, \dots, J,$$

then you should use the test function

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

where $\bar{T}_i(\mathbf{r})$ is a vector of mean values of functions in the group j . This can be done by choosing the summaryfun "means".

An alternative is to test the equivalent hypothesis

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

This test corresponds to the post-hoc test done usually after an ANOVA test is significant, but it can be directed tested by mean of the combined rank test (Mrkvička et al., 2017), if the test vector is taken to consist of the differences of the group averages of test functions, namely

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

The summaryfun option "contrasts" can be used to perform the test based on 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

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

where $\bar{T}(\mathbf{r})$ 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., Hahn, U. and Myllymäki, M. A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME] (<http://arxiv.org/abs/1612.03608>)

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

See Also

[graph.fanova2d](#), [frank.fanova](#)

Examples

```

#-- 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,
                        groups = attr(cgec, "group"),
                        test.equality = "cov", cov.lag = 1)
plot(res.cov1, max_ncols_of_plots=3,
     labels = paste("Group ", 1:3, sep=""),
     xlab=substitute(paste(i, "(", italic(j), ") ", sep=""), list(i="Year", j="r")),
     ylab=expression(italic(bar(W)[i](r))))
# Test for equality of variances among groups
res.var <- graph.fanova(nsim = nsim, curve_set = cgec,
                       groups = attr(cgec, "group"),
                       test.equality = "var")
plot(res.var, max_ncols_of_plots=3,
     labels = paste("Group ", 1:3, sep=""),
     xlab=substitute(paste(i, "(", italic(j), ") ", sep=""), list(i="Year", j="r")),
     ylab=expression(italic(bar(Z)[i](r))))

# Test for equality of means assuming equality of variances
# a) using 'means'
res <- graph.fanova(nsim = nsim, curve_set = cgec,
                   groups = attr(cgec, "group"),
                   variances = "equal",
                   summaryfun = "means")
plot(res, max_ncols_of_plots=3,
     labels = paste("Group ", 1:3, sep=""),
     xlab=substitute(paste(i, "(", italic(j), ") ", sep=""), list(i="Year", j="r")),
     ylab=expression(italic(bar(T)[i](r))))
# b) using 'contrasts'
res2 <- graph.fanova(nsim = nsim, curve_set = cgec,
                    groups = attr(cgec, "group"),
                    variances = "equal",
                    summaryfun = "contrasts")

plot(res2, max_ncols_of_plots=3,
     xlab=substitute(paste(i, "(", italic(j), ") ", sep=""), list(i="Year", j="r")),
     ylab=expression(italic(bar(T)[i](r)-bar(T)[j](r))))

#-- Rimov water temperatures example
# This is an example analysis of the water temperature data set
# in Mrkvicka et al. (arXiv:1612.03608v2).
data(rimov)

```

```

groups <- factor(c(rep(1, times=12), rep(2, times=12), rep(3, times=12)))
nsim <- 999

# Test for equality of variances in the groups
resV <- graph.fanova(nsim=nsim, curve_set=rimov, groups=groups, summaryfun="means",
                    test.equality="var")
plot(resV)
# Test for equality of lag 1 covariances in the groups
resC <- graph.fanova(nsim=nsim, curve_set=rimov, groups=groups, summaryfun="means",
                    test.equality="cov", cov.lag=1)
plot(resC)

# Test the equality of means in the groups (fANOVA), assuming equality of variances
res <- graph.fanova(nsim=nsim, curve_set=rimov, groups=groups, summaryfun="means")
plot(res)
res2 <- graph.fanova(nsim=nsim, curve_set=rimov, groups=groups, summaryfun="contrasts")
plot(res2)

```

graph.fanova2d

One-way graphical functional ANOVA for images

Description

One-way ANOVA tests for image data with graphical interpretation

Usage

```
graph.fanova2d(nsim, image_set, groups, ...)
```

Arguments

nsim	The number of random permutations.
image_set	A set of images containing the data, see create_image_set .
groups	The original groups (a factor vector representing the assignment to groups).
...	Additional parameters to be passed to graph.fanova .

