

# Package ‘Morphoscape’

May 18, 2023

**Type** Package

**Title** Computation and Visualization of Adaptive Landscapes

**Version** 1.0.2

**Description** Implements adaptive landscape methods first described by Polly et al. (2016) <[doi:10.1080/02724634.2016.1111225](https://doi.org/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 (>= 4.2.0)

**Imports** concaveman, ggplot2, spatial, sp, automap, scales, viridisLite, alphahull

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**URL** <https://blakedickson.github.io/Morphoscape/>

**RoxygenNote** 7.2.3

**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 <b.dickson@unsw.edu.au>

**Repository** CRAN

**Date/Publication** 2023-05-18 09:20:06 UTC

## R topics documented:

as_fnc_df . . . . .	2
calcGrpWprime . . . . .	3
calcPoly . . . . .	5

calcWprimeBy . . . . .	8
calc_all_lscps . . . . .	10
calc_lscp . . . . .	12
generate_weights . . . . .	13
krige_surf . . . . .	15
lands.grp.test . . . . .	17
lscp_operations . . . . .	20
plot.krige_surfaces . . . . .	22
plot.wtd_lscp . . . . .	23
resample_grid . . . . .	25
turtles . . . . .	26
warps . . . . .	27

<b>Index</b>	<b>29</b>
--------------	-----------

---

as_fnc_df	<i>Convert a data frame to a fnc_df</i>
-----------	---

---

## Description

as\_fnc\_df() converts a data frame containing coordinates and functional characteristics in a morphological space to a fnc\_df object for use in later functions, 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 that correspond to functional characteristics. These characteristics 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 indices, 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 <code>print.default</code> .

### 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  $\chi_i^2$  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  $\chi_i^2$  value using a  $\chi^2$  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 ( <code>wn</code> ), and, if <code>method</code> is "chi-square" or "quantile", summary statistics about the best sets of weights used to compute the optimal weights, including the standard error ( <code>wn.se</code> ), standard deviation ( <code>wn.sd</code> ), and range ( <code>wn.range</code> ).
W	a matrix containing all sets of weights (i.e., those supplied to the <code>grid_weights</code> argument of <code>calc_all_lscps()</code> ) along with the Z value each yields, ordered in descending order by the yielded Z value. When <code>index</code> is specified, the resulting Z values are computed only using the indexed subset.
Wprime	a <code>wtd_lscp</code> object containing the optimal weights ( <code>W</code> ) and the landscape grid and sample functional characteristics 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)
```

---

calcPoly

*Calculate polynomial fits over a surface*

---

## Description

calcPoly calls on the `spatial` package to fit rectangular spatial polynomial surface models by least-squares, or GLS. These methods allow the user to test whether data have spatial trends in morphospace. Outputs are a polynomial trend surface, and ANOVA table for the model fit. `multiPoly` applies calcPoly to a `fnc_df` with outputs for each trait. For more extensive documentation for model fitting see the `spatial` package.

**Usage**

```
calcPoly(fnc, npoly = 3, fnc.name = NULL,
         gls.covmod = list(covmod = expcov, d = 0.7, alpha = 0, se = 1),
         pad = 1.2, resample = 100, range = NULL, verbose = FALSE)

multiPoly(fnc_df, npoly = 3, ...)
```

**Arguments**

fnc	an XYZ dataframe or matrix of a spatially distributed trait.
fnc_df	a functional dataframe from <code>as_fnc_df</code> with colnames corresponding to X,Y and trait names.
npoly	singular numeric. Degree of polynomial to fit ranging from 1-4. For <code>multiPoly</code> this can also be a vector with length equal to the number of traits in order to specify the degree of polynomial to apply to each trait.
gls.covmod	Optional list of arguments to pass to <code>surf.gls</code> if fitting by generalized least-squares is desired. Defaults to NULL, and fitting is performed by least-squares. See <a href="#">surf.gls</a> and <a href="#">expcov</a> documentation for a full list of arguments and usage.
fnc.name	Optional specification of the trait name. Defaults to NULL, and will use column names instead.
pad	Degree by which to extrapolate input data. Defaults to 1.2.
resample	Resampling density. Corresponds to the number of points calculated along both X and Y axes. Defaults to 100. If no resampling is desired, set <code>resample = NULL</code>
range	Optional. Manually set X and Y ranges. Input is a 2x2 matrix with rows corresponding to X and Y ranges respectively.
verbose	Optional. Logical. If TRUE, will print ANOVA tables.
...	Arguments to pass onto <code>calcPoly</code> when using <code>multiPoly</code>

**Details**

Fits polynomial trend surfaces using the ‘spatial’ package. First, an `npoly` polynomial trend surface is fit by least squares using [surf.ls](#) or generalized least-squares by [surf.gls](#). GLS is fit by one of three covariance functions, exponential (`expcov`), gaussian (`gaucov`) or spherical (`sphercov`) and requires additional parameters to be passed as a list through `gls.covmod` (see examples). For a full description of arguments and usage see [surf.gls](#) and [expcov](#) documentation.

The surface is then evaluated using [trmat](#) within limits set by input data, or manually using `range`.

**Value**

An object of class `poly_surf`, or `multi_surf` with the following components:

fnc.name	name of trait
poly	Polynomial trend fit output from <a href="#">surf.ls</a>
surface	Evaluated trend surface output from <a href="#">trmat</a>
grid	Expanded surface in long XZY dataframe format
peak	Coordinates and height of the peak of the surface



