Package 'SoundShape'

October 29, 2020

Title Sound Waves Onto Morphometric Data

Version 1.0

Date 2020-10-07

Description Implement a promising, and yet little explored protocol for bioacoustical analysis, the eigensound method by MacLeod, Krieger and Jones (2013) <doi:10.4404/hystrix-24.1-6299>. Eigensound is a multidisciplinary method focused on the direct comparison between stereotyped sounds from different species. 'SoundShape', in turn, provide the tools required for anyone to go from sound waves to Principal Components Analysis, using tools extracted from traditional bioacoustics (i.e. 'tuneR' and 'seewave' packages), geometric morphometrics (i.e. 'geomorph' package) and multivariate analysis (e.g. 'stats' package). For more information, please see Rocha and Romano (in prep) and check 'SoundShape' repository on GitHub for news and updates <https://github.com/p-rocha/SoundShape>.

Depends R (>= 3.3.1)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

URL https://github.com/p-rocha/SoundShape

BugReports https://github.com/p-rocha/SoundShape/issues

Imports abind, geomorph (>= 3.0.2), graphics, grDevices, plot3D, reshape2, seewave, stats, tuneR, utils

Suggests vegan, knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Pedro Rocha [aut, cre], Pedro Romano [ctb]

Maintainer Pedro Rocha <p.rocha1990@gmail.com>

Repository CRAN

Date/Publication 2020-10-29 10:20:11 UTC

R topics documented:

`align.wave` *Automatic placement of calls at beggining of sound window*

Description

Recreate each `".wav"` file on a given folder while placing calls at the beggining of sound window. New `".wav"` files will be stored on a new folder, which is automatically created.

Usage

```
align.wave(
  wav.at = NULL,
  wav.to = "Aligned",
  time.length = 1,
  time.perc = 0.005,
  dBlevel = 25,
  f = 44100,
  wl = 512,
  ovlp = 70
)
```

Arguments

`wav.at`	filepath to the folder where `".wav"` files are stored. Should be presented between quotation marks. By default: `wav.at = NULL` (i.e. user must specify the filepath to `".wav"` files)
`wav.to`	name of the folder where new `".wav"` files will be stored. Should be presented between quotation marks. By default: `wav.to = "Aligned"`
`time.length`	intended length for the time (X-axis) in seconds. Should be a value that encompasses all sounds in the study. By default: `time.length = 1`

`time.perc`	slight time gap (in percentage) relative to the intended length that encompass all sounds in the study (i.e. `time.length`). Intervals are added before and after the minimum and maximum time coordinates (X-values) from the selected curve of relative amplitude (`dBlevel`). By default: `time.perc = 0.005` (i.e. 0.5%)
`dBlevel`	absolute amplitude value to be used as relative amplitude contour, which will serve as reference for call placement. By default: `dBlevel = 25`
`f`	sampling frequency of `".wav"` files (in Hz). By default: `f = 44100`
`wl`	length of the window for the analysis. By default: `wl = 512`
`ovlp`	overlap between two successive windows (in %) for increased spectrogram resolution. By default: `ovlp = 70`

Author(s)

Pedro Rocha

References

MacLeod, N., Krieger, J. & Jones, K. E. (2013). Geometric morphometric approaches to acoustic signal analysis in mammalian biology. *Hystrix, the Italian Journal of Mammalogy, 24*(1), 110-125.

Rocha, P. & Romano, P. (*in prep*) The shape of sound: A new `R` package that crosses the bridge between Bioacoustics and Geometric Morphometrics. *Methods in Ecology and Evolution*

See Also

`eigensound`

Useful links:

- https://github.com/p-rocha/SoundShape
- Report bugs at https://github.com/p-rocha/SoundShape/issues

Examples

```
library(seewave)
library(tuneR)

# Create temporary folder to store ".wav" files
wav.at <- file.path(base::tempdir(), "align.wave")
if(!dir.exists(wav.at)) dir.create(wav.at)

# Create temporary folder to store results
store.at <- file.path(base::tempdir(), "align.wave-output")
if(!dir.exists(store.at)) dir.create(store.at)

# Select acoustic units to be analyzed
data(cuvieri)
spectro(cuvieri, flim = c(0,3)) # Visualize sound data that will be used

# Cut acoustic units from original Wave
```

```
cut.cuvieri1 <- cutw(cuvieri, f=44100, from=0, to=0.5, output = "Wave")
cut.cuvieri2 <- cutw(cuvieri, f=44100, from=0.7, to=1.2, output = "Wave")
cut.cuvieri3 <- cutw(cuvieri, f=44100, from=1.4, to=1.9, output = "Wave")

# Export ".wav" files containing selected acoustic units and store on previosly created folder
writeWave(cut.cuvieri1, filename = file.path(wav.at, "cut.cuvieri1.wav"), extensible = FALSE)
writeWave(cut.cuvieri2, filename = file.path(wav.at, "cut.cuvieri2.wav"), extensible = FALSE)
writeWave(cut.cuvieri3, filename = file.path(wav.at, "cut.cuvieri3.wav"), extensible = FALSE)

# Align acoustic units selected at 1% of time lenght
align.wave(wav.at = wav.at, wav.to = "Aligned",
           time.length = 0.5, time.perc = 0.01, dBlevel = 25)

# Verify alignment using eigensound function featuring analysis.type = "twoDshape"
eigensound(analysis.type = "twoDshape", wav.at = file.path(wav.at, "Aligned"), store.at = store.at,
           flim=c(0, 3), tlim=c(0,0.5), dBlevel = 25, plot.exp = TRUE, plot.as = "jpeg")
# To see jpeg files created, check folder specified by store.at
```

`centralis` *Vocalization of the frog* Physalaemus centralis

Description

Recording of a series of three stereotyped calls emitted by a male frog *Physalaemus centralis* (Amphibia, Anura, Leptodactylidae). Edited from original `".wav"` file for optimal sinal to noise ratio and reduced time duration.

Usage

```
data(centralis)
```

Format

An object of the class `"Wave"` (tuneR package).

Details

Duration = 2.89 s. Sampling Frequency = 44100 Hz.

Recorded at Formoso do Araguaia Municipality, Tocantins State, Brazil, on 9 December 1992. Air temperature 25ºC.

Source

Original recording housed at Fonoteca Neotropical Jacques Vielliard (FNJV-0031188). Recorded by Adão José Cardoso.