Details

This function can be used to perform one-way graphical functional ANOVA tests described in Mrkvička et al. (2018). The function transforms images to vectors (1d), calls [graph.fanova](#) and transform results back to images (2d).

Value

A `global_envelope2d` or `combined_global_envelope2d` object, which can be printed and plotted directly.

References

Mrkvička, T., Myllymäki, M., Jilek, M. and Hahn, U. (2018) A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME] (<http://arxiv.org/abs/1612.03608>)

Examples

```
data("imageset1")
res <- graph.fanova2d(nsim = 19, # Increase nsim for serious analysis!
                     image_set = imageset1$image_set,
                     groups = imageset1$Group)

plot(res)
# Contrasts
res.c <- graph.fanova2d(nsim = 19, # Increase nsim for serious analysis!
                       image_set = imageset1$image_set,
                       groups = imageset1$Group,
                       summaryfun = "contrasts")

plot(res.c)
plot(res.c, contours=FALSE)
```

graph.fglm

Graphical functional GLM

Description

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

Usage

```
graph.fglm(nsim, formula.full, formula.reduced, curve_sets,
           factors = NULL, summaryfun = c("means", "contrasts"),
           savefuns = FALSE, ..., GET.args = NULL, mc.cores = 1L,
           mc.args = NULL)
```

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.
curve_sets	A named list of sets of curves giving the dependent variable (Y), and possibly additionally all the factors. The dimensions of the elements should match with each other, i.e. the factor values should be given for each argument value and each function. If factors are given in the argument factors, then can also be just the curve set representing Y. Also fdata objects allowed.

factors	A data frame of factors. An alternative way to specify factors when they are constant for all argument values. 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.
summaryfun	Possible values are "means" and "contrasts". See description for their meaning.
savefuns	Logical. If TRUE, then the functions from permutations are saved to the attribute simfuns.
...	Additional arguments to be passed to <code>lm</code> .
GET.args	A named list of additional arguments to be passed to <code>global_envelope_test</code> .
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 <code>mclapply</code> , 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 <code>mclapply</code> . Only relevant if mc.cores is more than 1.

Details

The function `graph.fglm` performs the graphical functional GLM of Mrkvička et al. (2019). 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.

Value

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

References

Mrkvička, T., Roskovec, T. and Rost, M. (2019) A nonparametric graphical tests of significance in functional GLM. arXiv:1902.04926 [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.fglm(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)

data(GDPtax)
factors.df <- data.frame(Group = GDPtax$Group, Tax = GDPtax$Profittax)
res.tax_within_group <- graph.fglm(nsim = 999,
  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)

```

graph.fglm2d

Graphical functional GLM for images

Description

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

Usage

```

graph.fglm2d(nsim, formula.full, formula.reduced, image_sets,
  factors = NULL, ...)

```

Arguments

<code>nsim</code>	The number of random permutations.
<code>formula.full</code>	The formula specifying the general linear model, see formula in lm .
<code>formula.reduced</code>	The formula of the reduced model with nuisance factors only.
<code>image_sets</code>	A named list of sets of images giving the dependent variable (Y), and possibly additionally all the factors. The dimensions of the elements should match with each other, i.e. the factor values should be given for each argument value and each function.
<code>factors</code>	A data frame of factors. An alternative way to specify factors when they are constant for all argument values. 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.
<code>...</code>	Additional parameters to be passed to graph.fglm .

Value

A `global_envelope2d` or `combined_global_envelope2d` object, which can be printed and plotted directly.

References

Mrkvička, T., Roskovec, T. and Rost, M. (2019) A nonparametric graphical tests of significance in functional GLM. arXiv:1902.04926 [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

See Also

[graph.fglm](#), [frank.fglm2d](#)

Examples