```
summary(polysurf)

## End(Not run)
#
# Adjust polynomial degree

multiPoly(warps_fnc, npoly = 2)

# Specify multiple degrees

multi_poly <- multiPoly(warps_fnc, npoly = c(2,3,4,3))

## Not run:
summary(polysurf)

## End(Not run)
plot(polysurf)
```

---

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 <a href="#">calcGrpWprime</a> 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 <a href="#">calcGrpWprime</a> for details.
digits	the number of significant digits to print.
...	passed to <a href="#">print.default</a> and <a href="#">print.table</a> .
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 containing 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	<i>Calculate adaptive landscapes for a matrix of weights</i>
----------------	--

---

## 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, file = NULL)
```

## Arguments

kr_data	a kriged_surfaces object; the output of a call to <a href="#">krige_surf</a> .
grid_weights	a grid_weights object; the output of a call to <a href="#">generate_weights</a> .
file	the path of a file to save the resulting output object, which may be quite large. The file path should contain an .rds or .rdata extension, which will be saved using <a href="#">saveRDS</a> or <a href="#">save</a> , respectively. See Details on how to load these files after saving them.

## 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).

Because the resulting objects are so large, it can be a good idea to save them after creation, which can be done automatically using the file argument. If the supplied file extension is .rds, [saveRDS](#) will be used to save the object to the supplied file path, and the file can be loaded using [readRDS](#). If the supplied file extension is .RData, [save](#) will be used to save the object to the supplied file path, and the file can be loaded using [load](#).

**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 weighted 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 <code>krige_surf</code> or <code>calc_all_lscps</code> , respectively. 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 <code>calc_all_lscps()</code> 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 additional 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 weights 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 <a href="#">as_fnc_df</a> ) or kriged_surfaces (the output of <a href="#">krige_surf</a> ) object. The number of variables 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 nvar will be used for varnames.
verbose	whether to display a message noting the number of sets of weights created.

**Details**

generate\_weights() works by finding 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 = NULL,
           alpha = 1, new_data = NULL)
```

```
krige_new_data(x, new_data)
```

## Arguments

<code>fnc_df</code>	a <code>fnc_df</code> object; the output of a call to <code>as_fnc_df</code> , which contains coordinates in morphological space and values of functional characteristics for the warps used to create the kriged surface.
<code>grid</code>	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 <code>resample_grid</code> . If <code>NULL</code> , the grid will be formed by calling <code>resample_grid()</code> on the inputs.
<code>resample</code>	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 <code>resample_grid</code> . Ignored when <code>grid</code> is not <code>NULL</code> .
<code>padding</code>	a number representing how much to expand the grid beyond the ranges of the x- and y-coordinates. For example, <code>padding = 1.2</code> (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 <code>hull = TRUE</code> . Passed to <code>resample_grid</code> . Ignored when <code>grid</code> is not <code>NULL</code> .
<code>hull</code>	method to restrict kriging to an alpha hull to prevent extrapolation beyond the coordinates present in <code>fnc_df</code> . Passed to <code>resample_grid</code> , which uses <code>alphahull</code> or <code>concaveman</code> packages. Default is <code>alphahull::ahull</code> . If no hull is desired set <code>hull = NULL</code> . If <code>NULL</code> , kriging will take place over a rectangular grid that spans the boundaries of the coordinates in <code>fnc_df</code> . Ignored when <code>grid</code> is not <code>NULL</code> .

alpha	the alpha value used to create the alpha hull. Passed to <code>resample_grid</code> and eventually to <code>alphahull::ahull</code> . 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 <code>krige_surf()</code> .

## 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 automatically finding the best variogram fit for each of the non-coordinate variables in `fnc_df`. For details on automatic variogram fitting, see `automap::autoKrige`.

Input data in `fnc_df` can be unevenly distributed (direct from specimens), 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 resolution 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 evenly 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 supplied 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:

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

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, hull = "concaveman", plot = TRUE)

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:
## Not run:
kr_surf <- krige_surf(warps_fnc, new_data = turtles, hull = "alphahull")
kr_surf

## End(Not run)

# No hull and padding
kr_surf <- krige_surf(warps_fnc, new_data = turtles, hull = NULL, padding = 1.2)
plot(kr_surf)
```

**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 <a href="#">calcGrpWprime</a> .
x	for multi.lands.grp.test(), a by_Wprime object, the output of a call to <a href="#">calcWprimeBy</a> . 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 compared between the two groups. Allowable options include "chi-squared" and "quantile". See <a href="#">calcGrpWprime</a> 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 <a href="#">calcGrpWprime</a> 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 <a href="#">print.default</a> .