Examples

```
data(centralis)

seewave::oscillo(centralis)
seewave::spectro(centralis)
threeDspectro(centralis,tlim=c(0, 0.8), flim=c(0, 4), samp.grid=FALSE, dBlevel=25)
```

---

cuvieri *Vocalization of the frog* Physalaemus cuvieri

Description

Recording of a series of three stereotyped calls emitted by a male frog *Physalaemus cuvieri* (Amphibia, Anura, Leptodactylidae). Edited from original ".wav" file for optimal sinal to noise ratio and reduced time duration.

Usage

```
data(cuvieri)
```

Format

An object of class "Wave"; see (tuneR package).

Details

Duration = 1.96 s. Sampling Frequency = 44100 Hz.

Recorded at São José dos Campos Municipality, São Paulo State, Brazil, on 24 September 2013. Air temperature 22ºC.

Source

Original recording housed at Coleção Bioacústica da Universidade Federal de Minas Gerais (CBUFMG-00196). Recorded by Pedro Rocha.

Examples

```
data(cuvieri)

seewave::oscillo(cuvieri)
seewave::spectro(cuvieri)
threeDspectro(cuvieri, tlim=c(0, 0.5), flim=c(0, 4), samp.grid=FALSE)
```

`eig.sample` *Sample of 3D point coordinates (semilandmarks) acquired with* `eigensound` *function*

Description

This sample file was acquired using `eigensound(analysis.type = "threeDshape")` and features 3D point coordinates (i.e. semilandmarks) from the calls of three frog species: *Physalaemus centralis*, *P. cuvieri* and *P. kroyeri* (Amphibia, Anura, Leptodactylidae).

Usage

```
data(eig.sample)
```

Format

An object of the class `"array"` (`base` package).

`"array"` is a special type of `"list"` that can be thought as a filing cabinet, in which the `array` is the cabinet and each element is an arquive. This special `list` can be used in the subsequent steps of the eigensound protocol (MacLeod et al., 2013; Rocha & Romano *in prep*).

Details

Each species is represented by three stereotyped acoustic units (i.e. notes from their advertisement calls), which are available as sample data from `SoundShape` (`"Wave"` objects: `centralis`, `cuvieri` and `kroyeri`). See Rocha & Romano (*in prep*) for details.

Prior to `eigensound` analysis, each of the sample calls had the acoustic units selected, stored as separate `".wav"` files, and aligned at beggining of sound window using `align.wave` (see `Examples` section).

`eig.sample` is composed of 9 elements (i.e. three species, each represented by three acoustic units). Each element is a matrix with 3200 rows and 3 columns (i.e. X, Y and Z coordinates of 3200 semilandmarks). The number of semilandmarks acquired will depend on the number of cells per side on the sound window (i.e. `x.length` and `y.length` arguments from `eigensound` function).

The analysis itself (`eigensound` function) featured relative amplitude backgroud at 25 dB (`dBlevel = 25`), sampling grid of 70 cells on the time (X-axis) and 47 cells on the frequency (Y-axis) (`x.length = 70`, `y.length = 47`, respectively). Sound window ranged from 0 to 0.8 s (X-axis), and from 0 to 4 kHz (Y-axis) (`tlim = c(0,0.8)`, `flim = c(0,4)`, respectively).

Spectrogram parameters were the same as `eigensound` default: `f = 44100`, `wl = 512`, `ovlp = 70`.

Source

Sample data of `"Wave"` objects employed on `eigensound` analysis:

- `centralis`: Advertisement call of *Physalaemus centralis*; original recording housed at Fonoteca Neotropical Jacques Vielliard (FNJV-0031188). Recorded by Adão José Cardoso.

- cuvieri: Advertisement call of *Physalaemus cuvieri*; original recording housed at Coleção Bioacústica da Universidade Federal de Minas Gerais (CBUFMG-00196). Recorded by Pedro Rocha.
- kroyeri: Advertisement call of *Physalaemus kroyeri*; Original recording housed at Fonoteca Neotropical Jacques Vielliard (FNJV-0032047). Recorded by Werner Bokermann.