```
data("imageset2")
# Testing discrete factor group
res.g <- graph.fglm2d(nsim = 19, # Increase nsim for serious analysis!
  formula.full = Y ~ group + z,
  formula.reduced = Y ~ z,
  image_sets = list(Y = imageset2$image_set),
  factors = data.frame(group = imageset2$Group,
    z = imageset2$z))

plot(res.g)
# Testing discrete factor group with contrasts
res.gc <- graph.fglm2d(nsim = 19, # Increase nsim for serious analysis!
  formula.full = Y ~ group + z,
  formula.reduced = Y ~ z,
  image_sets = list(Y = imageset2$image_set),
  factors = data.frame(group = imageset2$Group,
    z = imageset2$z),
  summaryfun = "contrasts")

plot(res.gc)

# Testing continuous factor z
res.z <- graph.fglm2d(nsim = 19, # Increase nsim for serious analysis!
  formula.full = Y ~ group + z,
  formula.reduced = Y ~ group,
  image_sets = list(Y = imageset2$image_set),
  factors = data.frame(group = imageset2$Group,
    z = imageset2$z))

plot(res.z)
```

imageset1	A simulated set of images
-----------	---------------------------

Description

A simulated set of images with a categorical factor

Usage

```
data(imageset1)
```

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 or 1 according to the group to which the image belongs to (10 images in the first group, 10 images in the second). The images were simulated in the square window $[-1,1]^2$ from the general linear model (GLM)

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

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 \mathbb{R}^2 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.

References

Mrkvička, T., Myllymäki, M. and Narisetty, N. N. (2019) New methods for multiple testing in permutation inference for the general linear model. arXiv:1906.09004 [stat.ME]

See Also

[graph.fanova2d](#), [frank.fanova2d](#)

Examples

```
data(imageset1)
plot(imageset1$image_set, idx=c(1:5, 11:15), max_ncols_of_plots = 5)
# Colors can be changed as follows:
plot(imageset1$image_set, idx=c(1:5, 11:15), max_ncols_of_plots = 5) +
  ggplot2::scale_fill_gradient(low="black", high="white")
```

 imageset2

 A simulated set of images

Description

A simulated set of images with two simulated covariates

Usage

```
data(imageset2)
```

Format

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

Details

We considered a categorical factor `Group` obtaining the values 0 or 1 according to the group to which the image belongs to (10 images in the first group, 10 images in the second), and a continuous factor `z` that was generated from the uniform distribution on (0,1). The images were simulated in the square window $[-1,1]^2$ from the general linear model (GLM)

$$Y(r) = \exp(-10 \cdot \|r\|) \cdot (1 + g + z) + \epsilon(r),$$

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 \mathbb{R} 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.

References

Mrkvička, T., Myllymäki, M. and Narisetty, N. N. (2019) New methods for multiple testing in permutation inference for the general linear model. arXiv:1906.09004 [stat.ME]

See Also

[graph.fglm2d](#), [frank.fglm2d](#)

Examples

```
data(imageset2)
plot(imageset2$image_set, idx=c(1:5, 11:15), max_ncols_of_plots=5)
```

`imageset3`*A simulated set of images*

Description

A simulated set of images with a categorical factor

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 \mathbb{R}^2 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.

References

Mrkvička, T., Myllymäki, M. and Narisetty, N. N. (2019) New methods for multiple testing in permutation inference for the general linear model. arXiv:1906.09004 [stat.ME]

See Also

[graph.fanova2d](#), [frank.fanova2d](#)

Examples

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

is.curve_set	<i>Check class.</i>
--------------	---------------------

Description

Check class.

Usage

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

Arguments

x	An object to be checked.
---	--------------------------

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, plot_style = c("ggplot2", "fv",
  "basic"), level = 1, outliers = TRUE, bp.col = 2, cr.col = 1,
  ...)
```

