Package 'Morphoscape'

June 17, 2022

Type Package

Title Computation and Visualization of Adaptive Landscapes

Version 1.0.0

Description Implements adaptive landscape methods first described by Polly et al. (2016) <doi:10.1080/02724634.2016.1111225> for the integration, analysis and visualization of biological trait data on a phenotypic morphospace - typically defined by shape metrics.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 4.0.0)

Imports concaveman, ggplot2, sp, automap

Suggests alphahull, knitr, rmarkdown

VignetteBuilder knitr

URL https://blakedickson.github.io/Morphoscape/

RoxygenNote 7.1.2

NeedsCompilation no

Author Blake Dickson [aut, cre] (<https://orcid.org/0000-0001-6299-5224>), Stephanie Pierce [aut] (<https://orcid.org/0000-0003-0717-1841>), Noah Greifer [aut] (<https://orcid.org/0000-0003-3067-7154>)

Maintainer Blake Dickson <blake.dickson@duke.edu>

Repository CRAN

Date/Publication 2022-06-17 07:00:07 UTC

R topics documented:

`as_fnc_df` *Convert a data frame to a* `fnc_df`

Description

`as_fnc_df()` converts a data frame containing coordinates and functional charactertics in a morphological space to a `fnc_df` object for use in later funcitons, most importantly `krige_surf`.

Usage

```
as_fnc_df(x, func.names = NULL, scale = TRUE)
```

Arguments

`x`	a data frame containing coordinates and functional characteristics (and possibly other variables, which are ignored). The first two columns must correspond to the x and y coordinates of the warps in morphological space.
`func.names`	the names of the variables in `x` the correspond to functional characteristics. These charcteristics must be numeric variables. If `NULL` (the default), all variables other than the first two will be taken to be the functional characteristics under study.
`scale`	whether to scale the functional characteristics to have a minimum of 0 and a maximum of 1. This should generally be left at its default (`TRUE`) unless the variables have already been scaled.

Details

Input data can be from a sampled grid of locations in morphospace, measured specimen data, species or group means, or a mix.

Value

A `fnc_df` object, which is a `data.frame` with the x and y coordinates in the first two columns and the functional characteristics in the other columns. The `"func.names"` attribute contains the names of the functional characteristics.

See Also

as.data.frame

krige_surf for using an fnc_df object to create a kriged surface.

Examples

```
data("warps")

warps_fnc <- as_fnc_df(warps,
                       func.names = c("hydro", "curve",
                                      "mech", "fea"))

str(warps_fnc)
```

---

calcGrpWprime *Compute optimally weighted adaptive landscapes*

Description

calcGrpWprime() computes the optimally weighted adaptive landscape by searching through the adaptive landscapes formed from sets of weights and performance surfaces, and finding the set of weights that yields the greatest overall (average) fitness value (Z) across a sample of data or a subset thereof.