Examples

```
data(eig.sample)

# PCA using 3D semilandmark coordinates
pca.eig.sample <- stats::prcomp(geomorph::two.d.array(eig.sample))

# Verify names for each acoustic unit and the order in which they appear
dimnames(eig.sample)[[3]]

# Create factor to use as groups in subsequent ordination plot
sample.gr <- factor(c(rep("centralis", 3), rep("cuvieri", 3), rep("kroyeri", 3)))

# Plot result of Principal Components Analysis
pca.plot(PCA.out = pca.eig.sample, groups = sample.gr, conv.hulls = sample.gr,
         main="PCA of 3D coordinates", leg=TRUE, leg.pos = "top")

# In addition, verify hypothetical sound surfaces for each PC
hypo.surf(threeD.out=eig.sample, PC=1, flim=c(0, 4), tlim=c(0, 0.8),
          x.length=70, y.length=47, plot.exp = FALSE)


#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
#                      Recreate eig.sample object                             #
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#


library(seewave)
library(tuneR)

# Create temporary folder to store ".wav" files
wav.at <- file.path(base::tempdir(), "eig.sample")
if(!dir.exists(wav.at)) dir.create(wav.at)

# Create temporary folder to store results
store.at <- file.path(base::tempdir(), "eig.sample-output")
if(!dir.exists(store.at)) dir.create(store.at)

# Select three acoustic units within each sound data
data(cuvieri)
spectro(cuvieri, flim = c(0,4))
cut.cuvieri1 <- cutw(cuvieri, f=44100, from=0, to=0.5, output = "Wave")
cut.cuvieri2 <- cutw(cuvieri, f=44100, from=0.7, to=1.2, output = "Wave")
cut.cuvieri3 <- cutw(cuvieri, f=44100, from=1.4, to=1.9, output = "Wave")

data("centralis")
```

```
spectro(centralis, flim = c(0,4))
cut.centralis1 <- cutw(centralis, f=44100, from=0.1, to=0.8, output = "Wave")
cut.centralis2 <- cutw(centralis, f=44100, from=1.05, to=1.75, output = "Wave")
cut.centralis3 <- cutw(centralis, f=44100, from=2.1, to=2.8, output = "Wave")

data("kroyeri")
spectro(kroyeri, flim = c(0,4))
cut.kroyeri1 <- cutw(kroyeri, f=44100, from=0.1, to=1, output = "Wave")
cut.kroyeri2 <- cutw(kroyeri, f=44100, from=1.5, to=2.3, output = "Wave")
cut.kroyeri3 <- cutw(kroyeri, f=44100, from=2.9, to=3.8, output = "Wave")

# Export new wave files containing acoustic units and store on previosly created folder
writeWave(cut.cuvieri1, filename = file.path(wav.at, "cut.cuvieri1.wav"), extensible = FALSE)
writeWave(cut.cuvieri2, filename = file.path(wav.at, "cut.cuvieri2.wav"), extensible = FALSE)
writeWave(cut.cuvieri3, filename = file.path(wav.at, "cut.cuvieri3.wav"), extensible = FALSE)
writeWave(cut.centralis1, filename = file.path(wav.at, "cut.centralis1.wav"), extensible = FALSE)
writeWave(cut.centralis2, filename = file.path(wav.at, "cut.centralis2.wav"), extensible = FALSE)
writeWave(cut.centralis3, filename = file.path(wav.at, "cut.centralis3.wav"), extensible = FALSE)
writeWave(cut.kroyeri1, filename = file.path(wav.at, "cut.kroyeri1.wav"), extensible = FALSE)
writeWave(cut.kroyeri2, filename = file.path(wav.at, "cut.kroyeri2.wav"), extensible = FALSE)
writeWave(cut.kroyeri3, filename = file.path(wav.at, "cut.kroyeri3.wav"), extensible = FALSE)

# Place sounds at beggining of sound window before analysis
align.wave(wav.at = wav.at, wav.to = "Aligned",
           time.length = 0.8, time.perc = 0.005, dBlevel = 25)

# Verify alignment using analysis.type = "twoDshape"
eigensound(analysis.type = "twoDshape", wav.at = file.path(wav.at, "Aligned"),
           store.at = store.at, flim=c(0, 4), tlim=c(0, 0.8),
           plot.exp = TRUE, plot.as = "jpeg", dBlevel = 25)
# Go to folder specified by store.at and check jpeg files created

# Run eigensound function using analysis.type = "threeDshape" on aligned wave files
# Store results as R object
eig.sample <- eigensound(analysis.type="threeDshape", wav.at = file.path(wav.at, "Aligned"),
                         flim=c(0, 4), tlim=c(0, 0.8), dBlevel=25, plot.exp = FALSE,
                         x.length=70, y.length = 47, log.scale = TRUE)
```

---

eigensound *Sound waves onto morphometric data*

Description

eigensound is the main feature of SoundShape package. For each ".wav" file on a given folder, the fuction will compute spectrogram data and acquire semilandmarks using a 3D representation of sound (analysis.type = "threeDshape"), allowing its users to acquire, and simultaneously store

point coordinates (i.e. semilandmarkns) as an R object, and/or in TPS format – the native file format of James Rohlf's TPS series (Rohlf, 2015).

Moreover, `eigensound` also allow its user to export 2D and 3D spectrogram images (`plot.exp = TRUE`) that are helpful during the protocol for error verification and for illustrative purposes (see Rocha & Romano *in prep*). Alternativaly, `eigensound` feature the option of acquiring semilandmarks as the cross-correlation between energy quantiles and a curve of relative amplitude from 2D spectrograms (`analysis.type = "twoDshape"`; see `Details` section).

Usage

```
eigensound(
  analysis.type = NULL,
  wav.at = NULL,
  store.at = wav.at,
  dBlevel = 25,
  flim = c(0, 10),
  tlim = c(0, 1),
  trel = tlim,
  x.length = 80,
  y.length = 60,
  log.scale = TRUE,
  back.amp = 35,
  add.points = FALSE,
  add.contour = TRUE,
  lwd = 1,
  EQ = c(0.05, 0.15, 0.3, 0.5, 0.7, 0.85, 0.95),
  mag.time = 1,
  f = 44100,
  wl = 512,
  ovlp = 70,
  plot.exp = TRUE,
  plot.as = "jpeg",
  plot.type = "surface",
  rotate.Xaxis = 60,
  rotate.Yaxis = 40,
  TPS.file = NULL
)
```

Arguments

`analysis.type`	type of analysis intended. If `analysis.type = "threeDshape"`, semilandmarks are acquired from spectrogram data using a 3D representation of sound (same as in MacLeod et al., 2013). If `analysis.type = "twoDshape"` and `add.points = TRUE`, semilandmarks are acquired using energy quantiles and a 2D curve of relative amplitude. By default: `analysis.type = NULL` (i.e. method must be specified before the analysis).
`wav.at`	filepath to the folder where `".wav"` files are stored. Should be presented between quotation marks. By default: `wav.at = NULL` (i.e. user must specify the filepath to `".wav"` files)

`store.at` filepath to the folder where spectrogram plots and `tps` file will be stored. Should be presented between quotation marks. By default: `store.at = wav.at` (i.e. store outputs in the same folder as `".wav"` files)

`dBlevel` absolute amplitude value to be used as relative amplitude contour, which will serve as reference for semilandmark acquisition in both `analysis.type = "threeDshape"` and `"twoDshape"`. By default: `dBlevel = 25`

`flim` modifications of the frequency limits (Y-axis). Vector with two values in kHz. By default: `flim = c(0,10)`

`tlim` modifications of the time limits (X-axis). Vector with two values in seconds. By default: `tlim = c(0,1)`

`trel` only applies when `analysis.type = "twoDshape"`. Set the relative scale to be used on the time (X-axis); relative to the numbers displayed on the X-axis of spectrogram plots. By default: `trel = tlim`

`x.length` only applies when `analysis.type = "threeDshape"`. Length of sequence (i.e. number of cells per side on sound window) to be used as sampling grid coordinates on the time (X-axis). By default: `x.length = 80`

`y.length` only applies when `analysis.type = "threeDshape"`. Length of sequence (i.e. number of cells per side on sound window) to be used as sampling grid coordinates on the frequency (Y-axis). By default: `y.length = 60`

`log.scale` only applies when `analysis.type = "threeDshape"`. A logical. If `TRUE`, `eigensound` will use a logarithmic scale on the time (X-axis), which is recommeded when the analyzed sounds present great variation on this axis (e.g. emphasize short duration sounds). If `FALSE`, a linear scale is used instead (same as MacLeod et al., 2013). By default: `log.scale = TRUE`

