Package: MorphoRegions (via r-universe)

October 21, 2024

Type Package

Title Analysis of Regionalization Patterns in Serially Homologous Structures

Version 0.1.0

Description Computes the optimal number of regions (or subdivisions) and their position in serial structures without a priori assumptions and to visualize the results. After reducing data dimensionality with the built-in function for data ordination, regions are fitted as segmented linear regressions along the serial structure. Every region boundary position and increasing number of regions are iteratively fitted and the best model (number of regions and boundary positions) is selected with an information criterion. This package expands on the previous 'regions' package (Jones et al., Science 2018) with improved computation and more fitting and plotting options.

License GPL (>= 2)

Encoding UTF-8

Depends R (>= 4.0.0)

Imports stats, parallel, utils, grDevices, RColorBrewer (>= 1.1-3), cluster (>= 2.1.4), scales (>= 1.3.0), ggplot2 (>= 3.5.1), chk (>= 0.9.0), pbapply (>= 1.7-2)

Suggests viridisLite, patchwork (>= 1.1.0), knitr, rmarkdown

VignetteBuilder knitr

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

URL https://aagillet.github.io/MorphoRegions/, https://github.com/aagillet/MorphoRegions

BugReports https://github.com/aagillet/MorphoRegions/issues

Repository https://aagillet.r-universe.dev

RemoteUrl https://github.com/aagillet/morphoregions

RemoteRef HEAD

RemoteSha 37cfab73b032cbae7d61396e51e7b658393424e8

Contents

`alligator` *Measurements from the vertebral column of an alligator*

Description

Linear and angular measurements from Alligator mississipiensis MCZ 81457

Usage

```
alligator
```

Format

A matrix with 22 vertebrae and 19 measurements. Column 1, vertebra, is the positional information.

calcBPvar *Calculate variability of breakpoints*

Description

`calcBPvar()` computes an estimate of the variability of the breakpoints for a given number of regions. This involves computing the weighted mean and standard deviation of each breakpoint using Akaike weights.

Usage

```
calcBPvar(regions_results, noregions, pct = 0.05, criterion = "aic")
```

Arguments

`regions_results`	a `regions_results` object; the output of a call to `calcregions()` or `addregions()`.
`noregions`	the number of regions for which the weighted mean and standard deviation are to be computed.
`pct`	the proportion of best model to keep from the original total number of possible models
`criterion`	string; the criterion used to compute the weights. Allowable options include `"aic"` and `"bic"`. Abbreviations allowed.

Value

A `regions_BPvar` object, which has two components:

- `WeightedBP` is a matrix containing the weighted mean and standard deviation of each breakpoint
- `BestModels` is a data frame containing the models used to compute the weighted breakpoint statistics and the weights each one is given.

See Also

`calcregions()` for fitting segmented regression models to all combinations of breakpoints.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)
```

```
# Fit segmented regression models for 1 to 7 regions
# using PCOs 1 to 4 and a continuous model with a
# exhaustive search
regionresults <- calcregions(alligator_PCO,
                             scores = 1:4,
                             noregions = 7,
                             minvert = 3,
                             cont = TRUE,
                             exhaus = TRUE,
                             verbose = FALSE)

# Compute Akaike-weighted location and SD of optimal
# breakpoints using top 10% of models with 4 regions
calcBPvar(regionresults, noregions = 4,
          pct = .1, criterion = "aic")
```

`calcmodel` *Calculate results of a single segmented regression model*

Description

`calcmodel()` fits a multivariate segmented regression model using the supplied PCOs and breakpoints.

Usage

```
calcmodel(x, scores, bps, cont = TRUE)
```

Arguments

`x`	a `regions_pco` object; the output of a call to `svdPCO()`.
`scores`	`numeric`; the indices of the PCO scores to use as the outcomes in fitting the model (e.g., `1:4` to use the first four scores).
`bps`	`numeric`; the indices of the breakpoints to use in fitting the model. To request a model with no breakpoints, set to `NA`.
`cont`	`logical`; whether to fit a model that is continuous (`TRUE`) or discontinuous (`FALSE`) at the breakpoints. Default is `TRUE`. Ignored when `bps` is `NA`.

Value

A `regions_results_single` object, which contains the results of the model (breakpoints and RSS of each PCO and overall) and model support statistics.

See Also

`calcregions()` and `addregions()` for computing all possible models instead of just a single one; `plotsegreg()`, for which the `plot` method is an alias, for plotting the fitted regression lines; `modelsupport()` for interpreting the model support statistics.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)

# Calculate a single segmented regression model
# using first 2 PCOs and a discontinuous model
regionsmodel <- calcmodel(alligator_PCO,
                          scores = 1:3,
                          bps = c(8, 16, 21),
                          cont = FALSE)

regionsmodel

#Evaluate performance (R2) on that model
modelperf(regionsmodel)

#Plot model results:
plotsegreg(regionsmodel, scores = 1:3)
```

`calcregions` *Fit segmented regression models for all combinations of breakpoints*

Description

`calcregions()` enumerates all possible combinations of breakpoints to fit multivariate segmented regression models. `addregions()` adds models with additional numbers of regions to the resulting output object. `ncombos()` computes an upper bound on the number of breakpoint combinations that will be tested.

