Package 'trackter'

April 19, 2021

Title Automated Kinematic Analysis of Image Data

Version 0.1.1

Description Analysis of shape and contours in regions of interest (ROIs) in image sequences and extracting midline and other kinematic data.

License GPL (>= 2)

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0),

biocViews

Imports EBImage, data.table, dplyr, Momocs, raster, zoo, plyr, pastecs, features, jpeg, ggplot2, graphics, stats, utils

RoxygenNote 7.1.1

Suggests testthat (>= 3.0.1), covr, knitr, rmarkdown, animation,

SystemRequirements FFmpeg

VignetteBuilder knitr

NeedsCompilation no

Author Christopher Kenaley [aut, cre]

Maintainer Christopher Kenaley <cpkenaley@gmail.com>

Repository CRAN

Date/Publication 2021-04-18 22:10:02 UTC

R topics documented:

`:=` *Assignment by reference*

Description

Assignment by reference

Usage

```
`:=`(...)
```

Arguments

`...`	See ? '...'

`amp.freq` *Computes amplitude and frequency of wave-like data*

Description

Computes amplitude(s) and wavelength(s) of a wave form, amongst other things, based on a sampling frequency

Usage

```
amp.freq(x = NULL, y, sf = 100)
```

Arguments

`x`	Numeric; x position (or sample number)
`y`	numeric; y position
`sf`	numeric; sample frequency (i.e., how often was x and y sampled) in Hz

Value

a list with amplitude "a", frequency "f", amplitude returned from a smoothed sign function "a.f" based on output from `features`, signal to noise ratio "snr".

See Also

`features`

Examples

```
#Compute waveform patterns
x <- seq(0,pi,0.1)
y <- sin(x^1.3*pi)
plot(x,y)

amp.freq(x=x,y=y)
```

`bearing.xy` *Computes the heading between to cartesian points*

Description

Computes the heading (radians counter clockwise from north or vertical)

Usage

```
bearing.xy(x1, x2, y1, y2)
```

Arguments

`x1`	Numeric; point A x value
`x2`	numeric; point B x value
`y1`	numeric; point A y value
`y2`	numeric; point B y value

Value

A single value in radians

a single value in radians

Examples

```
#example
A <- c(0,0)
B <- c(-5,5)
thet <- bearing.xy(A[1],B[1],A[2],B[2])
deg(thet)
```

`cosine.ang` *Computes angle between two segments sharing a point.*

Description

Computes angle between two segments of a triangle using law of cosines.

Usage

```
cosine.ang(l, r, o)
```

Arguments

`l`	Numeric; length of segment to the left
`r`	Numeric; length of segment to the right
`o`	Numeric; length of opposite segment

Value

A single value of the angle in radians

Examples

```
#a right triangle
L=3
R=3
O=sqrt(L^2+R^2)
cosine.ang(L,R,O)
```

deg *Converts radians to degrees*

Description

Converts radians to degrees

Usage

```
deg(x)
```

Arguments

x	Numeric; value in radians

Value

A single value

See Also

rad

dist.2d *Computes distance between two points in Cartesian space.*

Description

Computes distance between two points in Cartesian space using simple trigonometry functions

Usage

```
dist.2d(x1, x2, y1, y2)
```

Arguments

x1	Numeric; x position of coordinate 1
x2	numeric; x position of coordinate 2
y1	numeric; y position of coordinate 1
y2	numeric; y position of coordinate 2

Value

A single value of the distance between p[x1,y1] and p[x2,y2]

Examples

```
#Find the lengths of the sides of a tringle and print to plot
x <- c(0,3,2)
y <- c(0,3,0)
plot(x,y)
lines(x,y)
lines(x[c(1,3)],y[c(1,3)])
hyp <- dist.2d(x[1],x[2],y[1],y[2])
s1 <- dist.2d(x[1],x[3],y[1],y[3])
s2 <- dist.2d(x[2],x[3],y[2],y[3])
text(mean(x[1:2],mean(y[2:3])),labels=round(hyp,1))
text(mean(x[c(1,3)]),y[1]+0.25,labels=round(s1,1))
text(mean(x[c(2:3)]),mean(y[2:3]),labels=round(s2,1))
```

`fin.kin` *Tracking of fin-like extensions of body contours*

Description

Estimates the amplitudes of regions along a body contour that are protruding. Useful in computing paired-fin amplitudes from contour data produced from kin.simple and kin.search. Also computes a smoothed midline based on the body outline with the fin region removed.