Arguments

x	an 'combined_fboxplot' object
plot_style	One of the following "basic", "fv" or "ggplot2". The option "basic" (default) offers a very basic global envelope plot. The option "fv" utilizes the plot routines of the function value table fv.object . For "ggplot2", a plot with a coloured envelope ribbon is provided. Requires R library ggplot2. The option "fv" is currently only available for tests with one test function, whereas the other true allow also tests with several tests functions.
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.
outliers	Logical. If TRUE, then the functions outside the functional boxplot are drawn.
bp.col	The color for the boxplot bounds. Default 2 (red).
cr.col	The color for the central region bounds.
...	Additional arguments to be passed to plot.combined_global_envelope .

 plot.combined_global_envelope

Plot method for the class 'combined_global_envelope'

Description

Plot method for the class 'combined_global_envelope'

Usage

```
## S3 method for class 'combined_global_envelope'
plot(x, plot_style = c("ggplot2",
  "fv", "basic"), main, ylim, xlab, ylab, color_outside = TRUE,
  env.col = 1, base_size = 15, labels = NULL, add = FALSE,
  digits = 3, level = 1, max_ncols_of_plots = 2, nticks = 5,
  legend = TRUE, ...)
```

Arguments

x	An 'combined_global_envelope' object
plot_style	One of the following "basic", "fv" or "ggplot2". The option "basic" (default) offers a very basic global envelope plot. The option "fv" utilizes the plot routines of the function value table fv.object . For "ggplot2", a plot with a coloured envelope ribbon is provided. Requires R library ggplot2. The option "fv" is currently only available for tests with one test function, whereas the other true allow also tests with several tests functions.
main	See plot.default . A sensible default exists.
ylim	See plot.default . A sensible default exists.
xlab	See plot.default . A sensible default exists.
ylab	See plot.default . A sensible default exists.
color_outside	Logical. Whether to color the places where the data function goes outside the envelope. Currently red color is used. Relevant only for plot_style = "basic".
env.col	The color for the envelope lines (or dotplot arrows). Default 1 (black).
base_size	Base font size, to be passed to theme style when plot_style = "ggplot2".
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.
add	Whether to add the plot to an existing plot (TRUE) or to draw a new plot (FALSE). Not available for plot_style = "ggplot2".
digits	The number of digits used for printing the p-value or p-interval in the main, if using 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.

max_ncols_of_plots	The maximum number of columns for the figures. Default 2. (Relates to the number of curve_sets that have been combined.)
nticks	The number of ticks on the xaxis.
legend	Logical. If FALSE, then the legend is removed from the "ggplot2" style plot.
...	Additional parameters to be passed to plot or lines .

See Also

[central_region](#)

plot.combined_global_envelope2d
Plot method for the class 'combined_global_envelope2d'

Description

Plot method for the class 'combined_global_envelope2d'

Usage

```
## S3 method for class 'combined_global_envelope2d'
plot(x, plot_style = c("ggplot2",
  "basic"), fixedscales = 2, sign.col = "red", transparency = 85/255,
  contours = FALSE, main = NULL, digits = 3, ...)
```

Arguments

x	an 'combined_global_envelope2d' object
plot_style	Either "ggplot2" or "basic". (Similar to the argument in plot.global_envelope .)
fixedscales	0, 1 or 2. See details.
sign.col	The color for the significant regions. Default to "red".
transparency	A number between 0 and 1 (default 85/255, 33 Similar to alpha of rgb . Used in plotting the significant regions.
contours	Logical. Whether to draw contour lines to the observed function and the lower and upper envelope.
main	The overall main. Default exists.
digits	The number of digits used for printing the p-value or p-interval in the main, if using the default main.
...	Additional parameters to be passed to plot.im .

Details

Additional parameter `col` can be passed in ... to `plot.im`. If `col` not given, a `colourmap` of 255 grey values between the minimum and maximum of the function values is used for each image separately. If `col` is provided, the same specification will be used for each produced plot, which may make it easier to compare the figures with each other.