Usage

```
calcregions(
  pco,
  scores,
  noregions,
  minvert = 3,
  cont = TRUE,
  exhaus = TRUE,
  includebp = NULL,
  omitbp = NULL,
  ncombos_file_trigger = 1e+07,
  temp_file_dir = tempdir(TRUE),
  cl = NULL,
```

```
  verbose = TRUE
)

addregions(
  regions_results,
  noregions,
  exhaus = TRUE,
  ncombos_file_trigger = 1e+07,
  temp_file_dir = tempdir(TRUE),
  cl = NULL,
  verbose = TRUE
)

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

ncombos(pco, noregions, minvert = 3, includebp = NULL, omitbp = NULL)
```

Arguments

pco	a regions_pco object; the output of a call to svdPCO().
scores	numeric; the indices of the PCO scores to use as the outcomes in fitting the models (e.g., 1:4 to use the first four scores). Can also be the output of a call to PCOselect().
noregions	numeric; for calcregions(), the maximum number of regions for which models are fit (e.g, 4 to request models with 1 to 4 regions); for addregions(), a vector containing the numbers of regions to add (e.g., 5:6 to request models with 5 and 6 regions); for ncombos(), a vector containing the numbers of regions to check.
minvert	numeric; the minimum number of vertebrae allowed in each region. Default is 3.
cont	logical; whether to fit models that are continuous (TRUE) or discontinuous (FALSE) at the breakpoints. Default is TRUE.
exhaus	logical; whether to fit all possible models (TRUE) or use heuristics to reduce the number of models fit (FALSE). Default is TRUE. See Details. Setting to FALSE can reduce the size of the resulting object.
includebp	an optional vector of vertebrae that must be included in any tested set of breakpoints, e.g., if it is known that two regions are divided at that vertebra. includebp does not have to obey the minvert rules, but a warning will be thrown if it doesn't.
omitbp	an optional vector of vertebrae to be omitted from the list of possible breakpoints, e.g., if it is known that two adjacent vertebrae belong to the same region.
ncombos_file_trigger	numeric; when the number of eligible combinations of breakpoints exceeds this number, the problem will be split into smaller problems, with the results of each stored in its own temporary file in the directory supplied to temp_file_dir

before being re-read into memory. The primary purpose of this is to preserve memory when exhaus = FALSE by delegating storage of the results to disk instead of RAM.

temp_file_dir — string; the directory where the temporary files will be saved (and then deleted) when the number of breakpoint combinations exceeds ncombos_file_trigger. Default is the directory produced by tempdir(), but it is much safer to provide your own directory, which must already exist on your machine. See Details.

cl — a cluster object created by parallel::makeCluster(), an integer to indicate number of child-processes (integer values are ignored on Windows) for parallel evaluations, or "future" to use a future backend. NULL (the default) refers to sequential evaluation (no parallelization). See pbapply::pbapply() for details.

verbose — logical; whether to print information about the fitting process, including a progress bar. Default is TRUE.

regions_results, object — a regions_results object; the output of a call to calcregions() or addregions().

... — ignored.

Details

calcregions() enumerates all possible combinations of breakpoints that satisfy the constraint imposed by minvert (i.e., that breakpoints need to be at least minvert vertebrae apart) and fits the segmented regression models implied by each combination. These are multivariate regression models with the PCO scores specified by scores as the outcomes. When cont = TRUE, these regression models are continuous; i.e., the regression lines for each region connect at the breakpoints. Otherwise, the models are discontinuous so that each region has its own intercept and slope. The models are fit using .lm.fit(), which efficiently implements ordinary least squares regression.

When exhaus = FALSE, heuristics are used to reduce the number of models to fit, which can be useful for keeping the size of the resulting object down by avoiding fitting models corresponding to breakpoint combinations that yield a poor fit to the data. Only breakpoint combinations that correspond to the breakpoints of the best fitting model with a smaller number of regions +/- 3 vertebrae are used, and only models that have an RSS smaller than half a standard deviation more the smallest RSS are kept.

addregions() should be used on an existing regions_results object to add models with more regions. Internally, it works just the same as calcregions().

ncomobs() computes an upper bound on the number of possible breakpoint combinations. When exhaus = FALSE or includebp is specified, the actual number of combinations will be smaller than that produced by ncombos().

When the number of possible combinations of breakpoints for a given number of regions (as computed by ncombos()) is larger than ncombos_file_trigger, the problem will be split into smaller problems, with the results of each stored in temporary files that are deleted when the function completes. These temporary files will be stored in the directory supplied to temp_file_dir. By default, this is the temporary directory produced by tempdir(). However, this directory can be deleted by R at any time without warning, which will cause the function to crash, so it is a good idea to supply your own directory that will be preserved. You can use ncombos() to check to see if the number of breakpoint combinations exceeds ncombos_file_trigger.

Value

A `regions_results` object with the following components:

- `results` - the results of the fitting process for each combination of breakpoints
- `stats` - statistics summarizing the fitting process. Use `summary()` to view this information in a clean format.

`ncombos()` returns a numeric vector with the number of breakpoint combinations for each number of regions (which are stored as the names).

See Also

`calcmodel()` to fit a segmented regression model for a single set of breakpoints; `modelselect()` to select the best model for each number of regions based on RSS; `modelsupport()` to compute statistics the describe the support of the best models; `calcBPvar()` to compute the variability in the optimal breakpoints.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)

# Fit segmented regression models for 1 to 5 regions
# using PCOs 1 to 4 and a continuous model with a
# non-exhaustive search
regionresults <- calcregions(alligator_PCO,
                             scores = 1:4,
                             noregions = 5,
                             minvert = 3,
                             cont = TRUE,
                             exhaus = FALSE,
                             verbose = FALSE)

regionresults

# View model fitting summary
summary(regionresults)

# Add additional regions to existing results,
# exhaustive search this time
regionresults <- addregions(regionresults,
                            noregions = 6:7,
                            exhaus = TRUE,
                            verbose = FALSE)

regionresults
```

```
summary(regionresults)

# Fit segmented regression models for 1 to 5 regions
# using PCOs 1 to 4 and a discontinuous model with a
# exhaustive search, excluding breakpoints at vertebrae
# 10 and 15
regionresults <- calcregions(alligator_PCO,
                             scores = 1:4,
                             noregions = 5,
                             minvert = 3,
                             cont = FALSE,
                             omitbp = c(10, 15),
                             verbose = FALSE)

regionresults

summary(regionresults)

# Compute the number of breakpoint combinations for given
# specification using `ncombos()`; if any number exceeds
# the value supplied to `ncombos_file_trigger`, results
# will temporary be stored in files before being read in and
# deleted.
ncombos(alligator_PCO,
        noregions = 1:8,
        minvert = 3)
```

`dolphin` *Measurements from the vertebral column of a dolphin*

Description

Linear and angular measurements from Platanista gangetica SMNS 45653

Usage

```
dolphin
```

Format

A matrix with 40 vertebrae and 16 measurements. Column 1, vertebra, is the positional information.

modelperf *Assess model performance*

Description

`modelperf()` computes model performance statistics in the form of R^2 measures for a given combination of breakpoints.

Usage

```
modelperf(x, ...)