Usage

```
calcGrpWprime(x, index, method = "chi-squared",
              quantile = 0.05)
## S3 method for class 'grp_Wprime'
print(x,
      digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	for calcGrpWprime(), an all_lscps object; the output of a call to calc_all_lscps. for print(), a grp_Wprime object; the output of a call to calcGrpWprime()
index	an optional vector of indices indicating which subset of the new_data dataset originally supplied to krige_surf should be calculated. Can be specified as a vector of numerical indicies, logical indices, or row names. If unspecified, the optimal weights will be computed using the full sample. Supplied to subset, so the name of the dataset containing the subsetting variable does not need to be included if the subsetting variable is in new_data. See Examples.
method	the method used to compute the optimal weights. Allowable options include "chi-square" (the default), "quantile", or "max". "chi-square" and "quantile" involve averaging across the best several sets of weights, whereas "max" uses the singular best set of weights. Abbreviations allowed. See Details.

quantile	when method is "chi-square" or "quantile", the top quantile used to determine the best sets of weights to be included in the average to compute the optimal set of weights. Should be a number between 0 and 1, with a low value indicating that only the few top sets of weights will be used. Ignored when method = "max".
digits	the number of significant digits to print.
...	passed to print.default.

Details

calcGrpWprime() calculates an overall fitness score for each set of weights based on the average weighted fitness values of the indexed subgroup. The set of weights that optimizes this score is then produced as the weights defining the optimal adaptive landscape for that subgroup. The way the final set of weights is computed depends on the argument to method. When method = "max", the single best set of weights is used. However, often many of the upper sets of weights perform equally or nearly equally as well as the best set. It is instead recommended to use "quantile" or "chi-squared" methods. When method = "quantile", the top $X\%$ of weights are averaged to compute the optimal weights, where X corresponds to the value supplied to quantile. When method = "chi-square", the chi-squared value χ_i^2 is computed for each set of weights i as

$$\chi_i^2 = -2 \log \frac{Z_{max}}{Z_i}$$

where Z_{max} is the largest Z among the weights, and a p-value is computed for each χ_i^2 value using a χ^2 distribution with 2 d.f.; any set of weights with a p-value less than quantile is included to be averaged to compute the optimal set of weights.

Value

A grp_Wprime object, which contains the following components:

Zprime	a list containing the optimal weights and the Z value they yield (wn), and, if method is "chi-square" or "quantile", summary statistics about the best sets of weights used to compute the optimal weights, including the standard error (wn.se), standard deviation (wn.sd), and range (wn.range).
W	a matrix containing all sets of weights (i.e., those supplied to the grid_weights argument of calc_all_lscps()) along with the Z value each yields, ordered in descending order by the yielded Z value. When index is specified, the resulting Z values are computed only using the indexed subset.
Wprime	a wtd_lscp object containing the optimal weights (W) and the landscape grid and sample functional characteristcs weighted by the optimal weights.

References

Dickson, B. V., Clack, J. A., Smithson, T. R., & Pierce, S. E. (2021). Functional adaptive landscapes predict terrestrial capacity at the origin of limbs. Nature, 589(7841), 242-245.

Jones, K. E., Dickson, B. V., Angielczyk, K. D., & Pierce, S. E. (2021). Adaptive landscapes challenge the "lateral-to-sagittal"" paradigm for mammalian vertebral evolution. Current Biology, 31(9), 1883-1892.

See Also

calc_all_lscps for computing the landscapes which are to be optimized.

calcWprimeBy for finding optimal sets of weights for multiple subgroups defined by a subgrouping variable.

plot.grp_Wprime for plotting the resulting adaptive landscape.

Examples

```
data("warps")
data("turtles")

warps_fnc <- as_fnc_df(warps,
                       func.names = c("hydro", "fea"))

kr_surf <- krige_surf(warps_fnc, new_data = turtles)

grid_weights <- generate_weights(n = 3, data = kr_surf)

all_lscps <- calc_all_lscps(kr_surf,
                            grid_weights = grid_weights)

wprime_S <- calcGrpWprime(all_lscps,
                          index = Ecology == "S")
wprime_S
plot(wprime_S)
```

---

calcWprimeBy *Compute optimally weighted adaptive landscapes by subgroup*

Description

calcWprimeBy() computes the optimally weighted adaptive landscape by searching through the adaptive landscapes formed from sets of weights and performance surfaces, and finding the set of weights that yields the greatest overall (average) fitness value (Z) across subsets of a sample dataset.

Usage

```
calcWprimeBy(x, by, method = "chi-squared", quantile = 0.05)

## S3 method for class 'by_Wprime'
print(x,
      digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'by_Wprime'
summary(object, ...)

## S3 method for class 'summary.by_Wprime'
print(x,
      digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	for calcWprimeBy(), an all_lscps object; the output of a call to calc_all_lscps. for print.by_Wprime(), a by_Wprime object; the output of a call to calcWprimeBy(). for print.summary.by_Wprime(), a by_Wprime object; the output of a call to summary.by_Wprime().
by	a one-sided formula containing the grouping variable on the right hand side (e.g., ~g) or a vector containing the subgrouping variable. When supplied as a formula, the grouping variable must be present in the global environment or in the new_data component in the kriged_surfaces object originally supplied to calc_all_lscps().
method	the method used to compute the optimal weights. Allowable options include "chi-square" (the default), "quantile", or "max". "chi-square" and "quantile" involve averaging across the best several sets of weights, whereas "max" uses the singular best set of weights. Abbreviations allowed. See calcGrpWprime for details.
quantile	when method is "chi-square" or "quantile", the top quantile used to determine the best sets of weights to be included in the average to compute the optimal set of weights. Should be a number between 0 and 1, with a low value indicating that only the few top sets of weights will be used. Ignored when method = "max". See calcGrpWprime for details.
digits	the number of significant digits to print.
...	passed to print.default and print.table.
object	a by_Wprime object; the output of a call to calcWprimeBy().