**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  $\chi_i^2$  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  $\chi_i^2$  value using a  $\chi^2$  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")
```

---

lscp\_operations

*Simple Operations on Spatial Data*

---

## Description

Perform simple operations (sum, sub, mult, div) on one or multiple landscapes (see details for use cases)

## Usage

```
sum_lscps(lscps, num = NULL, average = TRUE)
mult_lscps(lscps, num = NULL)
sub_lscps(lscps, binary = FALSE)
div_lscps(lscps, binary = FALSE)
```

## Arguments

**lscps** A named list containing datasets of spatial data to apply operations (see details).

num	Optional. Defaults to NULL. A vector containing a single numeric scalar or numeric vector with length = lscps to scale by when using <code>sum_lscps</code> and <code>mult_lscps()</code> . If NULL, these functions will operate between lscps. If num is provided, these functions will operate between lscps and num.
binary	If subtracting or dividing landscapes, binarize result to obtain logical [0,1] result (see details for use case)
average	if summation is performed, should the result be averaged for the number of landscapes

### Details

Simple operations are applied to one or more landscapes depending on use case. Spatial datasets can be supplied in an XYZ dataframe, or as any landscape output from Morphoscape.

sum - sum a single spatial dataset/landscape with a single scalar ( $lscp1 + num1$ ); sum two or more spatial datasets/landscapes together ( $lscp1 + lscp2 \dots lscpN$ ); or sum multiple landscapes with multiple scalars ( $(lscp1, lscp2 \dots lscpN) + (num1, num2, \dots, numN)$ )

mult - multiply a single spatial dataset/landscape with a single scalar ( $lscp1 * num1$ ); multiply two or more spatial datasets/landscapes together ( $lscp1 * lscp2 \dots lscpN$ ); or multiply multiple landscapes with multiple scalars ( $(lscp1, lscp2 \dots lscpN) * (num1, num2, \dots, numN)$ )

sub - subtract one spatial dataset/landscape from another ( $lscp1 - lscp2$ ). If numeric subtraction is desired, use `sum_lscps` with negative num values.

div - divide one spatial dataset by another ( $lscp1 / lscp2$ ).

`sub_lscps()` and `div_lscps` can be used to construct transition landscapes per (Dickson et al 2020) which can compare performance between two adaptive regimes. If `binary = T` is used, the result will be a spatial representation of which parent landscape dominates. However, it is recommended to use `trans_lscps` or `adpt_regions` to calculate transitional landscapes or adaptive regions (not yet implemented).

### Value

An object of class "combined.surface" containing XYZ spatial data.

### Author(s)

Blake V. Dickson

### See Also

[krige\\_surf](#), [calcGrpWprime](#), [calc\\_lscp](#), [calcPoly](#), [trans\\_lscps](#), [adpt\\_regions](#)

### Examples

```
require("Morphoscape")

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

fnc_df <- as_fnc_df(warps,
```

```

func.names = c("hydro", "curve", "mech", "fea")

kr_surf <- krige_surf(fnc_df, new_data = turtles)

grid_weights <- generate_weights(n = 10, data = kr_surf)

all_lscps <- calc_all_lscps(kr_surf,
                           grid_weights = grid_weights)

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

wprime_T <- calcGrpWprime(all_lscps,
                          index = Ecology == "T")

lscps <- list(wprimeS = wprime_S, wprime_T = wprime_T)

# summing multiple landscapes together

summed_surfs <- sum_lscps(lscps, average = TRUE)

# summing landscapes by one or more numeric scalars

summed_surfs <- sum_lscps(lscps, num = c(1.5, -1.15)) # multiple numeric, with subtraction

# multiplying multiple landscapes together
mult_surfs <- mult_lscps(lscps) # multiply landscapes together

# multiplying landscapes by one or more numeric scalars
mult_surfs <- mult_lscps(lscps, num = 2) # apply numeric multiplier to all landscapes
mult_surfs <- mult_lscps(lscps, num = c(1.2, 0.8)) # apply numeric elements to each landscape

# subtract or divide two landscapes

sub_surf <- sub_lscps(lscps)
div_surf <- div_lscps(lscps)