## S3 method for class 'regions_pco'
modelperf(
  x,
  scores,
  modelsupport = NULL,
  criterion = "aic",
  model = 1,
  bps = NULL,
  cont = TRUE,
  ...
)

## S3 method for class 'regions_sim'
modelperf(
  x,
  scores = NULL,
  modelsupport = NULL,
  criterion = "aic",
  model = 1,
  bps = NULL,
  cont = TRUE,
  ...
)

## S3 method for class 'regions_results_single'
modelperf(x, scores = NULL, ...)
```

Arguments

`x`	a `regions_pco` object, the output of a call to `svdPCO()`, or a `regions_results_single` object, the output of a call to `calcmodel()`.
`...`	ignored.
`scores`	`numeric`; the indices of the PCO scores for which fit statistics are to be computed.

modelsupport	a regions_modelsupport object, the output of a call to modelsupport(). When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.
criterion	string; the criterion to use to select the best model for which breakpoints are to be displayed when modelsupport is specified. Ignored otherwise. Allowable options include "aic" to use the AICc and "bic" to use the BIC. Abbreviations allowed. Default is "aic". When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.
model	numeric; for which model among the best as determined by criterion should fit statistics be computed. 1 is the best model, 2 the second best, etc. Default is 1. When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.
bps	numeric; a vector of breakpoints for which model fit should be computed. When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.
cont	logical; whether to fit a model that is continuous (TRUE) or discontinuous (FALSE) at the breakpoints supplied to bps. Default is TRUE. When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.

Details

modelperf() operates on a single model identified by breakpoints and whether the model is continuous or not. When x is a regions_pco object, the model is selected either as the best model in the supplied modelsupport object (where "best" is determined by the arguments to criterion and model) or as specified by the user using the arguments to bps and cont. When x is a regions_results_single object, the breakpoints and model form are determined based on the supplied object.

Value

A regions_perf object containing the breakpoints of the specified model, the univariate R^2 and adjusted R^2 statistics for each PCO score, and the multivariate R^2 and adjusted R^2 statistics.

See Also

modelsupport() for assessing model support using information criteria; calcmodel() for fitting a single segmented regression model; plotsegreg() for plotting the results of a single segmented regression model.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)
```

```
# Evaluate model performance (R2) given supplied
# breakpoints for a continuous model
modelperf(alligator_PCO, scores = 1:3,
          bps = c(7, 15, 20), cont = TRUE)

plotsegreg(alligator_PCO, scores = 1:3,
           bps = c(7, 15, 20), cont = TRUE)
## See also `?calcmodel` for use with a single model

# Fit segmented regression models for 1 to 7 regions
# using PCOs 1 to 4 and a continuous model with a
# non-exhaustive search
regionresults <- calcregions(alligator_PCO,
                             scores = 1:4,
                             noregions = 7,
                             minvert = 3,
                             cont = TRUE,
                             exhaus = FALSE,
                             verbose = FALSE)

regionresults

# For each number of regions, identify best
# model based on minimizing RSS
bestresults <- modelselect(regionresults)

# Evaluate support for each model and rank
supp <- modelsupport(bestresults)

# Evaluate model performance (R2) for best model
# as chosen by BIC
modelperf(alligator_PCO, scores = 1:4,
          modelsupport = supp,
          criterion = "bic", model = 1)

# Plot that model for the first PCO score
plotsegreg(alligator_PCO, scores = 1:4,
           modelsupport = supp,
           criterion = "bic", model = 1)

## See `?simregions` for use with simulated data
```

`modelselect` *Select the best models*

Description

`modelselect()` narrows down the search for the best model by identifying the best model for each number of regions as determined by its residual sums of squares (RSS).

Usage

```
modelselect(results, scores = NULL)
```

Arguments

results	a regions_results object; the output of a call to calcregions() or addregions().
scores	numeric; a vector corresponding to the indices of the PCOs the R^2 of which will be used to determine the best model for each number of regions. If NULL, the default, all PCOs used included in the fitting will be used.

Value

A regions_modelselect object, which contains information about the best models for each number of regions extracted from results.

See Also

modelsupport() for computing statistics that describe the support of each model using information criteria; modelperf() for computing fit statistics for selected models.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)

# Fit segmented regression models for 1 to 7 regions
# using PCOs 1 to 4 and a continuous model with a
# non-exhaustive search
regionresults <- calcregions(alligator_PCO,
                             scores = 1:4,
                             noregions = 7,
                             minvert = 3,
                             cont = TRUE,
                             exhaus = FALSE,
                             verbose = FALSE)

regionresults

# For each number of regions, identify best
# model based on minimizing RSS
bestresults <- modelselect(regionresults)
bestresults

# Evaluate support for each model and rank models
supp <- modelsupport(bestresults)
supp
```

```
# 5 regions best based on AICc; 6 regions based on BIC
```

`modelsupport` *Evaluate model support*

Description

`modelsupport()` computes measures of the relative support of each of the best models identified by `modelselect()` to facilitate selecting the optimal number and position of regions. These measures are in the form of information criteria (AICc and BIC).

Usage