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

`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, ylim, xlab = "r", ylab = "obs",
     col_obs = 1, col_sim = grDevices::grey(0.7), ...)
```

Arguments

<code>x</code>	An <code>curve_set</code> object
<code>ylim</code>	The y limits of the plot with the default being the minimum and maximum over all curves.
<code>xlab</code>	The label for the x-axis. Default "r".
<code>ylab</code>	The label for the y-axis. Default "obs".
<code>col_obs</code>	Color for 'obs' in the argument x.
<code>col_sim</code>	Color for 'sim_m' in the argument x.
...	Additional parameters to be passed to plot and lines.

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

Description

Plot method for the class 'fboxplot'

Usage

```
## S3 method for class 'fboxplot'
plot(x, plot_style = c("ggplot2", "fv", "basic"),
     dotplot = length(x$r) < 10, outliers = TRUE, bp.col = 2,
     cr.col = 1, ...)
```

Arguments

x	an 'fboxplot' object
plot_style	One of the following "basic", "fv" or "ggplot2". The option "basic" (default) offers a very basic global envelope plot. The option "fv" utilizes the plot routines of the function value table fv.object . For "ggplot2", a plot with a coloured envelope ribbon is provided. Requires R library ggplot2. The option "fv" is currently only available for tests with one test function, whereas the other true allow also tests with several tests functions.
dotplot	Logical. If TRUE, then instead of envelopes a dot plot is done. Suitable for low dimensional test vectors. Only applicable if plot_style is "basic". Default: TRUE if the dimension is less than 10, FALSE otherwise.
outliers	Logical. If TRUE, then the functions outside the functional boxplot are drawn.
bp.col	The color for the boxplot bounds. Default 2 (red).
cr.col	The color for the central region bounds.
...	Additional arguments to be passed to plot.global_envelope .

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, plot_style = c("ggplot2", "fv",
  "basic"), dotplot = length(x$r) < 10, main, ylim, xlab, ylab,
  color_outside = TRUE, env.col = 1, base_size = 15, labels = NULL,
  add = FALSE, digits = 3, legend = TRUE, ...)
```

Arguments

x	An 'global_envelope' object
plot_style	One of the following "basic", "fv" or "ggplot2". The option "basic" (default) offers a very basic global envelope plot. The option "fv" utilizes the plot routines of the function value table fv.object . For "ggplot2", a plot with a coloured envelope ribbon is provided. Requires R library ggplot2. The option "fv" is currently only available for tests with one test function, whereas the other true allow also tests with several tests functions.
dotplot	Logical. If TRUE, then instead of envelopes a dot plot is done. Suitable for low dimensional test vectors. Only applicable if plot_style is "basic". Default: TRUE if the dimension is less than 10, FALSE otherwise.
main	See plot.default . A sensible default exists.

ylim	See plot.default . A sensible default exists.
xlab	See plot.default . A sensible default exists.
ylab	See plot.default . A sensible default exists.
color_outside	Logical. Whether to color the places where the data function goes outside the envelope. Currently red color is used. Relevant only for plot_style = "basic".
env.col	The color for the envelope lines (or dotplot arrows). Default 1 (black).
base_size	Base font size, to be passed to theme style when plot_style = "ggplot2".
labels	A character vector of suitable length. If dotplot = TRUE, then labels for the tests at x-axis.
add	Whether to add the plot to an existing plot (TRUE) or to draw a new plot (FALSE). Not available for plot_style = "ggplot2".
digits	The number of digits used for printing the p-value or p-interval in the main, if using the default main.
legend	Logical. If FALSE, then the legend is removed from the "ggplot2" style plot.
...	Additional parameters to be passed to plot or lines .