Usage

```
fin.kin(
  x,
  fin.pos = NULL,
  smooth.n = 50,
  tip.ang = 10,
  smoothing = "loess",
  x.bins = 0.2,
  ml.smooth = 0.25
)
```

Arguments

`x`	a data frame or matrix with 'x' and 'y' data as columns.
`fin.pos`	numeric, a vector of length 2 indicating the start and end of the contour region that contains the fins of interest as a proportion of the body length.
`smooth.n`	numeric, the number of smoothing operations undertaken by coo_smooth on the contour described by 'x'.
`tip.ang`	the minimum angle, in degrees, that defines tip of each fin. See Details.
`smoothing`	character, the midline smoothing method, either 'loess' or 'spline'.
`x.bins`	numeric, when less than or equal to 1, the proportion of contour coordinates to sample for midline estimation. If greater than 1, the absolute number of equally spaced x values from which to compute the midline. See Details.

`ml.smooth` numeric the smoothing value for the midline. If `smoothing` is set to 'loess', passed to 'span' value for `loess`. If `smoothing` is set to 'spline', passed to 'spar' value for `smooth.spline`

Details

The algorithm assumes a left-right orientation, i.e., the head of the contour is left. If otherwise oriented, contour can be flipped with `coo_flipx` and `coo_flipy` after converting contour to class `coo`.

`tip.angle` is used to define the tip of the fin, assuming that the tip of the fin is pointed and, for a sufficiently smoothed fin contour, will have contour edges that form the highest angles within the fin region defined by `fin.pos`. Low values of `smooth.n` ($<$5) should be avoided if the contour is jagged, perhaps due to digitization.

In addition to fin amplitude and contour extraction, also produces a composite contour of the body minus the fin area described by `fin.pos`. Fin contours are replaced by a simple linear prediction constructed from the coordinates of the first and last values covered by `fin.pos`. The result is a straight line between the start and end of each fin. From this composite body contour, a midline prediction is made based on the method indicated by `smoothing` and number of points indicated by `x.bins`.

`x.bins` controls the bin size of x values used to estimate the midline. From these bins, mean x and the range of y is calculated. The midpoint at each mean x is then calculated from the mid point of y. When less then 1, `x.bins` values approaching 1 may result in poor a midline as x values on one side of the contour may not have corresponding identical values on the other. Values closer to 0 will result in fewer points but a more robust midline. Higher `smooth.n` values will also result in a more robust midline estimation (but also a loss of contour information).

Value

A list with the following components:

`body` a data table consisting of x,y coordinates of the body contour

`fin` a data table describing the contour of the fins consisting of the following:

- x,y coordinates within the range of `fin.pos`
- 'ang': the angle formed by each coordinate and its adjacent points.
- 'fin': fin side, 'L' or 'R'
- 'y.pred': predicted y values according to `lm()` from start to end of fins.

`fin.pts` a data table describing fin position consisting of the following:

- x,y coordinates of the fin tips, start, and end within the range of `fin.pos`.
- 'ang': the angle formed by the coordinates and their adjacent points.
- 'pos': description of the coordinates' positions, 'start', 'end' or 'tip'.

`comp` a data table describing the composite contour of the body minus the fins.

- x,y coordinates of the body except the range of x values within `fin.pos`. These values take on a straight line described by the prediction of `lm()` based on the start and end of the fin. See Details.

midline a data table describing the estimated

- 'x': the mean x position within the bin.
- 'x.bin': the x bin in cut notation.
- 'y.m': the y midpoint at the bind and mean x value.
- 'ml.pred': the y midline value according to the smoothing parameters.

See Also

kin.simple, kin.LDA, kin.search, efourier, coo_angle_edges, coo_smooth, loess, smooth.spline