```
modelsupport(models)
```

Arguments

`models`	a `regions_modelselect` object; the output of a call to `modelselect()`.

Value

A `regions_modelsupport` object, which contains the best model for each number of regions as determined by the AICc and BIC. The computed statistics are `AICc`/`BIC`–the value of the information criterion (IC) for each model, `deltaAIC`/`deltaBIC`–the difference between the IC for the corresponding model and that of the model with the lowest IC value, `model_lik`–the likelihood ratio of the model against the model with the lowest IC value, and `Ak_weight`/`BIC_weight`–the Akaike weights for each model used to compute the region score. The region score is a weighted average of the numbers of regions, weighted by the Akaike weights to represent the variability around the optimal number of regions.

See Also

`modelselect()`, `calcregions()`, `calcBPvar()`, `modelperf()`, `plotsegreg()`

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)

# Fit segmented regression models for 1 to 7 regions
# using PCOs 1 to 4 and a continuous model with a
# non-exhaustive search
regionresults <- calcregions(alligator_PCO,
```

```
                              scores = 1:4,
                              noregions = 7,
                              minvert = 3,
                              cont = TRUE,
                              exhaus = FALSE,
                              verbose = FALSE)

regionresults

# For each number of regions, identify best
# model based on minimizing RSS
bestresults <- modelselect(regionresults)
bestresults

# Evaluate support for each model and rank models
supp <- modelsupport(bestresults)
supp

# 5 regions best based on AICc; 6 regions based on BIC
```

`musm` *Measurements from the vertebral column of a mouse*

Description

Linear and angular measurements from Mus musculus MCZ 59560

Usage

```
musm
```

Format

A matrix with 23 vertebrae and 19 measurements. Column 1, vertebra, is the positional information.

`PCOload` *Calculate PCO loadings*

Description

`PCOload()` computes the loadings for each principal coordinates (PCOs) analysis score, which are the correlations between the features used to compute the PCOs and the PCOs.

Usage

```
PCOload(x, scores)

## S3 method for class 'regions_pco_load'
plot(x, ...)
```

Arguments

x	for PCOload(), a regions_pco object; the output of a call to svdPCO(). For plot(), a regions_pco_load object.
scores	a numeric vector containing the indices of the desired scores.
...	ignored.

Details

the loadings for a constructed variable, vert.size, are also computed and displayed. This is computed as the mean of the features for each vertebra.

Value

PCOload() returns a regions_pco_load object, which is a matrix with a column for each PCO score requested and a row for each variable in the original dataset; values indicate the correlation between each variable and each PCO score. plot() returns a ggplot object, which can be manipulated using *ggplot2* syntax, that displays the loadings visually.

See Also

svdPCO() for computing the PCOs; plot.regions_pco() for visualizing the correlations between PCO scores.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)

# Compute PCO loadings
loadings <- PCOload(alligator_PCO, scores = 1:4)
loadings

# Plot loadings
plot(loadings)
```

PCOselect *Select PCO scores*

Description

PCOselect() provides several methods to select the number of principal coordinates (PCOs) analysis scores to be used in subsequent analyses.

Usage

```
PCOselect(
  pco,
  method = "manual",
  scores = NULL,
  cutoff = 0.05,
  nreps = 500,
  results = NULL,
  criterion = "aic",
  verbose = TRUE
)

## S3 method for class 'regions_pco_select'
plot(x, ...)

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

Arguments

pco	a regions_pco object; the output of a call to svdPCO().
method	string; the method used to select the number of PCOs. Allowable options include "manual", "boot", "variance", and "max". Default is "manual". Abbreviations allowed. See Details.
scores	when method = "manual", the number of PCO scores to use.
cutoff	when method = "variance", the cutoff for the variance explained by each PCO score.
nreps	when method = "boot", the number of bootstrap replications to use.
results	when method = "max", a regions_results object, the output of a call to calcregions().
criterion	when method = "max", which criterion should be used to select the number of scores. Allowable options include "aic" and "bic". Abbreviations allowed.
verbose	when method = "boot", whether to display a progress bar. Default is TRUE.
x	for plot.regions_pco_select(), a regions_pco_select object, the output of a call to PCOselect() with method = "boot" or "max".
...	ignored.
object	a regions_pco_select object, the output of a call to PCOselect() with method = "max".

Details

Each method provides an alternate way to select the number of scores. These are described below.

method = "manual"**::**

This simply returns the number supplied to scores after running some checks to ensure it is valid.

`method = "boot"`:

Bootstrapping works by comparing the eigenvalue distributions of PCOs to those with randomized data in order to extract PCO axes with significant signal, which are defined as those with eigenvalues greater than those from randomized data. The returned PCO cutoff is the largest PCO axis whose eigenvalues fall below the mean eigenvalue for that axis from the randomized data. Data are randomly sampled by row. Bootstrapping is sensitive to unequal variances of columns, so `scale = TRUE` should be set in the call to `svdPCO()`, which is the default; the data are scaled in the same way prior to bootstrapping. The `plot()` method displays the eigenvalues of the true PCOs and boxplots summarizing the distribution of the bootstrapped eigenvalues for each PCO.

`method = "variance"`:

This method works by computing the ratio of each eigenvalue to the sum of the eigenvalues (i.e., to compute the proportion of variance explained by each PCO score) and select the number of scores with ratios greater than the cutoff value supplied to `cutoff`.

`method = "max"`:

This method works by selecting the smallest number of PCOs that gives a region score within .001 of the maximum possible region score for the segmented models fit in the object supplied to `results`. Which criterion is maximized (AIC or BIC) is determined by the value supplied to `criterion`. The `summary()` method displays the region score (estimated number of regions) for each PCO (`RSind`) and for PCOs cumulatively (`RScum`) selected using the AICc or BIC as well as the cumulative proportion of variance explained by the PCOs. The `plot()` method displays this information graphically, with the left y-axis displaying the region score for the PCOs individually (pale blue triangles) and cumulatively (orange circles) using each of the two criteria, and the right y-axis displaying the cumulative percentage of variance explained by the PCOs.

Value

For `PCOselect()`, a `regions_pco_select` object, which is a numeric vector containing the indices of the chosen PCOs, with attributes containing information about the PCO scores chosen by the specified method. When `method = "boot"`, the bootstrap results are stored in the `"boot"` attribute. When `method = "max"`, the `regions_results` object passed to `regions` and other information about the quality of fit for each number of PCOs are stored in the `"pcomax"` attribute.