See Also

[central_region](#)

plot.global_envelope2d

Plot method for the class 'global_envelope2d'

Description

Plot method for the class 'global_envelope2d'

Usage

```
## S3 method for class 'global_envelope2d'
plot(x, plot_style = c("ggplot2", "basic"),
     fixedscales = TRUE, sign.col = "red", transparency = 85/255,
     contours = FALSE, main = NULL, digits = 3, ...)
```

Arguments

x	an 'global_envelope2d' object
plot_style	Either "ggplot2" or "basic". (Similar to the argument in plot.global_envelope .)
fixedscales	Logical. TRUE for the same scales for all images.
sign.col	The color for the significant regions. Default to "red".
transparency	A number between 0 and 1 (default 85/255, 33 Similar to alpha of rgb . Used in plotting the significant regions.

contours	Logical. Whether to draw contour lines to the observed function and the lower and upper envelope.
main	The overall main. Default exists.
digits	The number of digits used for printing the p-value or p-interval in the main, if using the default main.
...	Additional parameters to be passed to <code>plot.im</code> .

Details

Additional parameter `col` can be passed in ... to `plot.im`. If `col` not given, a `colourmap` of 255 grey values between the minimum and maximum of the function values is used for each image separately. If `col` is provided, the same specification will be used for each produced plot, which may make it easier to compare the figures with each other.

<code>plot.image_set</code>	<i>Plot method for the class 'image_set'</i>
-----------------------------	--

Description

Plot method for the class 'image_set'

Usage

```
## S3 method for class 'image_set'
plot(x, idx = 1, obs = TRUE,
     plot_style = c("ggplot2", "basic"), main, col,
     max_ncols_of_plots = 4, ...)
```

Arguments

<code>x</code>	an 'image_set' object
<code>idx</code>	Indices of the images in the image_set to be plotted.
<code>obs</code>	Logical. TRUE, then <code>idx</code> is understood as an index to <code>image_set\$obs</code> , otherwise to <code>image_set\$sim_m</code> .
<code>plot_style</code>	Either "ggplot2" or "basic". (Similar to the argument in <code>plot.global_envelope</code> .)
<code>main</code>	The title. Default exists.
<code>col</code>	Colours to be passed to <code>plot.im</code> if <code>plot_style = "basic"</code> . A default exists.
<code>max_ncols_of_plots</code>	The maximum number of columns for the figures. Default 4.
...	Additional parameters to be passed to <code>plot.im</code> .

```
print.combined_global_envelope
```

Print method for the class 'global_envelope'

Description

Print method for the class 'global_envelope'

Usage

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

Arguments

x an 'combined_global_envelope' object
... Ignored.

```
print.combined_global_envelope2d
```

Print method for the class 'combined_global_envelope2d'

Description

Print method for the class 'combined_global_envelope2d'

Usage

```
## S3 method for class 'combined_global_envelope2d'  
print(x, ...)
```

Arguments

x an 'combined_global_envelope2d' 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.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` an 'global_envelope' object
`...` Ignored.

`print.global_envelope2d`
Print method for the class 'global_envelope2d'

Description

Print method for the class 'global_envelope2d'

Usage

```
## S3 method for class 'global_envelope2d'  
print(x, ...)
```

Arguments

`x` an 'global_envelope2d' object
`...` Ignored.

print.image_set	<i>Print method for the class 'image_set'</i>
-----------------	---

Description

Print method for the class 'image_set'

Usage

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

Arguments

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

qdir_envelope	<i>Global scaled maximum absolute difference (MAD) envelope tests</i>
---------------	---

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 <code>curve_set</code> (see create_curve_set) or an <code>envelope</code> object. If an envelope object is given, it must contain the summary functions from the simulated patterns which can be achieved by setting <code>savefuns = TRUE</code> when calling envelope .
...	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" and "fv" (see [fv.object](#)), 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", quietly=TRUE)) {
  pp <- spruces
  nsim <- 999 # Number of simulations