Details

calcWprimeBy() splits the sample data based on the by variable and then calls calcGrpWprime on each subset. The main benefit of using calcWprimeBy() is that the subgrouping variable is part of the output object and therefore can be used in plotting using plot.by_Wprime.

Value

A by_Wprime object contaning the following components:

by	the subgrouping variable supplied to by, stored as a factor and with a "by_name" attribute containing the name of the variable.
grp_Wprimes	a list of grp_Wprime objects, one for each level of the subgrouping variable.

See Also

calc_all_lscps for computing the landscapes which are to be optimized.

calcGrpWprime for finding optimal sets of weights for a single subgroup.

plot.by_Wprime for plotting the resulting adaptive landscapes.

Examples

```
data("warps")
data("turtles")

warps_fnc <- as_fnc_df(warps,
                       func.names = c("hydro", "fea"))

kr_surf <- krige_surf(warps_fnc, new_data = turtles)

grid_weights <- generate_weights(n = 3, data = kr_surf)

all_lscps <- calc_all_lscps(kr_surf,
                            grid_weights = grid_weights)

wprime_Ecology <- calcWprimeBy(all_lscps, by = ~Ecology)
wprime_Ecology
summary(wprime_Ecology)
plot(wprime_Ecology)
```

`calc_all_lscps` *Calculate adaptive landscapes for a matrix of weights*

Description

`calc_all_lscps()` calculates adaptive landscapes from a set of kriged surfaces of functional characteristics and sets of weights for those characteristics.

Usage

```
calc_all_lscps(kr_data, grid_weights)
```

Arguments

`kr_data`	a `kriged_surfaces` object; the output of a call to `krige_surf`.
`grid_weights`	a `grid_weights` object; the output of a call to `generate_weights`.

Details

`calc_all_lscps()` computes a combined adaptive landscape for each of the supplied sets of weights. The optimal landscape overall or for certain subsets of the sample data can be found using `calcGrpWprime` or `calcWprimeBy`. `calc_lscp` can be used to extract the surface of the weighted functional characteristics for each set of weights (see Examples).

Value

An all_lscps object containing the following components:

dataframe	a list of the grid and new_data data frames stored in kr_data.
wtd_lscps	a list containing the weightred fitness values for each set of weights for the grid and new_data datasets. These are stored in matrices with a row for each data point in grid and new_data and a column for each set of weights.
grid_weights	the grid_weights object supplied to grid_weights.

See Also

calc_lscp for computing a single weighted landscape or extracting the weighted surface of functional characteristics for a single set of weights.

generate_weights for generating the required matrix of weights.

calcGrpWprime and calcWprimeBy for finding optimal sets of weights and adaptive landscapes for subgroups.

Examples

```
data("warps")
data("turtles")

warps_fnc <- as_fnc_df(warps,
                       func.names = c("hydro", "fea"))

kr_surf <- krige_surf(warps_fnc, new_data = turtles)

grid_weights <- generate_weights(n = 20, data = kr_surf)

all_lscps <- calc_all_lscps(kr_surf,
                            grid_weights = grid_weights)
all_lscps

# Extract the weighted surface for a single set
# of weights (here, the 6th set of weights)

grid_weights[6,]

wtd_lscp_6 <- calc_lscp(all_lscps, i = 6)
wtd_lscp_6

# This aligns with the weighted fitness value:
mean(all_lscps$wtd_lscps$new_data[,6])
```

calc_lscp *Calculate a single weighted adaptive landscape*

Description

calc_lscp() calculates a single weighted landscape from a set of kriged surfaces of functional characteristics and a set of weights for those characteristics. This landscape can then be plotted using plot.wtd_lscp. Additionally computes the fitness values for a sample of additional coordinates.

Usage

```
calc_lscp(data, weights, ...)

## S3 method for class 'kriged_surfaces'
calc_lscp(data, weights, ...)

## S3 method for class 'all_lscps'
calc_lscp(data, weights, i, ...)
```

Arguments

data	a kriged_surfaces or all_lscps object; the output of a call to krige_surf or calc_all_lscps, repsectively. If no new_data component is included in data, only the adaptive landscape will be produced.
weights	a vector of weights, one for each functional characteristic. These weights should be nonnegative and sum to 1.
i	when data is an all_lscps object, the index of the set of weights in the grid_weights object supplied to calc_all_lscps() to use to create the weighted landscape.
...	ignored.