The `plot()` methods each return a `ggplot` object that can manipulated using **ggplot2** syntax. The `summary()` method returns a data.frame of results.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)

# Select which PCOs to use
## Manually (first 4 PCOs)
(PCOs <- PCOselect(alligator_PCO, "manual", scores = 4))
```

```
## Using variance cutoff: PCOs that explain 5% or more
## of total PCO variance
(PCOs <- PCOselect(alligator_PCO, "variance", cutoff = .05))

## Using bootstrapping with 50 reps (more reps should
## be used in practice; default is fine)
(PCOs <- PCOselect(alligator_PCO, "boot", nreps = 50))

plot(PCOs) #plot true eigenvalues against bootstrapped

## Using PCOs that optimize region score:
regionresults <- calcregions(alligator_PCO, scores = 1:21, noregions = 7,
                             minvert = 3, cont = TRUE, exhaus = TRUE,
                             verbose = FALSE)

(PCOs <- PCOselect(alligator_PCO, "max",
                   results = regionresults,
                   criterion = "bic"))

plot(PCOs)

summary(PCOs)
```

`plot.regions_pco` *Plot PCO axes*

Description

`plot()` visualizes the relationship between a PCO axis and the vertebra or between pairs of PCO axes.

Usage

```
## S3 method for class 'regions_pco'
plot(x, pco_y = 1, pco_x = NULL, ...)
```

Arguments

`x`	a `regions_pco` object; the output of a call to `svdPCO()`.
`pco_y, pco_x`	number; PCO score indices for the y- and x-axes, respectively. `pco_x` can be `NULL`.
`...`	arguments passed to `ggplot2::geom_text()` when `pco_x` is not `NULL`. If `scores` is supplied as an argument, it will replace `pco_y` if unspecified.

Details

When `pco_x` is `NULL` (the default), `plot()` will display a scatterplot of the PCO axis identified by `pco_y` and vertebra position using `ggplot2::geom_point()`. This plot is similar to that generated by `plotsegreg()`. Otherwise, `plot()` uses `ggplot2::geom_text()` to identify vertebrae positions in the space corresponding to the requested PCOs.

Value

A ggplot object.

See Also

svdPCO() for generating the PCO scores. plot.regions_sim() for plotting PCO scores against vertebra position for simulated PCOs. plotsegreg() for plotting PCO scores against vertebra position after selecting breakpoints for a segmented regression.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data,
                        metric = "gower")

alligator_PCO

# Plot PCOs against vertebra index
plot(alligator_PCO, pco_y = 1:2)

# Plot PCOs against each other
plot(alligator_PCO, pco_y = 1, pco_x = 2)
```

---

plotsegreg *Plot a segmented regression model*

Description

plotsegreg() plots the fitted lines resulting from a segmented regression model.

Usage

```
plotsegreg(x, scores, ...)

## S3 method for class 'regions_pco'
plotsegreg(
  x,
  scores,
  modelsupport = NULL,
  criterion = "aic",
  model = 1,
  bps = NULL,
  cont = TRUE,
```

```
  ...
)

## S3 method for class 'regions_sim'
plotsegreg(
  x,
  scores,
  modelsupport = NULL,
  criterion = "aic",
  model = 1,
  bps = NULL,
  cont = TRUE,
  ...
)

## S3 method for class 'regions_results_single'
plotsegreg(x, scores, ...)
```

Arguments

x	a regions_pco object, the output of a call to svdPCO(), or a regions_results_single object, the output of a call to calcmodel().
scores	numeric; the indices of the PCO scores for which the fitted lines should be plotted.
...	ignored.
modelsupport	a regions_modelsupport object, the output of a call to modelsupport(). When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.
criterion	string; the criterion to use to select the best model for which breakpoints are to be displayed when modelsupport is specified. Ignored otherwise. Allowable options include "aic" to use the AICc and "bic" to use the BIC. Abbreviations allowed. Default is "aic". When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.
model	numeric; for which model among the best as determined by criterion should fitted lines be plotted. 1 is the best model, 2 the second best, etc. Default is 1. When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.
bps	numeric; a vector of breakpoints for which model fitted lines should be plotted. When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.
cont	logical; whether to fit a model that is continuous (TRUE) or discontinuous (FALSE) at the breakpoints supplied to bps. Default is TRUE. When x is a regions_pco object, either modelsupport, criterion, and model must be supplied or bps and cont must be supplied. See Details.

Details

`plotsegreg()` operates on a single model identified by breakpoints and whether the model is continuous or not. When `x` is a `regions_pco` object, the model is selected either as the best model in the supplied `modelsupport` object (where "best" is determined by the arguments to `criterion` and `model`) or as specified by the user using the arguments to `bps` and `cont`. When `x` is a `regions_results_single` object, the breakpoints and model form are determined based on the supplied object.

`plot()` is an alias for `plotsegreg()` for `regions_results_single` objects.

Value

A `ggplot` object that can be manipulated using *ggplot2* syntax.

See Also

`modelsupport()` for assessing model support using information criteria; `calcmodel()` for fitting a single segmented regression model; `modelperf()` for computing fit statistics for a single segmented regression model.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)

# Evaluate model performance (R2) given supplied
# breakpoints for a continuous model
modelperf(alligator_PCO, scores = 1:3,
          bps = c(7, 15, 20), cont = TRUE)

plotsegreg(alligator_PCO, scores = 1:3,
           bps = c(7, 15, 20), cont = TRUE)
## See also `?calcmodel` for use with a single model

# Fit segmented regression models for 1 to 7 regions
# using PCOs 1 to 4 and a continuous model with a
# non-exhaustive search
regionresults <- calcregions(alligator_PCO,
                             scores = 1:4,
                             noregions = 7,
                             minvert = 3,
                             cont = TRUE,
                             exhaus = FALSE,
                             verbose = FALSE)

regionresults
```

```
# For each number of regions, identify best
# model based on minimizing RSS
bestresults <- modelselect(regionresults)