`back.amp` only applies when `analysis.type = "twoDshape"` and `plot.exp = TRUE`. Absolute amplitude value to be used as background in the 2D spectrogram plot. By default: `back.amp = 35`

`add.points` only applies when `analysis.type = "twoDshape"`. A logical. If `TRUE`, `eigensound` will compute semilandmarks acquired by cross-correlation between energy quantiles (i.e. `EQ`) and a curve of relative amplitude (i.e. `dBlevel`). If `plot.exp = TRUE`, semilandmarks will be included in spectrogram plots. By default: `add.points = FALSE` (see `Details`)

`add.contour` only applies when `analysis.type = "twoDshape"` and `plot.exp = TRUE`. A logical. If `TRUE`, exported spectrogram plots will include the curves of relative amplitude at the level specified by `dBlevel`. By default: `add.contour = TRUE`

`lwd` only applies when `plot.exp = TRUE` and `add.contour = TRUE`. The line width for the curves of relative amplitude at the level specified by `dBlevel`. Same as in `par`. By default: `lwd = 1`

`EQ` only applies when `analysis.type = "twoDshape"` and `add.points = TRUE`. A vector of energy quantiles intended (with `0 < EQ < 1`). By default: `EQ = c(0.05,0.15,0.3,0.5,0.7,0.85`
Note: When dealing with narrow banded calls, consider reducing the number of quantiles to prevent errors in the analysis.

`mag.time` only applies when `analysis.type = "twoDshape"`. Optional argument for magnifying the time coordinates (X-axis). This is sometimes desired for small sound

windows (e.g. less than 1 s), in which the time coordinates will be on a different scale than that of frequency coordinates. In those cases, it is recommended to include `mag.time = 10` or `mag.time = 100`, depending on the lenght of sound window. By default: `mag.time = 1` (i.e. no magnification is performed)

`f` — sampling frequency of `Wave`'s for the analysis (in Hz). By default: `f = 44100`

`wl` — length of the window for the analysis. By default: `wl = 512`

`ovlp` — overlap between two successive windows (in %) for increased spectrogram resolution. By default: `ovlp = 70`

`plot.exp` — a logical. If `TRUE`, for each `".wav"` file on the folder indicated by `wav.at`, `eigensound` will store a spectrogram image on the folder indicated by `store.at`. Depending on the `analysis.type`, plots may consist of 2D or 3D spectrogram images. By default: `plot.exp = TRUE`

`plot.as` — only applies when `plot.exp = TRUE`. `plot.as = "jpeg"` will generate compressed images for quick inspection of semilandmarks; `plot.as = "tiff"` or `"tif"` will generate uncompressed high resolution images that can be edited and used for publication. By default: `plot.as = "jpeg"`

`plot.type` — only applies when `analysis.type = "threeDshape"` and `plot.exp = TRUE`. `plot.type = "surface"` will produce simplified 3D sound surfaces from the calculated semilandmarks (same output employed by MacLeod et al., 2013); `plot.type = "points"` will produce 3D graphs with semilandmarks as points. By default: `plot.type = "surface"`

`rotate.Xaxis` — only applies when `analysis.type = "threeDshape"` and `plot.exp = TRUE`. Rotation of the X-axis. Same as `theta` from `persp3D` (`plot3D` package). By default: `rotate.Xaxis = 60`

`rotate.Yaxis` — only applies when `analysis.type = "threeDshape"` and `plot.exp = TRUE`. Rotation of the Y-axis. Same as `phi` from `persp3D` (`plot3D` package). By default: `rotate.Yaxis = 40`

`TPS.file` — Desired name for the `tps` file containing semilandmark coordinates. Should be presented between quotation marks. **Note:** Whenever `analysis.type = "twoDshape"`, it will only work if `add.points = TRUE`. By default: `TPS.file = NULL` (i.e. prevents `eigensound` from creating a `tps` file)

Details

When `analysis.type = "twoDshape"` and `add.points = TRUE`, `eigensound` will compute semilandmarks acquired by cross-correlation between energy quantiles (i.e. `EQ`) and a curve of relative amplitude (i.e. `dBlevel`). However, this is often subtle and prone to incur in errors (e.g. bad alignment of acoustic units; inappropriate X and Y coordinates for the sound window; narrow banded calls). Therefore, a more robust protocol of error verification is achieved using `add.points = FALSE` and `add.contour = TRUE` (default), which allow for quick verification of acoustic units alignment and the shape of each curve of relative amplitude (specified by `dBlevel`).

Note

In order to store the results from `eigensound` function and proceed with the Geometric Morphometric steps of the analysis (e.g. `geomorph` package; Adams et al., 2017), one can simultaneosly

assign the function's output to an R object and/or store them as a tps file to be used by numerous softwares of geometric analysis of shape, such as the TPS series (Rohlf, 2015).

Additionally, one may also export 2D or 3D plots as jpeg (compressed image) or tiff (uncompressed image) file formats, which can be edited for publication purposes.

Author(s)

Pedro Rocha

References

Adams, D. C., M. L. Collyer, A. Kaliontzopoulou & Sherratt, E. (2017) *Geomorph: Software for geometric morphometric analyses*. R package version 3.0.5. https://cran.r-project.org/package=geomorph.

MacLeod, N., Krieger, J. & Jones, K. E. (2013). Geometric morphometric approaches to acoustic signal analysis in mammalian biology. *Hystrix, the Italian Journal of Mammalogy, 24*(1), 110-125.

Rocha, P. & Romano, P. (*in prep*) The shape of sound: A new R package that crosses the bridge between Bioacoustics and Geometric Morphometrics.

Rohlf, F.J. (2015) The tps series of software. *Hystrix 26*, 9-12.