Examples

```
###plot pectoral-fin amplitudes of a swimming sunfish
## Not run:
require(ggplot2)

#download example avi video
f <- "https://github.com/ckenaley/exampledata/blob/master/sunfish_pect.avi?raw=true"
download.file(f,"sunfish.avi")

#extract images with ffmpeg opereations and reduce them to 600 px wide with a filter
filt.red <- " -vf scale=600:-1 " #filter
vid.to.images2(vid.path="sunfish.avi",filt = filt.red) #extract

#number of frames
fr <- length(list.files("images"))
#extract contours and other data
kin <- kin.simple(image.dir = "images",frames=c(1:fr),thr=0.9,ant.per = 0.25)
#fin amplitudes by frame with data.table
fin.pos <- c(0.25,.5)
fin.dat <- kin$cont[, { f <- fin.kin(data.frame(x=x,y=y),fin.pos =fin.pos);
list(amp=f$amp$amp,fin=f$amp$fin,amp.bl=f$amp$amp.bl)},by=list(frame)]
p <- ggplot(dat=fin.dat,aes(x=frame,y=amp,col=fin))+geom_line()+theme_classic(15)
print(p)


## plot body and fin contours of frame 1
cont <- data.frame(x=kin$cont[frame==2,list(x,y)]$x,y=kin$cont[frame==2,list(y)]$y)
fins <- fin.kin(cont,fin.pos =fin.pos,x.bins=100)

#plot body contour and fins
p <- qplot(data=fins$body,x=x,y=y)+geom_point(data=fins$fin,aes(x,y),col="red",size=3)
p+geom_point(data=fins$fin.pts,aes(x,y,shape=pos))+xlim(c(0,kin$dim[1]))+ylim(c(0,kin$dim[2]))

#plot body contour minus fins and the body midline
p <- qplot(data=fins$comp,x=x,y=y)+geom_point(data=fins$midline,aes(x,ml.pred),col="red",size=2)
p+xlim(c(0,kin$dim[1]))+ylim(c(0,kin$dim[2]))


## End(Not run)
```

`fishshapes` *Contour data of fish and arbitrary shapes*

Description

Contour data constructed with the `Momocs` package of several fish shapes and arbitrary polygons. Intended for training of PCA and LDA analysis with `LDA`. Shapes are classified with shape, type, and edge factors.

Usage

```
data(fishshapes)
```

Format

An object of class `"Out"` and `"Coo"`; see `Out`.

Details

- 'shape': names the contour
- 'type': classifies the shape as 'fish' or not.fish'
- 'edge': is every coordinate of the contour represented, i.e., not cut off by an edge of the field? One level of 'FALSE'.

Shapes of type 'fish' include the follow classification levels:

- 'eel' (genus Anguilla) swimming with body undulations
- 'sunfish_BCF' (genus Lepomis) swimming with body-caudal fin propulsion
- 'sunfish_pect' (genus Lepomis) swimming with both body-caudal fin and pectoral fin oscillations
- 'trout' (genus Oncorhynchus) swimming with body-caudal fin propulsion

Each of these fish types include contours sampled regularly over one tail-beat cycle.

Shapes of type 'not.fish' include 'Ellipse', 'Lshape', 'Ushape', Ushape2', 'Ushape3','triangle', 'rectangle', and 'square'. Each 'not.fish' type was resampled 6 times with different `efourier` analyses with 'nb.h' values ranging from 5 to 30. This produces shape classes with subtly variable contours.

The 'edge' factor is included so as to have the classification factors match those of the `efourier` and `LDA` analysis in `kin.LDA`.

Examples

```
library(Momocs)
data(fishshapes)
panel(fishshapes)
```

`halfwave` *Compute half wavelengths from a sine-like waveform*

Description

Computes half wavelengths and their positions and amplitude from a sine-like waveform based on either peak-to-trough or internodal distance.

Usage

```
halfwave(
  x,
  y,
  method = "zeros",
  zero.begin = TRUE,
  fit = TRUE,
  dens = 10,
  smooth = 0.1,
  smoothing = "loess"
)
```

Arguments

`x`	Numeric; x position
`y`	numeric; y position
`method`	character; how half waves should be found and classified, where it crosses zero/the internodal length ("zeros") or peak to trough/trough to peak ("p2t"). See Details.
`zero.begin`	logical; does wave begin at zero? Default is 'TRUE' and will help find waves beginning at first x,y values if y=0
`fit`	logical; if 'method="zeros"', should zeros be detected by a fitting operation. See Details.
`dens`	numeric; factor by which to increase the sample density used in fitting when 'method="zeros"'. See Details.
`smooth`	numeric; if `smoothing` is set to 'loess', 'span' parameter value for `loess`. If `smoothing` is set to 'spline' 'spar' parameter value for `smooth.spline`
`smoothing`	character; the smoothing method when 'fit=TRUE', either 'loess' or 'spline'. See Details.