# Evaluate support for each model and rank
supp <- modelsupport(bestresults)

# Evaluate model performance (R2) for best model
# as chosen by BIC
modelperf(alligator_PCO, scores = 1:4,
          modelsupport = supp,
          criterion = "bic", model = 1)

# Plot that model for the first PCO score
plotsegreg(alligator_PCO, scores = 1:4,
           modelsupport = supp,
           criterion = "bic", model = 1)

## See `?simregions` for use with simulated data
```

`plotvertmap` *Plot a vertebra map*

Description

`plotvertmap()` plots a map of the supplied vertebrae, optionally adding colors, marks, and text to identify existing and estimated features of the vertebrae.

Usage

```
plotvertmap(
  x,
  type = "count",
  bps = NULL,
  modelsupport = NULL,
  criterion = "aic",
  model = 1,
  bpvar = NULL,
  bp.sd = NULL,
  sd.col = "black",
  dropNA = FALSE,
  text = FALSE,
  name = NULL,
  centraL = NULL,
  reg.lim = NULL,
  lim.col = "black",
  block.cols = NULL,
  block.lim = NULL
)
```

Arguments

x	a regions_data, regions_pco, or regions_sim object; the output of a call to process_measurements(), svdPCO(), or simregions(), respectively.
type	string; the labeling of the x-axis of the plot. Either "count" to identify the vertebra index (or absolute position when centraL is supplied) or "percent" to identify the percent vertebra count (or percent total length when centraL is supplied). Abbreviations allowed. Default is "count".
bps	an optional vector containing the region breakpoints. One of bps, modelsupport, or bpvar should be specified to display regions breakpoints. See Details.
modelsupport	an optional regions_modelsupport object; the output of a call to modelsupport(). One of bps, modelsupport, or bpvar should be specified to display regions breakpoints. See Details.
criterion	string; the criterion to use to select the best model for which breakpoints are to be displayed when modelsupport is specified. Ignored otherwise. Allowable options include "aic" to use the AICc and "bic" to use the BIC. Abbreviations allowed. Default is "aic".
model	numeric; from which model among the best as determined by criterion should breakpoints be selected when modelsupport is supplied. Ignored otherwise. 1 is the best model, 2 the second best, etc. Default is 1.
bpvar	an optional regions_BPvar object; the output of a call to calcBPvar(). One of bps, modelsupport, or bpvar should be specified to display regions breakpoints. See Details.
bp.sd	an optional vector of the standard deviations of the breakpoints (e.g., as calculated by calcBPvar()). When bpvar is supplied, the weighted standard deviations are used.
sd.col	when bp.sd is specified, the color of the mark on plot indicating the standard deviations. Default is black.
dropNA	logical; when some vertebrae are missing, e.g., due to subsampling or starting the analysis at a vertebra beyond the first, whether to remove the missing vertebrae from the plot (TRUE) or retain them and label them as missing (i.e., lacking a region) (FALSE). Default is FALSE to retain them.
text	logical; whether to print the vertebra index on each vertebra. Default is FALSE.
name	an optional string containing a label used on the left side of the plot.
centraL	an optional numeric vector containing centrum length for each vertebra, which is used to change the size of the plotted vertebrae, or a string containing the name of the variable in the original dataset containing centrum length. Should be of length equal to the number of included vertebrae (i.e., the length of the original dataset). Any vertebrae with centrum length of 0 will be omitted.
reg.lim	a vector of breakpoints indicating other region limits, e.g., anatomic regions.
lim.col	when reg.lim is specified, the color of the lines separating the regions. Default is black.
block.cols	when breakpoints are specified (i.e., using bps, modelsupport, or bpvar) and block.lim is not specified, a vector of color names or hex codes, one for each

region. If not specified, RColorBrewer::brewer.pal() with name = "paired" will be used to generate colors. When block.lim is specified, a named list of vectors of color names or hex codes. See Details.

block.lim — a vector of breakpoints indicating the limits of traditional regions, which will be colored using block.cols. See Details.

Details

plotvertmap() uses ggplot2::geom_rect() to create the plot. The plots are best viewed with a short height and a long width.

Specifying breakpoints::

There are three ways to specify regions in plotvertmap(). First is to supply the vector of breakpoints directly to bps. Second is to supply a regions_modelsupport object to modelsupport. When supplied, the criterion and model arguments can be used to select which of the sets of breakpoints in the object is to be used. model selects which breakpoint model is to be used (1 for first best, 2 for second best, etc.), and criterion determines which criterion (AICc or BIC) is used to rank the models. Third is to supply regions_BPvar object to bpvar. The weighted average breakpoints will be used after rounding (e.g., a weighted average breakpoint of 3.3 will place vertebrae 1, 2, and 3 in a region, and a weighted average breakpoint of 3.9 will place vertebrae 1, 2, 3, and 4 in a region).

Using block.cols::

When block.lim is specified, block.cols must be specified as a list of vectors of colors, with an entry for each "block". Blocks are predefined regions separate from those specified using the above arguments, e.g., traditional regions. For each region, the most common block is found and assigned to that region. A color of that block as supplied in block.cols is used to color that region. So, each block needs as many colors as there are regions assigned to it. For example, if regions 1 and 2 are both assigned to block 1 (i.e., because block 1 is the most common block in those regions), the entry in block.cols for that block must have (at least) 2 colors. If an incorrect number of colors per block is supplied, an error will be thrown identifying which blocks are lacking colors. See Examples.

Value

A ggplot object that can be manipulated using ggplot2 syntax.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)

# Plot vertebral map with specified breakpoints
plotvertmap(alligator_PCO,
            type = "percent",
```

```
            name = "Alligator",
            bps = c(8, 15, 19),
            text = TRUE)

# Fit segmented regression models for 1 to 7 regions
# using PCOs 1 to 4 and a continuous model with a
# exhaustive search
regionresults <- calcregions(alligator_PCO,
                             scores = 1:4,
                             noregions = 7,
                             minvert = 3,
                             cont = TRUE,
                             exhaus = TRUE,
                             verbose = FALSE)

# For each number of regions, identify best
# model based on minimizing RSS
bestresults <- modelselect(regionresults)

# Evaluate support for each model and rank models
supp <- modelsupport(bestresults)

# Plot vertebral map with breakpoints corresponding to
# best segmented regression model as determined by
# AICc
plotvertmap(alligator_PCO,
            type = "percent",
            name = "Alligator",
            modelsupport = supp,
            model = 1,
            criterion = "aic",
            text = TRUE)

# Plot vertebral map with breakpoints corresponding to
# best segmented regression model as determined by
# AICc, using centrum length to size vertebrae
plotvertmap(alligator_PCO,
            name = "Alligator",
            modelsupport = supp,
            model = 1,
            criterion = "aic",
            centraL = "CL",
            text = TRUE)

# Compute Akaike-weighted location and SD of optimal
# breakpoints using top 10% of models with 4 regions
bpvar <- calcBPvar(regionresults, noregions = 5,
                   pct = .1, criterion = "aic")

#Using weighted BPs and SDs from calcBPvar()
plotvertmap(alligator_PCO, name = "Dolphin",
            bpvar = bpvar,
            text = TRUE)
```