See Also

align.wave, geomorph, seewave

Useful links:

- https://github.com/p-rocha/SoundShape
- Report bugs at https://github.com/p-rocha/SoundShape/issues

Examples

```
library(seewave)
library(tuneR)

# Create temporary folder to store ".wav" files
wav.at <- file.path(base::tempdir(), "eigensound")
if(!dir.exists(wav.at)) dir.create(wav.at)

# Create temporary folder to store results
store.at <- file.path(base::tempdir(), "eigensound-output")
if(!dir.exists(store.at)) dir.create(store.at)

# Cut acoustic units from original Wave
cut.cuvieri <- cutw(cuvieri, f=44100, from=0, to=0.9, output = "Wave")
cut.centralis <- cutw(centralis, f=44100, from=0, to=0.9, output = "Wave")
cut.kroyeri <- cutw(kroyeri, f=44100, from=0.2, to=1.1, output = "Wave")

# Export ".wav" files containing acoustic units and store on previosly created folder
writeWave(cut.cuvieri, filename = file.path(wav.at, "cut.cuvieri.wav"), extensible = FALSE)
writeWave(cut.centralis, filename = file.path(wav.at, "cut.centralis.wav"), extensible = FALSE)
writeWave(cut.kroyeri, filename = file.path(wav.at, "cut.kroyeri.wav"), extensible = FALSE)
```

```
# Create 2D spectrograms using analysis.type = "twoDshape"
eigensound(analysis.type = "twoDshape", flim=c(0, 4), tlim=c(0, 0.8),
           plot.exp=TRUE, wav.at = wav.at, store.at = store.at)

# Create 3D spectrograms using analysis.type = "threeDshape" and store point coordinates
eig.data <- eigensound(analysis.type = "threeDshape", plot.exp=TRUE,
              wav.at = wav.at, store.at = store.at, flim=c(0, 4), tlim=c(0, 0.8))
```

`hypo.surf` *Hypothetical 3D sound surfaces representing a sample of sound waves*

Description

Using the coordinates acquired by `eigensound(analysis.type = "threeDshape")`, this function creates 3D plots containing hypothetical sound surfaces that represent either the mean shape configuration (consensus), or minimum and maximum deformations relative to Principal Components in a Principal Components Analysis (PCA).

Note: The output of `hypo.surf` must be interpreted along with the ordination of Principal Components (e.g. `pca.plot`), both featuring the same object used for `threeD.out` argument. By doing so, `hypo.surf` enhance the comprehension on how sound shape changed along the ordination plot .

Usage

```
hypo.surf(
  threeD.out = NULL,
  PC = 1,
  flim = c(0, 4),
  tlim = c(0, 0.8),
  x.length = 70,
  y.length = 47,
  log.scale = TRUE,
  f = 44100,
  wl = 512,
  ovlp = 70,
  plot.exp = FALSE,
  plot.as = "jpeg",
  store.at = NULL,
  rotate.Xaxis = 60,
  rotate.Yaxis = 40,
  cex.axis = 0.5,
  cex.lab = 0.9,
  cex.main = 1.1,
  lwd = 0.1,
```

```
  xlab = "Time coordinates",
  ylab = "Frequency coordinates",
  zlab = "Amplitude",
 colkey = list(plot = TRUE, cex.clab = 0.9, cex.axis = 0.8, side = 4, length = 0.5,
    width = 0.7, labels = TRUE, tick = TRUE, lty = 1, lwd = 1, lwd.ticks = 1)
)
```

Arguments

threeD.out	the output of eigensound analysis with analysis.type = "threeDshape". By default: threeD.out = NULL (i.e. output must be specified before ploting)
PC	Principal Component intended for the plot. Alternativaly, it is also possible to create mean shape configuration (consensus) from sample PC = "mean". By default: PC = 1
flim	modifications of the frequency limits (Y-axis). Vector with two values in kHz. Should be the same employed on eigensound(analysis.type="threeDshape") By default: flim = c(0,10)
tlim	modifications of the time limits (X-axis). Vector with two values in seconds. Should be the same employed on eigensound(analysis.type="threeDshape"). By default: tlim = c(0,1)
x.length	length of sequence (i.e. number of cells per side on sound window) to be used as sampling grid coordinates on the time (X-axis). Should be the same employed on eigensound(analysis.type="threeDshape"). By default: x.length = 70
y.length	length of sequence (i.e. number of cells per side on sound window) to be used as sampling grid coordinates on the frequency (Y-axis). Should be the same employed on eigensound(analysis.type="threeDshape"). By default: y.length = 47
log.scale	a logical. If TRUE, hypo.surf will use a logarithmic scale on the time (X-axis), which is recommeded when the analyzed sounds present great variation on this axis (e.g. emphasize short duration sounds). If FALSE, a linear scale is used instead (same as MacLeod et al., 2013). Should be the same employed on eigensound(analysis.type="threeDshape"). By default: log.scale = TRUE
f	sampling frequency of ".wav" files (in Hz). Should be the same employed on eigensound(analysis.type="threeDshape"). By default: f = 44100
wl	length of the window for the analysis. Should be the same employed on eigensound(analysis.type="tl By default: wl = 512
ovlp	overlap between two successive windows (in %) for increased spectrogram resolution. Should be the same employed on eigensound(analysis.type="threeDshape"). By default: ovlp = 70
plot.exp	a logical. If TRUE, exports the 3D output plot containing mean shape (PC = "mean"), or minimum and maximum deformations for the desired Principal Component (e.g. PC = 1). Exported plot will be stored on the folder indicated by store.at. By default: plot.exp = FALSE

`plot.as` only applies when `plot.exp = TRUE`. `plot.as = "jpeg"` will generate compressed images for quick inspection; `plot.as = "tiff"` or `"tif"` will generate uncompressed high resolution images that can be edited and used for publication. By default: `plot.as = "jpeg"`

`store.at` only applies when `plot.exp = TRUE`. Filepath to the folder where output plots will be stored. Should be presented between quotation marks. By default: `store.at = NULL` (i.e. user must specify the filepath where plots of hypothetical sound surfaces will be stored)

`rotate.Xaxis` rotation of the X-axis. Same as `theta` from `persp3D` (`plot3D` package). By default: `rotate.Xaxis = 60`

`rotate.Yaxis` rotation of the Y-axis. Same as `phi` from `persp3D` (`plot3D` package). By default: `rotate.Yaxis = 40`

`cex.axis` similarly as in `par`, magnification to be used for axis values. By default: `cex.axis = 0.9`

`cex.lab` similarly as in `par`, magnification to be used for x and y labels. By default: `cex.lab = 1.2`

`cex.main` similarly as in `par`, magnification to be used for main titles. By default: `cex.main = 1.3`

`lwd` Similarly as in `par`, intended line width for sampling grid. By default: `lwd = 0.1`