Details

If 'method="p2t"', half waves are found using critical points (i.e., local maxima and minima) with `features`. Detected half waves with this method can be either peak to trough or trough to peak.

If 'method="zeros"' and 'fit=TRUE', zero crossings are determined by first increasing the sample density by a factor determined by `dens`. A more dense `loess` or `smooth.spline` model is then

fit to the data and new y values predicted. Wave positions and lengths are determined based on these predicted values. This option should be useful when the sampling density of the waveform is relatively low and therefor detected wave positions and zero crossings (the internodes) may be rather coarse.

Value

A list with the following components:

`method` the method chosen to find half waves

`names` a data table with columns 'x', 'y', and 'wave' describing the x and y positions of the wave and a numeric name of each half wave detected, resptively. If 'method="zeros"' and 'fit=TRUE', these values reflect the predicted, more dense data as determined by `smoothing`,`smooth`, and `dens`.

`dat` a data table describing each have wave detected.

- 'zeros': x value where y crosses zero. Returns NA if 'method=p2t'.
- 'wave.begin': x value where each half wave begins.
- 'wave.end': x value where each half wave ends.
- 'begin.index': x index of where each half wave begins.
- 'end.index': x index of where each half wave ends.
- 'wave': numeric name of each wave.
- 'l': the length of each half wave.
- 'amp1': If method is set to 'p2t' this is the begin amplitude. If "method='zeros'", this is the maximum absolute amplitude between internodes.
- 'amp2': If method is set to 'p2t', this is the end amplitude. If "method='zeros' value is NA.
- 'pos1': If method is set to 'p2t', the x position of begin amplitude for each half wave and identical to 'begin'. If "method='zeros'", the position of maximum absolute amplitude between the internodes.
- 'pos2': If method is set to 'p2t', the x position of end amplitude for each half wave and identical to 'end'. If "method='zeros'", value is NA

If 'method="zeros"' and 'fit=TRUE', these values reflect the predicted, more dense data as determined by `smoothing`, `smooth`, and `dens`.

See Also

`features`, `loess`, `smooth.spline`

Examples

```
require(ggplot2)

#Find length of the half waves
x <- seq(0,pi,0.01)
y <- sin(x^2*pi)
qplot(x,y)
```

```
#zero method predicting zeros
w.z <- halfwave(x,y,method="zeros",fit=TRUE,smoothing="spline")

#plot waveform with detected half waves using fitted 'zeros' method
p <- ggplot()+geom_point(aes(x=x,y=y))
p <- p+geom_line(data=w.z$names,aes(x=x,y=y,col=wave),alpha=0.4,size=3,inherit.aes=FALSE)
p+theme_classic()

#plot lambda as it varies with position
qplot(data=w.z$dat,x=pos1,y=l)

#peak-to-trough method
w.p <- halfwave(x,y,method="p2t")
qplot(data=w.p$names,x=x,y=y,col=wave)
```

`images.to.video` *Stitches images into a video file*

Description

Stitches images into a video file of type indicated by "vid.ext"

Usage

```
images.to.video(
  image.dir = NULL,
  out.dir = NULL,
  vid.name = NULL,
  qual = 50,
  frame.rate = 10,
  overwrite = FALSE,
  silent = TRUE
)
```

Arguments

`image.dir`	character; directory containing images to stitch.
`out.dir`	character; directory in which to store video.
`vid.name`	character; file name given to video including extension. mp4 currently works best.
`qual`	numeric; the quality of the video rendered from 1-100%. Defaults to 50%.
`frame.rate`	numeric; video frame rate in fps.
`overwrite`	logical; should path described by vid.name be overwritten if it exhists.
`silent`	logical; should output of `system` call for ffmpeg operation be suppressed.

Details

Assumes images are appended with a numeric sequence.

Value

Outputs a video of name "video.name+vid.ext".