porpoise *Measurements from the vertebral column of three porpoises*

Description

Linear and angular measurements from Phocoena phocoena NRM 815072 (porpoise1), NRM 835011 (porpoise2), and NRM 855083 (porpoise3).

Usage

```
data("porpoise")
```

Format

Each is a data frame with 58, 56, and 59 vertebrae (respectively) and 16 measurements. Column 1, Vertebra, contains the positional information.

References

Gillet, A., Frederich, B., Pierce, S. E., & Parmentier, E. (2022). Iterative habitat transitions are associated with morphological convergence of the backbone in delphinoids. Journal of Mammalian Evolution, 29(4), 931-946.

process_measurements *Process vertebra measurements*

Description

process_measurements() initializes the analysis workflow by processing a dataset of vertebra measurements into an object usable by **MorphoRegions**. Such processing includes identifying the vertebra indices and the measurements and filling in missing values.

Usage

```
process_measurements(data, pos = 1L, measurements, fillNA = TRUE)
```

Arguments

data	a data.frame containing a column of vertebra indices and measurements for each vertebra, or a list thereof for multiple specimens.
pos	the name or index of the variable in data containing the vertebra indices. Default is to use the first column.
measurements	the names or indices of the variables in data containing the relevant vertebra measurements. If unspecified, will use all variables other than that specified in pos.
fillNA	logical; whether to fill in missing values using a simple linear imputation. Default is TRUE. See Details.

Details

Any rows with missing values for all measurements will be removed. When missing values in non-removed rows are present and `fillNA` is set to `TRUE`, `process_measurements()` fills them in if the sequence of missing values is no greater than 2 in length. For numeric variables, it uses a linear interpolation, and for categorical variables, it fills in the missing values with the surrounding non-missing values if they are identical and leaves them missing otherwise. Otherwise, missing values are left as they are.

When a list of data frames is supplied to `data`, only the variables named in `measurements` that are common across datasets will be stored as measurement variables.

Value

A `regions_data` object, which is a list of data.frames (one for each specimen) with attributes containing metadata.

See Also

`svdPCO()` for computing principal coordinate axes from processed vertebra data.

Examples

```
# Process dataset; vertebra index in "Vertebra" column
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Process multiple datasets; vertebra index in first column
data("porpoise")

porpoise_data <- process_measurements(list(porpoise1,
                                           porpoise2,
                                           porpoise3),
                                      pos = 1)
```

---

simregions *Simulate regions data*

Description

`simregions()` simulates vertebrae and PCOs that satisfy certain constraints.

Usage

```
simregions(
  nvert,
  nregions,
  nvar = 1,
```

```
  r2 = 0.95,
  minvert = 3,
  cont = TRUE,
  sl.dif = 0
)

## S3 method for class 'regions_sim'
plot(x, scores = 1, lines = TRUE, ...)
```

Arguments

nvert	numeric; the number of vertebrae for which to simulate data.
nregions	numeric; the desired number of regions in the simulated data.
nvar	numeric; the number of PCO axes to simulate. Default is 1.
r2	numeric; a vector containing the R^2 of the true segmented regression model for each simulated PCO. If a single value is supplied, all PCOs will receive that value. Otherwise, one value should be supplied for each simulated PCO.
minvert	numeric; the minimum number of vertebrae allowed in each region. Default is 3.
cont	logical; whether to use models that are continuous (TRUE) or discontinuous (FALSE) at the breakpoints to generate the data. Default is TRUE.
sl.dif	numeric; the minimum required difference in slopes between adjacent regions, expressed as a proportion of the maximal difference between allowable slopes. Must be between 0 and 1. See Details.
x	a regions_sim object.
scores	numeric; for which simulated PCO scores the simulated values should be plotted.
lines	logical; whether to display the simulated regression lines on the plot. Default is TRUE.
...	ignored.

Details

simregions() generates PCO scores for each requested vertebra such that certain conditions are met. The slopes for each region are drawn from a uniform distribution with limits of -.5 and .5. If a set of slopes contains two adjacent slopes that have a difference less than sl.dif, it is rejected and a new one is drawn. The scaling of the PCOs is determined by the slopes and the requested R^2. The PCOs will not necessarily be in order from most variable to least variable as they are in a traditional PCO analysis.

Intercepts (the intercept of the first region when cont = TRUE and the intercept of all regions when cont = FALSE) are drawn from a uniform distribution with limits of $-n/4$ and $n/4$, where n is the number of breakpoints, one less than nregions. Intercepts other than the first when cont = TRUE are determined by the slopes.