`xlab` a character string indicating the label to be written on the *x*-axis. By default: `xlab="Time coordinates"`

`ylab` a character string indicating the label to be written on the *y*-axis. By default: `ylab="Frequency coordinates"`

`zlab` a character string indicating the label to be written on the *z*-axis. By default: `zlab="Amplitude"`

`colkey` Similarly as `plot3D`, a list with parameters for the color key (legend). By default: `colkey = list(plot = TRUE, cex.clab = 0.9, cex.axis = 0.8, side = 4, length = 0.5, width = 0.7, labels = TRUE, tick = TRUE, lty = 1, lwd = 1, lwd.ticks = 1)`. See also `colkey`

Note

Some of codes from `hypo.surf` were adapted from `plotTangentSpace` (`geomorph` package). More specifically, the chunk related to the acquisition of hypothetical point configurations for each Principal Component (calculated by `prcomp`, `stats` package) is exactly the same as in `plotTangentSpace`. However, the hypothetical configurations from `plotTangentSpace` are ploted along with an ordination of PCs, whereas `hypo.surf` focuses solely on hypothetical 3D surfaces that represent minimum, maximum or mean deformations relative to each PCs (see also Rocha & Romano *in prep*).

Author(s)

Pedro Rocha

See Also

plotTangentSpace, geomorph, eigensound, pca.plot

Useful links:

- https://github.com/p-rocha/SoundShape
- Report bugs at https://github.com/p-rocha/SoundShape/issues

Examples

```
data(eig.sample)

# PCA using three-dimensional semilandmark coordinates
pca.eig.sample <- stats::prcomp(geomorph::two.d.array(eig.sample))

# Verify names for each acoustic unit and the order in which they appear
dimnames(eig.sample)[[3]]

# Create factor to use as groups in subsequent ordination plot
sample.gr <- factor(c(rep("centralis", 3), rep("cuvieri", 3), rep("kroyeri", 3)))

# Clear current R plot to prevent errors
grDevices::dev.off()

# Plot result of Principal Components Analysis
pca.plot(PCA.out = pca.eig.sample, groups = sample.gr, conv.hulls = sample.gr,
         main="PCA of 3D coordinates", leg=TRUE, leg.pos = "top")

# Verify hypothetical sound surfaces using hypo.surf
hypo.surf(threeD.out=eig.sample, PC=1, flim=c(0, 4), tlim=c(0, 0.8), x.length=70, y.length=47)
```

---

kroyeri *Vocalization of the frog* Physalaemus kroyeri

Description

Recording of a series of three stereotyped calls emitted by a male frog *Physalaemus kroyeri* (Amphibia, Anura, Leptodactylidae). Edited from original ".wav" file for optimal sinal to noise ratio and reduced time duration.

Usage

```
data(kroyeri)
```

Format

An object of the class "Wave" (tuneR package).

Details

Duration = 3.91 s. Sampling Frequency = 44100 Hz.

Recorded at Ilhéus Municipality, Bahia State, Brazil, on 05 August 1972. Air temperature 24ºC.

Source

Original recording housed at Fonoteca Neotropical Jacques Vielliard (FNJV-0032047). Recorded by Werner Bokermann.

Examples

```
data(kroyeri)

seewave::oscillo(kroyeri)
seewave::spectro(kroyeri)
threeDspectro(kroyeri,tlim=c(0, 1), flim=c(0, 4), samp.grid=FALSE, dBlevel=25)
```

`pca.plot` *Plot ordination of Principal Components with convex hulls*

Description

Ordination of Principal Components from the output of a Principal Components Analysis performed by `prcomp` function (`stats` package). Require a `factor` object with `groups`, which enable the plot to feature colored groups and convex hulls (if desired).

Usage

```
pca.plot(
  PCA.out = NULL,
  groups = NULL,
  col.gp = grDevices::rainbow(length(levels(groups))),
  conv.hulls = NULL,
  col.conv = grDevices::rainbow(length(levels(conv.hulls))),
  PCs = c(1, 2),
  main = "Ordination of PCA coordinates",
  sp.as = "points",
  sp.text = NULL,
  cross.origin = TRUE,
  lwd = 1,
  lty = "dotted",
  leg = TRUE,
  leg.labels = groups,
  leg.pos = "topright",
  cex.leg = 1,
  cex = 1,
  cex.axis = 1,
```

```
  cex.lab = 1,
  cex.main = 1
)
```

Arguments

PCA.out	the output of a Principal Components Analysis performed by prcomp (stats package). By default: PCA.out = NULL (i.e. output must be specified before ploting)
groups	groups to use as colors and/or convex hulls. Must be a factor object with the same length as the number of rows in the coordinates of PCA.out (i.e. length(groups) == nrow(PCA.out$x)). By default: groups = NULL (i.e. factor must be specified before ploting)
col.gp	a factor object with the colours intended for groups. Must be the same length as the number of levels from groups (i.e. length(col.gp) == length(levels(groups))). By default: col.gp = grDevices::rainbow(length(levels(groups))) (i.e. colors defined automatically)
conv.hulls	groups to use for convex hulls. Must be a factor object with the same length as the number of rows in the coordinates of PCA.out (i.e. length(conv.hulls) == nrow(PCA.out$x)). By default: conv.hulls = NULL (i.e. plot without convex hulls)
col.conv	a factor object with the colours intended for conv.hulls. Must be the same length as the number of levels in conv.hulls (i.e. length(col.conv) == length(levels(conv.hulls)) By default: col.conv = grDevices::rainbow(length(levels(conv.hulls))) (i.e. colors defined automatically)
PCs	a vector of length two with the Principal Components intended for the plot. By default: PCs = c(1,2)
main	main title of output plot. Should be presented between quotation marks. By default: main = "Ordination of PCA coordinates"
sp.as	enables one to choose between ploting elements as "points" or "text". If sp.as = "text", also input a factor of characters to use as text (i.e. sp.text). By default: sp.as = "points"
sp.text	only applies when sp.as = "text". A factor including elements as texts intended in the plot. Has to be the same length as the number of rows in the coordinates of PCA.out (i.e. length(sp.text) == nrow(PCA.out$x)) . By default: sp.text = NULL
cross.origin	A logical. If TRUE, the ordination plot will include lines crossing at the origin (i.e. x = 0, y = 0). By default: cross.origin = TRUE
lwd	only applies when cross.origin = TRUE. The width for lines crossing at the origin. Same as in par. By default: lwd = 1
lty	only applies when cross.origin = TRUE. The type of lines crossing at the origin. Same as in par: "Line types can either be specified as an integer (0=blank, 1=solid (default), 2=dashed, 3=dotted, 4=dotdash, 5=longdash, 6=twodash) or as one of the character strings "blank", "solid", "dashed", "dotted", "dotdash", "longdash", or "twodash", where "blank" uses 'invisible lines' (i.e., does not draw them)." By default: lty = "dotted"