Details

calc_lscp() operates on the kriged surfaces stored in data by multiplying the functional characteristic values of each point on the surface grid by the weights and computing the sum of those values to arrive at a "fitness" value that is represented by the maximum height of the combined adaptive landscape. When a new_data component is present in data (e.g., because a new_data argument was supplied to krige_surf() or data is the output of a call to krige_new_data()), the weighted fitness values will be computed for the coordinates in new_data as well.

Value

A wtd_lscp object, which contains the following components:

W	a named vector of the supplied weights

Wprime — a list containing the weighted grid and new_data components of data, where the values of the functional characteristics for each location on the surface are weighted by the supplied weights and an additiona column, Z, has been added containing the height of the adaptive landscape at that point.

See Also

plot.wtd_lscp for plotting the resulting weighted landscape.

generate_weights for generating a matrix of weights.

calc_all_lscps for computing weighted landscapes for a matrix of weights (i.e., rather than the single set of weights that can be used with calc_lscp). For finding an optimal set of weights, calc_all_lscps should be used, though it only produces the weighted fitness values for each set of weightd and not the weighted functional characteristic surfaces.

Examples

```
data("warps")

warps_fnc <- as_fnc_df(warps, func.names = c("hydro", "fea"))

kr_surf <- krige_surf(warps_fnc)

weights <- c(hydro = .5, fea = .5)

w_lscp <- calc_lscp(kr_surf, weights = weights)

plot(w_lscp)

# Adding new_data
data("turtles")
kr_surf <- krige_new_data(kr_surf, new_data = turtles)

w_lscp <- calc_lscp(kr_surf, weights = weights)
w_lscp
plot(w_lscp)

## See further use with calc_all_lscps()
## at help("calc_all_lscps")
```

---

generate_weights *Generate a matrix containing weight combinations*

Description

generate_weights() generates a matrix containing weight combinations for a set of variables such that each set of weights sums to 1. This can be supplied to calc_all_lscps to calculate fitness landscapes corresponding to a variety of possible sets of weights for weighting functional characteristics. The weights are generated by partitioning a weight of 1 across however many variables are requested in all possible ways.

Usage

```
generate_weights(step, n, data = NULL, nvar = NULL,
                 varnames = NULL, verbose = TRUE)
```

Arguments

step	numeric. The step size between weight partitions. Only one of step and n can be specified.
n	numeric. The number of weight partitions between 0-1. Only one of step and n can be specified.
data	an optional fnc_df (the output of as_fnc_df) or kriged_surfaces (the output krige_surf) object. The number of variabes and their names will be extracted from the data as the functional characteristics present in them.
nvar	the number of variables across which to allocate the weights. Ignored if data is not NULL. If nvar = NULL and varnames is supplied, the length of varnames will be used for nvar.
varnames	the names of the variables across which to allocate the weights. Ignored if data is not NULL. If varnames = NULL and nvar is supplied, the sequence from 1 to nar will be used for varnames.
verbose	whether to display a message noting the number of sets of weights created.

Details

generate_weights() works by fining all possible allocations of n objects into nvar bins. When step is supplied, n is computed as round(1/step), so the resulting weight partitions may not be exactly equal to step when its inverse is not an integer. The larger n is (or the smaller step) is, the more possible allocations will be produced (i.e., and the resulting object will have more rows). The output of generate_weights() can quickly become very large with increasing number of variables, and will make subsequent analyses slow. It is recommended to start with a large step size, or small n, and increment up.

Value

A grid_weights object, which is a matrix with a row for each each set of weights and a column for each variable over which the weights are allocated. The weights in each row will sum to 1.

Examples

```
# Allocating 10 partitions of .1 across 3 variables
wmat <- generate_weights(n = 10, nvar = 3)
head(wmat)

# Allocating 5 partitions of .2 across the 4 functional
# characteristics in the warps dataset
data("warps")

warps_fnc <- as_fnc_df(warps)
wmat <- generate_weights(n = 5, data = warps_fnc)
```

```
head(wmat)

# Using 'step' for the same result:
wmat <- generate_weights(step = .2, data = warps_fnc)
head(wmat)
```

`krige_surf` *Interpolate functional characteristics over a grid*

Description

`krige_surf()` performs kriging (i.e., interpolation) of one or more functional characteristics that are spatially distributed over a morphospace to create a smoothly kriged surface. Interpolated values can also be produced for a new dataset given their coordinates in morphological space.