  ## Test for complete spatial randomness (CSR)
  # Generate nsim simulations under CSR, calculate L-function for the data and simulations
  env <- envelope(pp, fun="Lest", nsim=nsim, savefuns=TRUE, correction="translate",
    simulate=expression(runifpoint(ex=pp)))
  res_qdir <- qdir_envelope(env) # The directional quantile envelope test
  plot(res_qdir)
  # or (requires R library ggplot2)
  plot(res_qdir, plot_style="ggplot2")

  ## 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)
# For better visualisation, take the L(r)-r function
curve_set <- residual(curve_set, use_theo=TRUE)
# The directional quantile envelope test
res_qdir <- qdir_envelope(curve_set); plot(res_qdir, plot_style="ggplot2")
# The studentised envelope test
res_st <- st_envelope(curve_set); plot(res_st, plot_style="ggplot2")
# The unscaled envelope test
res_unscaled <- unscaled_envelope(curve_set); plot(res_unscaled, plot_style="ggplot2")
}

```

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 carried 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. If an envelope object is given, it must contain the summary functions from the simulated patterns which can be achieved by setting <code>savefuns = TRUE</code> when calling envelope .
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 "er1", "cont" or "area", then the global envelope corresponds to these measures.

Value

An object of class "global_envelope" and "fv" (see [fv.object](#)), which can be printed and plotted directly. See [global_envelope_test](#) for more details.

Number of simulations

The global "er1", "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. (2018). A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608 [stat.ME]

See Also

[global_envelope_test](#)

Examples

```
# See ?global_envelope_test for more examples

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

  # Generate nsim simulations under CSR, calculate L-function for the data and simulations
  env <- envelope(pp, fun="Lest", nsim=nsim, savefuns=TRUE, correction="translate",
                 simulate=expression(runifpoint(ex=pp)))

  # The rank envelope test
  res <- rank_envelope(env)
  # Plot the result.
  # - The central curve is now obtained from env[['theo']], which is the
```

```

# value of the L-function under the null hypothesis (L(r) = r).
plot(res)
# or (requires R library ggplot2)
plot(res, plot_style="ggplot2")

## 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)
# For better visualisation, take the L(r)-r function
curve_set <- residual(curve_set, use_theo=TRUE)
# Do the rank envelope test
res <- rank_envelope(curve_set); plot(res, plot_style="ggplot2")
}

```

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

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

Details

The mean of the functions in the `curve_set` is

- the mean of the functions provided in `obs`, if `obs` is a matrix (i.e. many functions in `obs`).
- the mean of the functions provided in `sim_m`, if `obs` is a vector (i.e. only one function in `obs`).

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	<i>Year temperature curves</i>
-------	--------------------------------

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., Hahn, U. and Myllymäki, M. (2018) A one-way ANOVA test for functional data with graphical interpretation. arXiv:1612.03608v2 [stat.ME] (<http://arxiv.org/abs/1612.03608v2>)

See Also

`graph.fanova`

Examples

```
data(rimov)
groups <- factor(c(rep(1, times=12), rep(2, times=12), rep(3, times=12)))
# Plot data in groups
subs <- function(group, ...) {
  cset <- rimov
  cset$obs <- rimov$obs[, groups == group]
  plot(cset, ...)
}
for(i in 1:3) subs(i, main=paste("group ", i, sep=""), ylab="Temperature")
# See example analysis in ?graph.fanova
```

 saplings

Saplings data set

Description

Saplings data set

Usage

```
data(saplings)
```

Format

An object of class `ppp.object` representing the point pattern of tree locations.

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

Examples

```
# This is an example analysis of the saplings data set in Myllymäki et al. (2017).
if(require("spatstat", quietly=TRUE)) {
  data(saplings)