# with binary result

sub_surf <- sub_lscps(lscps, binary = TRUE)
div_surf <- div_lscps(lscps, binary = TRUE)

```

---

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 <a href="#">krige_surf</a> .
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 <a href="#">ggplot2::geom_raster</a> .
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::splot](#) 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 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 <code>wtd_lscp</code> , <code>grp_Wprime</code> , or <code>by_Wprime</code> object, the output of a call to <code>calc_lscp</code> , <code>calcGrpWprime</code> , or <code>calcWprimeBy</code> , respectively.
alpha	the transparency of the points for the data sample. A number between 0 (fully transparent) and 1 (fully opaque). Passed to <code>ggplot2::geom_point</code> .
pt.col	the color of the points for the data sample. Passed to <code>ggplot2::geom_point</code> .
interpolate	whether to interpolate across pixels in the grid. Passed to <code>ggplot2::geom_raster</code> .
contour	whether to display contours in the grid.
level	which level of the <code>by</code> (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.krige_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 = NULL, 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 is specified.
hull	method to restrict the grid to an alpha hull using alphahull] or concaveman packages. Default is 'NULL' and no hull will be calculated.

alpha	when hull != NULL, the alpha value used to create the hull. Passed to <a href="#">ahull</a> or <a href="#">concaveman</a> .
plot	Logical. When hull is specified, 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 is specified, any points not in the hull will be absent.

**See Also**

[krige\\_surf](#), which uses `resample_grid` for kriging.  
[ahull](#) and [inahull](#), or [concaveman](#) for creating the hull.

**Examples**

```
data("warps")

warps_fnc <- as_fnc_df(warps)

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

## Not run:
# Alpha hull with plot to see the hull
grid <- resample_grid(warps_fnc[c("x", "y")],
                      hull = "alphahull", plot = TRUE)

## End(Not run)
```

---

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 determined by a between-groups principal components analysis

y the second axis of shape variation as determined 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 analysis of 1028 pseudolandmarks on 40 turtle humeri. Hypothetical shape warps were then produced on a 4x6 grid across 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 determined by a between-groups principal components analysis of the `turtles` dataset

y the second axis of shape variation as determined 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](https://doi.org/10.1111/evo.13747)

# Index

- \* **datasets**
  - turtles, 26
  - warps, 27
  
- ahull, 26
- alphahull::ahull, 15, 16
- as.data.frame, 3
- as\_fnc\_df, 2, 14, 15, 17
- attribute, 3
- automap::autoKrige, 16, 17
  
- calc\_all\_lscps, 3, 5, 8, 9, 10, 12, 13, 23, 24
- calc\_lscsp, 10, 11, 12, 21, 23, 24
- calcGrpWprime, 3, 9–11, 18, 19, 21, 23, 24
- calcPoly, 5, 21
- calcWprimeBy, 5, 8, 10, 11, 18, 19, 23, 24
- concaveman, 26
  
- div\_lscps (lscp\_operations), 20
  
- expcov, 6, 7
  
- fnc\_df (as\_fnc\_df), 2
  
- generate\_weights, 10, 11, 13, 13
- ggplot2::geom\_contour, 23, 24
- ggplot2::geom\_point, 24
- ggplot2::geom\_raster, 23, 24
- ggplot2::ggplot, 23
  
- inahull, 26
  
- krige\_new\_data, 23
- krige\_new\_data (krige\_surf), 15
- krige\_surf, 2, 3, 10, 12, 14, 15, 21–23, 26
- kriged\_surfaces (krige\_surf), 15
  
- lands.grp.test, 17
- load, 10
- lscp\_operations, 20
  
- mult\_lscps (lscp\_operations), 20
  
- multi.lands.grp.test (lands.grp.test), 17
- multiPoly (calcPoly), 5
  
- plot.by\_Wprime, 9
- plot.by\_Wprime (plot.wtd\_lscsp), 23
- plot.grp\_Wprime, 5
- plot.grp\_Wprime (plot.wtd\_lscsp), 23
- plot.kriged\_surfaces, 17, 22, 24
- plot.wtd\_lscsp, 12, 13, 23
- print.by\_Wprime (calcWprimeBy), 8
- print.default, 4, 9, 18
- print.grp\_Wprime (calcGrpWprime), 3
- print.lands.grp.test (lands.grp.test), 17
- print.multi.lands.grp.test (lands.grp.test), 17
- print.summary.by\_Wprime (calcWprimeBy), 8
- print.table, 9
  
- readRDS, 10
- resample\_grid, 15–17, 25
  
- save, 10
- saveRDS, 10
- sp::spplot, 23
- sub\_lscps (lscp\_operations), 20
- subset, 3
- sum\_lscps (lscp\_operations), 20
- summary.by\_Wprime (calcWprimeBy), 8
- surf.gls, 6, 7
- surf.ls, 6, 7
  
- trmat, 6, 7
- turtles, 26, 27
  
- warps, 27
- wtd\_lscsp, 4
- wtd\_lscsp (calc\_lscsp), 12