Usage

```
krige_surf(fnc_df, grid = NULL, resample = 100,
           padding = 1.2, hull = TRUE, alpha = 1,
           new_data = NULL)

krige_new_data(x, new_data)
```

Arguments

`fnc_df`	a `fnc_df` object; the output of a call to `as_fnc_df`, which contains coordinates in morphological space and values of functional characteristics for the warps used to create the kriged surface.
`grid`	a matrix or data frame containing the grid of points over which the surface is to be interpolated. Should be the output of a call to `resample_grid`. If `NULL`, the grid will be formed by calling `resample_grid()` on the inputs.
`resample`	the number of points (or pixels) in the x and y dimensions over which to create the grid. Default is 100 for a kriged surface of 100x100=10,000 pixels. Passed to `resample_grid`. Ignored when `grid` is not `NULL`.
`padding`	a number representing how much to expand the grid beyond the ranges of the x- and y-coordinates. For example, `padding = 1.2` (the default) expands the grid by 20% of the coordinates' ranges in each direction. Must be a number greater than or equal to 1. Large numbers imply greater extrapolation, and whatever padding is added will be negated if `hull = TRUE`. Passed to `resample_grid`. Ignored when `grid` is not `NULL`.
`hull`	whether to restrict kriging to an alpha hull to prevent extrapolation beyond the coordinates present in `fnc_df`. Passed to `resample_grid`, which uses `alphahull::ahull`. Default is `TRUE`. If `FALSE`, kriging will take place over a rectangular grid that spans the boundaries of the coordinates in `fnc_df`. Ignored when `grid` is not `NULL`.

alpha	the alpha value used to create the alpha hull. Passed to resample_grid and eventually to alphahull::ahull. Ignored when grid is not NULL.
new_data	a dataset of coordinates for a new sample; the values of the functional characteristics for this sample will be interpolated using the kriged surface.
x	a "kriged_surfaces" object; the output of a call to krige_surf().

Details

krige_surf() implements the automap::autoKrige function on one or more spatially distributed traits to produce an interpolated (or extrapolated) surface of evenly spaced gridpoints. This is done by automatiically finding the best variogram fit for each of the non-corrdinate variables in fnc_df. For details on automatic variogram fitting, see automap::autoKrige.

Input data in fnc_df can be unevenly distributed (direct from speciments), or gridded (determined from evenly distributed hypothetical shapes) in morphospace. Trait data inputted directly from specimen measurements will be subject to error based on the how unevenly points are distributed, with high resoultion gridded datapoints producing the least potential reconstruction error (see Smith et al 2021).

By default krige_surf will create a hull to strictly prevent any extrapolation beyond the provided data. This will produce the most conservative landscapes. If hull is set to FALSE, then the function will reconstruct a rectangle determined by the XY coordinate ranges supplied in fnc_df. Padding will be applied by default (an extra 20%) as defined by padding, to expand the rectangle beyond the supplied points. Reconstructions without a hull would be most appropriate for trait data determined from evenely spaced hypothetical gridpoints. If grid is provided the function will strictly interpolate at these gridded points.

krige_new_data() adds a new data set to the supplied kriged_surfaces object and interpolates the values of the functional characteristics on the suppllied sample. This should only be used to add a new dataset to a kriged_surfaces object produced without new_data supplied or to replace an existing new_data component.

Value

An object of class kriged_surfaces containing the following components:

fnc_df	the original data frame of functional characteristics passed to fnc_df.
autoKrige	a named list of the kriged surfaces, one for each functional characteristic. Each surface is of class autoKrige, the output of the call to automap::autoKrige.
dataframes	a list of two data frames, grid and new_data. grid contains the coordinates of the kriged surface grid (in the x and y columns) as well as the interpolated values of the functional characteristics. new_data contains the sample coordinates supplied to new_data as well as the interpolated values of the functional characteristics for each sample. This second component is absent if new_data = NULL in the call to krige_surf().

For krige_new_data(), the original kriged_surfaces object, with the $dataframes$new_data component containing the sample coordinates supplied to new_data as well as the interpolated values of the functional characteristics for each sample.

References

Smith, S. M., Stayton, C. T., & Angielczyk, K. D. (2021). How many trees to see the forest? Assessing the effects of morphospace coverage and sample size in performance surface analysis. Methods in Ecology and Evolution, 12(8), 1411-1424.

See Also

`plot.kriged_surfaces` for plotting the kriged surfaces.

`as_fnc_df` for creating the input dataset.

`resample_grid` for creating the grid over which the kriging occurs prior to using `krige_surf`.

`automap::autoKrige` for the function that does the kriging.