The cont, r2, and sl.dif arguments control how easy it is for fitted segmented regression models to capture the true structure. When cont = TRUE, it can be harder to determine exactly where regions

begin and end, especially if `sl.dif` is 0. When `r2` is high, there is little variation around the true line, so the fitted lines will be more precise and region boundaries clearer. When `sl.dif` is high, slopes of adjacent regions are different from each other, so it is easier to detect region boundaries. Setting `sl.dif` to between .5 and 1 ensures that the slopes in adjacent regions have different signs.

Value

`simregions()` returns a `regions_sim` object, which contains the vertebra indices in the `Xvar` entry, the PCO scores in the `Yvar` entry, the simulated breakpoints in the `BPs` entry, the simulated model coefficients in the `coefs` entry, and the simulated error standard deviation in the `ersd` entry. The attribute `"design"` contains the design matrix, which when multiplied by the coefficients and added to a random normal variate with standard deviation equal to the error standard deviation yields the observed PCO scores.

`plot()` returns a `ggplot` object that can be manipulated using `ggplot2` syntax. The plot is similar to that produced by `plot.regions_pco()` and to that produced by `plotsegreg()` except that the displayed lines (if requested) are the true rather than fitted regression lines.

See Also

`calcregions()` for fitting segmented regression models to the simulated data; `calcmodel()` for fitting a single segmented regression model to the simulated data; `plotsegreg()` for plotting estimated regression lines.

Examples

```
# Simulate 40 vertebra, 4 regions (3 breakpoints), 3 PCOs,
# true model R2 of .9, continuous
set.seed(11)
sim <- simregions(nvert = 30, nregions = 4, nvar = 3, r2 = .95,
                  minvert = 3, cont = TRUE)

sim

# Plot the true data-generating lines and breakpoints
plot(sim, scores = 1:3)

# Run segmented regression models on simulated data,
# up to 6 regions
simresults <- calcregions(sim, scores = 1:3, noregions = 6,
                          minvert = 3, cont = TRUE,
                          verbose = FALSE)

summary(simresults)

# Select best model for each number of regions
(simmodels <- modelselect(simresults))

# Evaluate support for each model and rank models
(simsupp <- modelsupport(simmodels))
# AICc supports 3-4 regions
```

```
# Evaluate model performance of best model
modelperf(sim, modelsupport = simsupp,
          criterion = "aic", model = 1)
# Second best model (3 regions) does quite well, too
modelperf(sim, modelsupport = simsupp,
          criterion = "aic", model = 2)

#Plot best model fit
plotsegreg(sim, scores = 1:3,
           modelsupport = simsupp,
           criterion = "aic", model = 1)

# Calculate variability of estimate breakpoints for
# 3-region model; high uncertainty for breakpoints
# 1 and 2. Note weighted value for breakpoint 2
# differs from that of best model
bpvar <- calcBPvar(simresults, noregions = 4,
                   pct = .05, criterion = "aic")
bpvar

# Plot estimated vertebral map with variability
plotvertmap(sim, modelsupport = simsupp, model = 1,
            criterion = "aic", text = TRUE)

# True map; pretty close
plotvertmap(sim, bps = c(3, 7, 24),
            text = TRUE)
```

`subsample` *Subsample a dataset*

Description

`subsample()` creates a smaller version of the original dataset by sampling its rows. Because PCOs should be computed on the full dataset and most other functions take in `regions_pco` objects, `subsample()` requires a `regions_pco` object as its input.

Usage

```
subsample(pco, sample = NULL, type = "seq")
```

Arguments

`pco`	a `regions_pco` object; the output of a call to `svdPCO()`.
`sample`	`numeric`; either the number or proportion of vertebrae to remain the sampled dataset. If `NULL`, the original dataset is returned.
`type`	string; the type of subsampling to do, either `"seq"` for sampling in sequence or `"random"` for random sampling. Default is `"seq"`. Abbreviations allowed.

Value

A regions_pco object, a subset of the original supplied to pco. The original dataset is stored as an attribute, which itself contains the subsampling indices.

See Also

svdPCO(), process_measurements(), plotvertmap() to visualize the vertebral map after subsampling.

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data)

# Plot vertebrae before subsampling
plotvertmap(alligator_PCO, dropNA = FALSE,
            text = TRUE)

# Subsample data after estimating PCOs; subsample down
# to 15 vertebrae
alligator_PCO_sample <- subsample(alligator_PCO,
                                  sample = 15)

# Plot vertebrae after subsampling; gray vertebrae
# have been dropped
plotvertmap(alligator_PCO_sample, dropNA = FALSE,
            text = TRUE)
```

---

svdPCO *Calculate PCO (principal co-ordinates analysis) based on SVD*

Description

Calculates distance matrix from raw data, then conducts a PCO ordination using a single value decomposition (SVD). This differs from other PCO functions which use stats::cmdscale() and rely on a spectral decomposition.

Usage

```
svdPCO(x, metric = "gower", scale = TRUE)
```

Arguments

x	a regions_data object; the output of a call to process_measurements().
metric	string; the distance matrix calculation metric. Allowable options include those support by cluster::daisy(), which are "euclidean", "manhattan", or "gower". Default is "gower". Abbreviations allowed.
scale	logical; whether to scale the variables prior to including them in the PCO estimation. Default is TRUE, which is especially advisable when using the bootstrap to select the number of PCOs to use in downstream analyses. Passed to the stand argument of cluster::daisy(). Ignored if metric = "gower".

Value

A regions_pco object, which contains eigenvectors in the scores component and eigenvalues in the eigen.val component. The original dataset is stored in the data attribute.

See Also

plot.regions_pco() for plotting PCO axes

cluster::daisy(), which is used to compute the distance matrix used in the calculation; stats::cmdscale() for a spectral decomposition-based implementation

Examples

```
data("alligator")

alligator_data <- process_measurements(alligator,
                                       pos = "Vertebra")

# Compute PCOs
alligator_PCO <- svdPCO(alligator_data,
                        metric = "gower")

alligator_PCO

# Plot PCOs against vertebra index
plot(alligator_PCO, pco_y = 1:2)

# Plot PCOs against each other
plot(alligator_PCO, pco_y = 1, pco_x = 2)
```

Index