leg	a logical. If TRUE, a legend is added to plot. By default: leg = TRUE
leg.labels	only applies when leg = TRUE. Must be the same length of levels in groups (i.e. length(leg.labels) == length(levels(groups))). By default: leg.labels = groups (i.e. labels will be the same as the levels from groups)
leg.pos	only applies when leg = TRUE. Specify legend location in the plot by using a keyword from the list: "bottomright", "bottom", "bottomleft", "left", "topleft", "top", "topright", "right" and "center". Alternatively, a single value can be provided and used for both margins, or two values, in which the first is used for X distance, and the second for Y distance. See also legend for details. By default: leg.pos = "topright"
cex.leg	only applies when leg = TRUE. The magnification to be used in the legend. By default: cex.leg = 1
cex	same as in par: "A numerical value giving the amount by which plotting text and symbols should be magnified relative to the default". By default: cex = 1
cex.axis	same as in par. Relative to cex. The magnification to be used for axis annotation. By default: cex.axix = 1
cex.lab	same as in par. Relative to cex. The magnification to be used for x and y labels. By default: cex.lab = 1
cex.main	same as in par. Relative to cex. The magnification to be used for main title. By default: cex.main = 1

Value

Require the output of prcomp and a vector with groups to plot. In addition, it is also possible to include convex hulls around each group (i.e. conv.hulls) and to control the colors intended for each group (i.e. col.gp) and for each convex hull (i.e. col.conv).

Author(s)

Pedro Rocha

References

Rocha, P. & Romano, P. (*in prep*) The shape of sound: A new R package that crosses the bridge between Bioacoustics and Geometric Morphometrics.

See Also

prcomp, palette, rgb, rainbow, legend

Useful links:

- https://github.com/p-rocha/SoundShape
- Report bugs at https://github.com/p-rocha/SoundShape/issues

Examples

```
data(eig.sample)

# PCA using 3D semilandmark coordinates
pca.eig.sample <- stats::prcomp(geomorph::two.d.array(eig.sample))

# Verify names for each acoustic unit and the order in which they appear
dimnames(eig.sample)[[3]]

# Create factor to use as groups in subsequent ordination plot
sample.gr <- factor(c(rep("centralis", 3), rep("cuvieri", 3), rep("kroyeri", 3)))

# Clear current R plot to prevent errors
grDevices::dev.off()

# Plot result of Principal Components Analysis
pca.plot(PCA.out = pca.eig.sample, groups = sample.gr, conv.hulls = sample.gr,
         main="PCA of 3D coordinates", leg=TRUE, leg.pos = "top")
```

SoundShape *Sound Waves Onto Morphometric Data*

Description

SoundShape package provide the tools required to implement a promising, and yet little explored method for bioacoustical analysis, the so called eigensound protocol developed by MacLeod, Krieger, & Jones (2013). Eigensound was developed for taxonomy-based bioacoustics and focuses on the comparison between acoustic units from different species. The method consists on applying a sampling grid over the 3D representation of sound (i.e. spectrogram data) and then translate the spectrogram into a dataset that can be analyzed similarly to 3D coordinate sets used in Geometric Morphometrics Methods, thus enabling the direct comparison between stereotyped calls from different species. For more information on SoundShape and the eigensound method, see Rocha & Romano (*in prep*) and MacLeod et al. (2013).

The following set of functions crosses the bridge between Bioacoustics and Geometric Morphometrics:

- `align.wave`: Automatic placement of calls at the beggining of a sound window.
- `eigensound`: Calculate spectrogram data for each `".wav"` file on a given folder and acquire semilandmarks using a 3D representation of sound.
- `pca.plot`: Plot ordination of Principal Components with convex hulls.
- `hypo.surf`: Hypothetical 3D plots of sound surfaces representing a sample of sounds submited to `eigensound`.
- `threeDspectro`: Colorful 3D spectrograms from a single object of class `"Wave"`.

Author(s)

Maintainer: Pedro Rocha <p.rocha1990@gmail.com>

Other contributors:

- Pedro Romano <psrromano@gmail.com> [contributor]

See Also

Useful links:

- https://github.com/p-rocha/SoundShape
- Report bugs at https://github.com/p-rocha/SoundShape/issues

---

`threeDspectro` *3D spectrogram plots from* `"Wave"` *objects*

Description

Create colorful spectrograms from a single object of class `"Wave"`.

This function works similarly as `spectro` (`seewave` package), with spectrogram data actually computed by `spectro`. However, the 3D plot is generated by `persp3D` (`plot3D` package).

Usage

```
threeDspectro(
  wave,
  tlim = NULL,
  flim = NULL,
  samp.grid = FALSE,
  plot.type = "surface",
  x.length = 100,
  y.length = 70,
  lwd = 0.1,
  plot.exp = FALSE,
  log.scale = FALSE,
  cex = 0.5,
  cex.axis = 0.5,
  cex.lab = 0.9,
  cex.main = 1,
  store.at = NULL,
  plot.as = "jpeg",
  color = seewave::spectro.colors(80),
  f = 44100,
  wl = 512,
  ovlp = 70,
  dBlevel = 30,
```

```
  resfac = 1,
  rotate.Xaxis = 60,
  rotate.Yaxis = 40,
  main = "Spectrogram 3D",
  scalelab = expression("Amplitude (dB)"),
 colkey = list(plot = TRUE, cex.clab = 0.8, cex.axis = 1, side = 4, length = 1, width
    = 1, labels = TRUE, tick = TRUE, lty = 1, lwd = 1, lwd.ticks = 1)
)
```