Examples

```
data("warps")

warps_fnc <- as_fnc_df(warps)

grid <- resample_grid(warps)

kr_surf <- krige_surf(warps_fnc, grid = grid)
kr_surf

# Add new data
data("turtles")
kr_surf <- krige_new_data(kr_surf, new_data = turtles)
kr_surf
plot(kr_surf)

# Doing it all in one step:
kr_surf <- krige_surf(warps_fnc, new_data = turtles)
kr_surf
```

`lands.grp.test` *Significance tests between sets of weights*

Description

`lands.grp.test()` performs a statistical test for whether the optimal adaptive landscape for two subgroups are significantly different from each other. The p-value of the test is the proportion of weight sets that are shared between the two subgroups among their respective top weight sets. `multi.lands.grp.test()` performs this test for all pairs of subgroups.

Usage

```
lands.grp.test(grpa, grpb, method = "chi-squared",
               quantile = 0.05)

multi.lands.grp.test(x, method = "chi-squared",
                     quantile = 0.05)

## S3 method for class 'lands.grp.test'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'multi.lands.grp.test'
print(x, digits = max(3L, getOption("digits") - 3L),
      style = "matrix", ...)
```

Arguments

grpa, grpb	for lands.grp.test(), the two grp_Wprime objects containing the adaptive landscapes to be compared; these are the output of calls to calcGrpWprime.
x	for multi.lands.grp.test(), a by_Wprime object, the output of a call to calcWprimeBy. for print(), the output of a call to lands.grp.test() or multi.lands.grp.test().
method	the method used to determine which sets of weights are in the "best" sets of weights that are to be comapred between the two groups. Allowable options include "chi-squared" and "quantile". See calcGrpWprime for details.
quantile	the top quantile used to determine the best sets of weights to be included in the average to compute the optimal set of weights. Should be a number between 0 and 1, with a low value indicating that only the few top sets of weights will be used. See calcGrpWprime for details.
digits	the number of significant digits to print.
style	how to display the results of the pairwise tests; allowable options include "matrix" and "table". Abbreviations allowed.
...	passed to print.default.

Details

lands.grp.test() performs pairwise comparisons between two adaptive groups by comparing the number of shared landscapes n_{A+B} in the top percentile of each group with the total number of landscapes in this top percentile n_{total}. The probability $P(A = B)$ thus is calculated as:

$$P(A = B) = n_{A+B}/n_{total}$$

If method = "quantile" is used, then the top percentile is defined by quantile. If method = "chi-squared" is used, then the top percentile is calculated from the chi-squared value χ_i^2 as:

$$\chi_i^2 = -2 \log \frac{Z_{max}}{Z_i}$$

where Z_{max} is the largest Z among the weights, and a p-value is computed for each χ_i^2 value using a χ^2 distribution with 2 d.f.; any set of weights with a p-value less than `quantile` is included in the optimal set of weights.

`multi.lands.grp.test()` is a wrapper for `lands.grp.test()`, applying the function pairwise to all combinations of groups calculated by `calcWprimeBy`.

Value

For `lands.grp.test()`, a `lands.grp.test` object containing the following components:

`n.match`	the number of sets of weights that match between the two supplied subgroups
`p.val`	the p-value of the test, computed as the number of sets of weights that match divided by the number of sets of weights compared
`matching`	a matrix containing the sets of weights that match between the two subgroups
`method`	the argument supplied to `method`
`quantile`	the argument supplied to `quantile`

For `multi.lands.grp.test()`, a `multi.lands.grp.test` object containing the following components:

`res`	a data frame containing the results of the tests, with the columns `Group A` and `Group B` indicating the groups involved in the comparison, the column `Matches` containing the number of matching sets of weights in the comparison, and the column `p value` containing the p-value of the test.
`method`	the argument supplied to `method`
`quantile`	the argument supplied to `quantile`

For `print.multi.lands.grp.test()`, setting `style = "table"` prints the `res` component as-is; setting `style = "matrix"` creates a matrix where the p-values of the test are below the diagonal and the number of matches of the test are above the diagonal.

References

Jones, K. E., Dickson, B. V., Angielczyk, K. D., & Pierce, S. E. (2021). Adaptive landscapes challenge the "lateral-to-sagittal"" paradigm for mammalian vertebral evolution. Current Biology, 31(9), 1883-1892.