See Also

vid.to.images

Examples

```
#make some images

dir.create(paste0(tempdir(),"/images")) #make a directory to store images

a <- 2
b <- 3
theta <- seq(0,10*pi,0.01)
r <- a + b*theta
df <- data.frame(x=r*cos(theta), y=r*sin(theta)) # Cartesian coords
every.i <- 30
for(i in seq(1,length(theta),30)) {
 jpeg(paste0(tempdir(),"/images/image_",sprintf("%03d",which(i==seq(1,length(theta),30))),".jpg"))
  with(df[1:i,],plot(x,y,xlim=range(df$x),ylim=range(df$y),col="red"))
  dev.off()
  }

images.to.video(image.dir=paste0(tempdir(),"/images"),
vid.name="spiral.mp4",out.dir=tempdir(),
frame.rate=5,qual=100,silent=TRUE,overwrite=TRUE)

file.exists(paste0(tempdir(),"/spiral.mp4"))

#clean up
unlink(paste0(tempdir(),"/spiral.mp4"))
unlink(paste0(tempdir(),"/images"),recursive=TRUE)
```

---

images.to.video2 *Stitches images from video file passing filters to ffmpeg*

Description

Wrapper for ffmpeg video operations. Permits flexible filtering.

Usage

```
images.to.video2(
  image.dir = NULL,
  out.dir = NULL,
  vid.name = NULL,
  overwrite = TRUE,
  qual = 50,
  vid.ext = ".mp4",
  frame.rate = 10,
  raw = TRUE,
  filt = NULL,
  silent = TRUE
)
```

Arguments

image.dir	character; directory containing images to stitch.
out.dir	character; directory in which to store video.
vid.name	character; file name to be given video (should not include file extension).
overwrite	logical; should path described by vid.name be overwritten if it exhists.
qual	numeric; the quality of the video rendered from 1-100%. Defaults to 50%.
vid.ext	character; video type to output. mp4 currently works best.
frame.rate	numeric; video frame rate in fps.
raw	logical; encodes a raw AVI video with the "rawvideo" codec.
filt	character; video filter that should be applied to ffmpeg operation. See https://ffmpeg.org/ffmpeg-filters.html.
silent	logical; should output of system call for ffmpeg operation be suppressed.

Details

Assumes images are appended with a numeric sequence beginning with "_".

Value

Outputs a video of name "video.name+vid.ext".

See Also

vid.to.images2

Examples

```
#make some spiralled images and video


dir.create(paste0(tempdir(),"/images")) #make a directory to store images
```

```
a <- 2
b <- 3
theta <- seq(0,10*pi,0.01)
r <- a + b*theta
df <- data.frame(x=r*cos(theta), y=r*sin(theta)) # Cartesian coords
every.i <- 30
for(i in seq(1,length(theta),30)) {
 jpeg(paste0(tempdir(),"/images/image_",sprintf("%03d",which(i==seq(1,length(theta),30))),".jpg"))
  with(df[1:i,],plot(x,y,xlim=range(df$x),ylim=range(df$y),col="red"))
  dev.off()
  }

images.to.video2(image.dir=paste0(tempdir(),"/images"),
vid.name="spiral",out.dir=tempdir(),
frame.rate=5,qual=100,silent=TRUE,overwrite=TRUE)

file.exists(paste0(tempdir(),"/spiral.mp4"))

#clean up
unlink(paste0(tempdir(),"/images"),recursive=TRUE)
```

`kin.LDA` *Midline tracking over image sequences with ROI search using LDA*

Description

Experimental and untested (in the unit-testing sense). Automatically retrieves the midline of a detected ROI in each image of a sequence through thresholding and segmentation. Chose a fish-like ROI class detected through linear discriminate analysis (LDA) of PCA on elliptical Fourier described shapes. Initial training of ROIs is user defined or with the 'fishshapes' data set loaded with `trackter` (see details). For each detected ROI, `kin.LDA` finds the y-value midpoint along the x-value array of the ROI and fits a midline according to a chosen smoothing method (loess or spline). Also outputs the midline amplitude relative to a reference line determined by an anterior section of the ROI. Supported image formats are jpeg, png, and tiff.

Usage

```
kin.LDA(
  image.dir = NULL,
  frames = NULL,
  thr = 0.7,
  ant.per = 0.2,
  tips = 0.2,
  edges = FALSE,
  train.dat = NULL,
  rescale = FALSE,
  harms = 15,
```

```
  enorm = TRUE,
  retrain = 5,
  after.train = "LDA",
  ties = "fish",
  size.min = 0.05,
  show.prog = FALSE,
  smoothing = "loess",
  smooth = 0.3,
  smooth.points = 200,
  save = TRUE,
  out.dir = NULL,
  image.type = "orig",
  plot.pml = TRUE,
  flip = TRUE
)
```