Arguments

wave	an R object of the class "Wave"
tlim	modifications of the time limits (X-axis). Vector with two values in seconds. By default: tlim = NULL
flim	modifications of the frequency limits (Y-axis). Vector with two values in kHz. By default: flim = NULL
samp.grid	a logical. If TRUE, a sampling grid with dimensions x.length and y.length is applied over the three-dimensional surface. An amplitude value is collected at each node in the grid and a new matrix of amplitude coordinates is used in the plot. Recomended for faster plots and as protocol for error verification in point acquisition. See also plot.type. By default: samp.grid = FALSE
plot.type	only applies when samp.grid = TRUE. plot.type = "surface" will produce a simplified 3D spectrogram surface using the amplitude values colected by the sampling grid (same output employed by MacLeod et al., 2013). plot.type = "points" will produce 3D graphs with semilandmarks as points. By default: plot.type = "surface"
x.length	only applies when samp.grid = TRUE. Length of sequence (i.e. number of cells per side on sound window) to be used as sampling grid coordinates on the time (X-axis). By default: x.length = 100
y.length	only applies when samp.grid = TRUE. Length of sequence (i.e. number of cells per side on sound window) to be used as sampling grid coordinates on the frequency (Y-axis). By default: y.length = 70
lwd	only applies when samp.grid = TRUE and plot.type = "surface". Similarly as in par, intended line width for sampling grid. By default: lwd = 0.1
plot.exp	a logical. If TRUE, 3D spectrogram plot from wave object is exported and stored on the folder indicated by store.at. By default: plot.exp = FALSE
log.scale	only applies when samp.grid = TRUE. A logical. If TRUE, threeDspectro will use a logarithmic scale on the time (X-axis), which is recommeded when the analyzed sounds present great variation on this axis (e.g. emphasize short duration sounds). If FALSE, a linear scale is used instead (same as MacLeod et al., 2013). By default: log.scale = FALSE
cex	only applies when samp.grid = TRUE and plot.type = "points". Similarly as in par, intended size for points. By default: cex = 0.5
cex.axis	Similarly as in par, the magnification to be used for axis annotation. By default: cex.axis = 0.5

cex.lab	Similarly as in par, the magnification to be used for x and y labels. By default: cex.lab = 0.9
cex.main	Similarly as in par, the magnification to be used for main titles. By default: cex.main = 1
store.at	only applies when plot.exp = TRUE. Filepath to the folder where 3D plot will be stored. Should be presented between quotation marks. By default: store.at = NULL (i.e. user must specify the filepath where spectrogram plots will be stored)
plot.as	only applies when plot.exp = TRUE. plot.as = "jpeg" will generate compressed images for quick inspection; plot.as = "tiff" or "tif" will generate uncompressed high resolution images that can be edited and used for publication. By default: plot.as = "jpeg"
color	Color palette to be used for the amplitude (Z-axis). Same default as spectro: color=spectro.colors(80). See also Details section.
f	sampling frequency of Wave object (in Hz). By default: f = 44100
wl	length of the window for spectrogram calculation. By default: wl = 512
ovlp	overlap between two successive windows (in %) for increased spectrogram resolution. By default: ovlp = 70
dBlevel	absolute amplitude value to be used as relative background on 3D plot. Same as dBlevel from eigensound and align.wave. By default: dBlevel = 30
resfac	resolution factor, in which an value > 1 will increase the resolution. Can be one value or a vector of two numbers, for the x and y values, respectively. **Note:** Same as in persp3D (plot3D package). By default: resfac = 1
rotate.Xaxis	rotation of the X-axis. Same as theta from persp3D (plot3D package). By default: rotate.Xaxis = 60
rotate.Yaxis	rotation of the Y-axis. Same as phi from persp3D (plot3D package). By default: rotate.Yaxis = 40
main	main title of output plot. Should be presented between quotation marks. By default: main = "3D spectrogram"
scalelab	Similarly as plot3D, the label to be written on top of the color key. Should be a character string wrapped by expression(). By default: scalelab = expression("Amplitude (dB)"). See also colkey
colkey	Similarly as plot3D, a list with parameters for the color key (legend). By default: colkey = list(plot = TRUE, cex.clab = 0.8, cex.axis = 1, side = 4, length = 1, width = 1, labels = TRUE, tick = TRUE, lty = 1, lwd = 1, lwd.ticks = 1). See also colkey

Details

Similarly as spectro (seewave package), any colour pallete can be used to describe the amplitude (Z-axis). Some suggestions: seewave::temp.colors, seewave::spectro.colors, seewave::reverse.heat.colors, se

Author(s)

Pedro Rocha

References

MacLeod, N., Krieger, J. & Jones, K. E. (2013). Geometric morphometric approaches to acoustic signal analysis in mammalian biology. *Hystrix, the Italian Journal of Mammalogy, 24*(1), 110-125.

Rocha, P. & Romano, P. (*in prep*) The shape of sound: A new R package that crosses the bridge between Bioacoustics and Geometric Morphometrics.

See Also

spectro, seewave, eigensound, align.wave, persp3D, plot3D, align.wave

Useful links:

- https://github.com/p-rocha/SoundShape
- Report bugs at https://github.com/p-rocha/SoundShape/issues

Examples

```
# As simple as this
threeDspectro(centralis)
threeDspectro(cuvieri)
threeDspectro(kroyeri)

# Controling some arguments
threeDspectro(cuvieri, tlim=c(0, 0.5), flim=c(0, 4))
threeDspectro(cuvieri, tlim=c(0, 0.5), flim=c(0, 4), samp.grid=FALSE)
threeDspectro(cuvieri, tlim=c(0, 0.5), flim=c(0, 4), samp.grid=FALSE, dBlevel=60)

# Try different colors
threeDspectro(cuvieri, color=seewave::reverse.terrain.colors(80),
              samp.grid=FALSE, tlim=c(0, 0.5), flim=c(0, 4))
threeDspectro(cuvieri, color=seewave::reverse.cm.colors(80),
              samp.grid=FALSE, tlim=c(0, 0.5), flim=c(0, 4))
threeDspectro(cuvieri, color=grDevices::heat.colors(80),
              samp.grid=FALSE, tlim=c(0, 0.5), flim=c(0, 4))

# Rotation
threeDspectro(cuvieri, tlim=c(0, 0.5), flim=c(0, 4), rotate.Xaxis=40, rotate.Yaxis=50)

# Export your graph
threeDspectro(cuvieri, plot.exp=TRUE, store.at=tempdir(), tlim=c(0,0.5), flim=c(0,4))
```

Index