See Also

`calcGrpWprime` and `calcWprimeBy` for creatign the objects used as inputs to these functions

Examples

```
data("warps")
data("turtles")

warps_fnc <- as_fnc_df(warps,
                       func.names = c("hydro", "fea"))
```

```
kr_surf <- krige_surf(warps_fnc, new_data = turtles)

grid_weights <- generate_weights(n = 3, data = kr_surf)

all_lscps <- calc_all_lscps(kr_surf,
                            grid_weights = grid_weights)

# Comparing adaptive landscapes of Ecology groups S and M
wprime_S <- calcGrpWprime(all_lscps,
                          index = Ecology == "S")
wprime_M <- calcGrpWprime(all_lscps,
                          index = Ecology == "M")
lands.grp.test(wprime_S, wprime_M)

# Comparing adaptive landscapes of all Group subgroups
wprime_by_Group <- calcWprimeBy(all_lscps, by = ~Group)
tests <- multi.lands.grp.test(wprime_by_Group)
tests
print(tests, style = "table")
```

`plot.kriged_surfaces` *Plots Kriged surfaces of functional characteristics*

Description

`plot.kriged_surfaces()` produces spatial landscape plots of kriged surfaces produced by `krige_surf`.

Usage

```
## S3 method for class 'kriged_surfaces'
plot(x, alpha = 0.5, pt.col = "black",
     interpolate = TRUE, contour = TRUE, ...)
```

Arguments

`x`	a `kriged_surfaces` object; the output of a call to `krige_surf`.
`alpha, pt.col`	when a `new_data` component is present in `x`, the transparency (`alpha`) and color (`pt.col`) of the points plotted for the new samples.
`interpolate`	`logical`; whether to smooth the plot by interpolating across pixels in the grid. Passed to `ggplot2::geom_raster`.
`contour`	`logical`; whether to add contour lines to the plot to illustrate changes in the fitness landscape.
`...`	ignored.

Details

`plot.kriged_surfaces()` is a wrapper for **ggplot2** raster plotting functions. For more precise control of raster plotting see `ggplot2::geom_raster`.

Value

A ggplot object, which can be further manipulated using **ggplot2** functionality.

See Also

ggplot2::ggplot, ggplot2::geom_raster, and ggplot2::geom_contour for the underlying plotting functions. See also sp::spplot for alternative plotting functions.

krige_surf for generating the kriged surfaces. krige_new_data for adding a new_data component to an existing kriged surface before plotting.

Examples

```
# See examples at help("krige_surf")
```

---

plot.wtd_lscp *Plot Adaptive Landscapes*

Description

These plot plot methods plot an adaptive landscape, a weighted combination of functional surfaces. These landscape arise from calls to calc_lscp, calc_all_lscps, calcGrpWprime, and calcWprimeBy.

Usage

```
## S3 method for class 'wtd_lscp'
plot(x, alpha = 1, pt.col = "black",
     interpolate = TRUE, contour = TRUE, ...)
## S3 method for class 'grp_Wprime'
plot(x, alpha = 1, pt.col = "black",
     interpolate = TRUE, contour = TRUE, ...)
## S3 method for class 'by_Wprime'
plot(x, level, ncol = 1, alpha = 1,
     pt.col = "black", interpolate = TRUE, contour = TRUE,
     ...)
```

Arguments

x	a wtd_lscp, grp_Wprime, or by_Wprime object, the output of a call to calc_lscp, calcGrpWprime, or calcWprimeBy, respectively.
alpha	the transparency of the points for the data sample. A number between 0 (fully transparent) and 1 (fully opaque). Passed to ggplot2::geom_point.
pt.col	the color of the points for the data sample. Passed to ggplot2::geom_point.
interpolate	whether to interpolate across pixels in the grid. Passed to ggplot2::geom_raster.
contour	whether to display contours in the grid.

level	which level of the by (subgrouping) variable to be plotted. If missing, all will be plotted.
ncol	when multiple subgroups are plotted, in how many columns should the plots be arranged.
...	ignored.

Details

These plotting functions are wrappers for ggplot2 raster plotting functions. For more precise control of raster plotting see ggplot2::geom_raster.

Value

A ggplot object that can be further adjusted using functions from **ggplot2**.

See Also

calc_lscp, calc_all_lscps, calcGrpWprime, and calcWprimeBy for the functions used to create the objects that are plotted

plot.kriged_surfaces for plotting functional surfaces prior to combining them into an adaptive landscape.

ggplot2::geom_raster, ggplot2::geom_point, and ggplot2::geom_contour for the underlying plotting functions.