Arguments

image.dir	character, directory containing images to analyze.
frames	numeric, vector indicating which images to process.
thr	numeric or character ('otsu') threshold to determine binary image. See Details.
ant.per	numeric; left-most percentage of ROI that establishes the horizontal reference for the midline displacement.
tips,	numeric, the proportion the the midline data to use in calculation of the head and tail position.
edges	logical, should ROIs on image edges by evaluated. See Details.
train.dat	Classified Out and Coo outlines that are produced from Momocs. See Details.
rescale	logical, should all shapes in PCA be rescaled. Performs best as 'FALSE'.
harms	numeric, the number of harmonics to use. If missing, Momocs sets 'nh.b' to 12. Will produce messages.
enorm	logical, should the EFA coefficients from efourier operations be normalized or not. See details and efourier
retrain	numeric, the number of frames on which to retrain the LDA data set. See details.
after.train	character, if set to 'size', LDA will be skipped after retrain and the ROI with a size closest to the ROI found by the LDA $>=$ will be chosen. This speeds calculations considerably. If 'LDA', the default, LDA will continue using the retraining classifications from frames $<=$ 'train'.
ties	character, how to chose ROI's in any one frame that appear fish-like. See details.
size.min	numeric, indicating the minimum size of ROIs as a proportion of the pixel field to be considered in analysis. May be useful if smaller unimportant ROIs appear in the frame. Default is 0.05.
show.prog	logical value indicating if outputted image should be displayed during analysis.
smoothing	character, the midline smoothing method, either 'loess' or 'spline'.

smooth	numeric; if smoothing is set to 'loess', 'span' parameter value for loess. If smoothing is set to 'spline' 'spar' parameter value for smooth.spline
smooth.points	numeric, number of equally spaced points along the ROI midline on which the smoothed midline is computed.
save	logical, value indicating if images should be outputted with midline and predicted midline based on the ant.per lm() overlaying original or binary images.
out.dir	character, the directory to which ouputted images should be saved. If NULL, then a subdirectory 'processed_images' in the working directory.
image.type	character; the type of image to be outputted, either 'orig' or 'bin' representing the original or binary images, respectively. Ignored if 'save==FALSE'.
plot.pml	logical, value indicating if outputted images should include the predicted midline (in blue) and the points according to ant.per used to construct the predicted midline (in green).
flip	logical, indicating if binary should be flipped.

Details

The algorithm assumes a left-right orientation, i.e., the head of the ROI is positioned left, the tail right. ffmpeg operations or even imageJ can rotate images not in this orientation. The ant.per value therefor establishes the reference line (theoretical straight midline) based on that portion of the head. The midline is calculated as the midpoints between the y extrema for each x position. If 'save=TRUE', images are saved as binary or the original with a body midline overlay and, if chosen, with the theoretical midline (based on ant.per).

Thresholding operations can be performed with an arbitrary (user-defined) numeric value or with Otsu's method ('thr="otsu"'). The latter chooses a threshold value by minimizing the combined intra-class variance. See otsu.

Before train, ROIs are chosen according to LDA of a PCA object constructed from efourier analysis. LDA is trained by a user define 'train.dat' when the frame $<=$ retrain. LDA will proceed after retrain if after.train='LDA', but the LDA will be trained by the contours classified as 'fish' and 'not.fish' found during the chosen training period.

enorm Normalization of EFA coefficients is often perilous, especially for symmetrical shapes, a conditional met for undulating, bilaterally symmetrical organisms at least some of the time and perhaps for many of the frames included in any analysis. Thus, 'enorm' by default is set to 'FALSE'. 'enorm=TRUE' may produce odd ROI choices and should be used cautiously.

train.dat This should be a Coo and Out object produced by efourier analysis of predefined shapes. A user defined dataset or the fishshapes dataset in trackter must be used for training. fishshapes includes several arbitrary shapes (circles, squares, U-shapes, etc.) as well as several fish shapes: sunfish (genus Lepomis), eel (genus Anguilla), and trout (genus Onchorhynchus) swimming over one tail-beat cycle. A user-defined dataset must have shapes classified with factors identical to the fishshapes contours, that is by shape, type, and edge. Shape levels should indicate what type of shape is described by the contour (e.g., 'circle', 'L-shape', 'trout', 'eel', etc). The type levels must describe the shape as 'fish' or 'not.fish'. The edge levels must be 'FALSE'.

edges Set by default to 'FALSE'. It is not advisable to include shapes that are on the edge of any frame and are therefore incomplete. retrain After this value, the LDA analysis will use the ROIs determined as 'fish' and 'not.fish' in the frames $>=$ retrain to discriminate fish from non-fish

shapes. This speeds up analysis considerably. `ties` Determines how to chose ROIs if more than one fish-like ROI is found in any frame. 'fish' will result in choosing the ROI with shape types in which the best *and* second-best fish-like shape (according to posterior probabilities) match a fish-like shape in the training and/or retraining datasets.'post' will chose the best fish-like shape according the the highest posterior probability from LDA.