  # 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(saplings$n, win=saplings>window)), # 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)
# Plot the result.
plot(res, 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", 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, 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, 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)
}
```

Index

- *Topic **curves**
 - cgec, 13
 - GDPtax, 30
 - imageset1, 57
 - imageset2, 58
 - imageset3, 59
 - rimov, 75
- *Topic **datasets**
 - adult_trees, 7
 - cgec, 13
 - fallen_trees, 21
 - GDPtax, 30
 - imageset1, 57
 - imageset2, 58
 - imageset3, 59
 - rimov, 75
 - saplings, 76
- *Topic **pattern**
 - fallen_trees, 21
- *Topic **point**
 - fallen_trees, 21
- *Topic **spatial**
 - adult_trees, 7
 - saplings, 76
- adult_trees, 6, 7
- attr, 10
- central_region, 3, 4, 8, 12, 15, 22, 41, 42, 44, 62, 65
- central_region2d, 4, 12
- cgec, 6, 13
- colourmap, 63, 66
- combined_scaled_MAD_envelope, 14
- create_curve_set, 4, 5, 8, 14, 16, 18, 23, 25, 32, 33, 40, 41, 49, 70, 72, 74
- create_image_set, 4, 12, 17, 26, 47, 52
- crop_curves, 5, 18
- deviation_test, 4, 18, 23, 24
- dg.global_envelope_test (GET.composite), 31
- envelope, 4–6, 8, 15, 18, 20, 23, 32–34, 40, 41, 70, 72, 74
- envelope_to_curve_set, 6, 20
- fallen_trees, 6, 21
- fBoxplot, 4, 22
- fdata, 25, 27, 49, 53
- filter, 49
- forder, 4, 8, 9, 22, 40–42
- frank.fanova, 4, 25, 26, 27, 50
- frank.fanova2d, 4, 6, 26, 57, 59
- frank.fglm, 4, 27, 29
- frank.fglm2d, 4, 6, 29, 56, 58
- fv, 10
- fv.object, 10, 41, 60, 61, 64, 71, 73
- GDPtax, 6, 30
- GET (global_envelope_test), 39
- GET-package, 3
- GET.2d (global_envelope_test2d), 47
- GET.composite, 4–6, 31, 44
- GET.necdf, 4, 37
- GET.variogram, 4, 38
- global_envelope (central_region), 8
- global_envelope_test, 4, 5, 11, 15, 25, 28, 32–34, 37, 38, 39, 41, 47, 49, 54, 70–73
- global_envelope_test2d, 4, 44, 47
- graph.fanova, 4, 6, 14, 37, 49, 52
- graph.fanova2d, 4, 6, 50, 52, 57, 59
- graph.fglm, 4, 28, 31, 53, 55, 56
- graph.fglm2d, 4, 6, 29, 55, 58
- growth, 3
- imageset1, 6, 57
- imageset2, 6, 58
- imageset3, 6, 59

is.curve_set, 60

kppm, 5, 32

lines, 62, 65

lm, 25, 27–29, 53–55

mclapply, 28, 33, 54

plot, 62, 65

plot.combined_fboxplot, 60

plot.combined_global_envelope, 60, 61

plot.combined_global_envelope2d, 62

plot.curve_set, 63

plot.default, 61, 64, 65

plot.fboxplot, 63

plot.global_envelope, 3, 4, 34, 44, 62, 64, 64, 65, 66

plot.global_envelope2d, 65

plot.im, 62, 63, 66

plot.image_set, 66

ppm, 5, 32

ppp, 5, 32–34

ppp.object, 7, 21, 76

print.combined_global_envelope, 67

print.combined_global_envelope2d, 67

print.curve_set, 68

print.deviation_test, 68

print.global_envelope, 69

print.global_envelope2d, 69

print.image_set, 70

qdir_envelope, 70

quantile, 8, 23, 40

rank_envelope, 72

residual, 6, 74

rgb, 62, 65

rimov, 6, 75

saplings, 4–6, 34, 76

spatstat, 3, 5, 32, 33

st_envelope (qdir_envelope), 70

str, 68, 70

unscaled_envelope (qdir_envelope), 70

variogram, 38