Examples

```
data("warps")
data("turtles")

warps_fnc <- as_fnc_df(warps, func.names = c("hydro", "fea"))

kr_surf <- krige_surf(warps_fnc, new_data = turtles)

weights <- c(hydro = .5, fea = .5)

w_lscp <- calc_lscp(kr_surf, weights = weights)

plot(w_lscp)
plot(w_lscp, countour = FALSE, pt.col = "white")

# See help("calc_lscp"), help("calcGrpWprime"), and
# help("calcWprimeBy") for examples when used with
# those functions
```

resample_grid *Create a full grid from a set of coordinates*

Description

`resample_grid()` creates a rectangular grid around supplied coordinates by resampling evenly spaced points between the minimum and maximum values of each coordinate dimension. The grid can optionally be reduced to a convex or concave hull around the supplied coordinates.

Usage

```
resample_grid(coords2D, resample = 100, padding = 1.2,
              hull = TRUE, alpha = 1, plot = FALSE)
```

Arguments

`coords2D`	a 2-column matrix data frame of coordinates with the x-coordinates in the first column and the y-coordinates in the second column. The ranges of each column will be used to create the resampled grid.
`resample`	the number of points (or pixels) in the x and y dimensions over which to create the grid. Default is 100 for a kriged surface of 100x100=10,000 pixels.
`padding`	a number representing how much to expand the grid beyond the ranges of the x- and y-coordinates. For example, `padding = 1.2` (the default) expands the grid by 20% of the coordinates' ranges in each direction. Must be a number greater than or equal to 1. Large numbers imply greater extrapolation, and whatever padding is added will be negated if `hull = TRUE`.
`hull`	whether to restrict the grid to a convex or concave hull. Default is `TRUE`.
`alpha`	when `hull = TRUE`, the alpha value used to create the hull. Passed to `concaveman::concaveman`.
`plot`	when `hull = TRUE`, whether to plot the resulting hull overlayed over the original grid. Default is `TRUE`.

Value

A data frame with two columns, `x` and `y`, containing the resampled coordinate grid. When `hull = TRUE`, any points not in the hull will be absent.

See Also

`krige_surf`, which uses `resample_grid` for kriging.

`concaveman::concaveman` for convex and concave hulls.

`alphahull::ahull` and `alphahull::inahull` for creating an alpha hull.

Examples

```
data("warps")

warps_fnc <- as_fnc_df(warps)

# Alpha hull with plot to see the hull
grid <- resample_grid(warps_fnc[c("x", "y")],
                      hull = TRUE, plot = TRUE)
str(grid)
```

`turtles` *Turtle Humeri*

Description

A dataset containing a sample of 40 turtle humeri used in Dickson and Pierce (2019).

Usage

```
data("turtles")
```

Format

A data frame with 40 observations on the following 4 variables.

`x` the first axis of shape variation as determiend by a between-groups principal components analysis

`y` the second axis of shape variation as determiend by a between-groups principal components analysis

`Group` the locomotor ecologies of the turtles

`Ecology` the three ecological groups as determined by a Procrustes ANOVA; "M" (marine), "S" (semiaquatic), and "T" (terrestrial)

Source

Dickson, B.V. and Pierce, S.E. (2019), Functional performance of turtle humerus shape across an ecological adaptive landscape. Evolution, 73: 1265-1277. doi:10.1111/evo.13747

warps *Simulated Shape Warps*

Description

Trait data for simulated shape warps of turtle humeri used to study the morphological evolution of turtles in Dickson and Pierce (2019). The morphospace was defined by a geometric morphometric analyises of 1028 psuedolandmarks on 40 turtle humeri. Hypothetical shape warps were then produced on a 4x6 grid accross morphospace. For each shape warp, four functional traits were measured, corresponding to a locomotory performance trait: stress under simulated load (strength), bone curvature (stride length), muscular mechanical advantage (mechanical advantage), and frontal area (hydrodynamics).

Usage

```
data("warps")
```

Format

A data frame with 24 observations on the following 6 variables.

`x` the first axis of shape variation as determiend by a between-groups principal components analysis of the `turtles` dataset

`y` the second axis of shape variation as determiend by a between-groups principal components analysis of the `turtles` dataset

`hydro` hydrodynamics

`curve` stride length

`mech` mechanical advantage

`fea` strength (assessed using finite element analysis)

Source

Dickson, B.V. and Pierce, S.E. (2019), Functional performance of turtle humerus shape across an ecological adaptive landscape. Evolution, 73: 1265-1277. doi:10.1111/evo.13747

Index