Value

A list with the following components:

`kin.dat` a data table consisting of frame-by-frame position parameters for the ROI determined by LDA analysis.

- the frame number
- 'x' and "y': the position of the tail (rightmost or posteriormost)
- 'head.x' and 'head.y': the x and y position of the head (leftmost or anteriormost)
- 'amp': the amplitude (`amp`) of the tail relative to thr theoretical midline determined by the `lm()` predictions from `ant.per`
- 'roi': a character indicating the ROI ranked by size ('a' being the largest)
- 'head.pval': p values of the `lm()` fit that describes the position of the head as determined by `ant.per` (green points in the outputted images/video)

`midline` A data table containing, for each frame described by `frames`, the following:

- 'x' and 'y.m': x and y positions of the midline of the ROI #'
- 'y.min' and 'y.max': min and max y positions ROI's contour used in y.m calculation
- 'mid.pred': the predicted linear midline based on the points/pixels defined by `head.per` (green points in the outputted images/video)
- 'y.pred': midline points fit to a smooth spline or loess model with spar or span equal to `smooth` (red curve in the outputted images/video)
- 'wave.y': midline points 'y.pred' relative to 'mid.pred'
- 'roi': a character indicating ROI size ('a' being the largest)

`cont` A data table containing x and y positions of the contours used to calculate the data in 'kin.dat'. Contains the following:

- 'frame': the frame #'
- 'x' and 'y': the x and y positions of the contours

`all.classes` A data table containing the following for all ROIs detected:

- 'frame': the frame
- 'roi': the name of each ROI found in a frame.
- 'size': the size of each ROI

`dim` the x and y dimensions of the images analyzed

See Also

kin.simple, kin.search, efourier LDA, fishshapes.

Examples

```
# produce a classic midline waveform plot of swimming
# fish searching a image field with a two fish-like ROIs
## Not run:
require(wesanderson)
require(ggplot2)
require(data.table)
require(dplyr)
require(EBImage)

#download example images and place in 'example' subdirectory
f <- "https://github.com/ckenaley/exampledata/blob/master/example.zip?raw=true"
download.file(f, "temp.zip")
unzip("temp.zip")
unlink("temp.zip")

#load fishshapes data
data(fishshapes)


kin <- kin.LDA(image.dir = "example",frames=1:20,thr=0.7,
              ant.per=.25,enorm=FALSE,show.prog = FALSE,retrain=2,
              train.dat = fishshapes,after.train="LDA",edges=FALSE,
              )
ml <- kin$midline
 #x start at 0
ml <-ml[,x2:=x-x[1],by=frame]

#compute instantaneous amplitude of tail (last/rightmost point) and wave crest x position  by frame
ml2 <-ml[,.(amp.i=abs(last(wave.y))),by=frame]

ml <- merge(ml,ml2,by="frame") #merge these

pal <- wes_palette("Zissou1", 100, type = "continuous") #"Zissou" color palette
p <- ggplot(dat=ml,aes(x=x2,y=wave.y))+theme_classic(15)+scale_color_gradientn(colours = pal)
p <- p+geom_line(aes(group=frame,color=amp.i),
stat="smooth",method = "loess", size = 1.5,alpha = 0.5)
print(p)


### Make a video of processed frames

images.to.video2(image.dir="processed_images",
vid.name="trout_test",frame.rate=5,qual=100,raw=FALSE)
file.exists("trout_test_red.mp4")


## End(Not run)
```

`kin.search` *Midline tracking over image sequences*

Description

Automatically retrieves the midline of a detected ROI in each image of a sequence through thresholding and segmentation; finds the y-value midpoint along the x-value array of the ROI and fits a midline according to a chosen smoothing method (loess or spline). Also outputs the midline amplitude relative to a reference line determined by an anterior section of the ROI. Supported image formats are jpeg, png, and tiff.

Usage

```
kin.search(
  image.dir = NULL,
  frames = NULL,
  thr = "otsu",
  plot.pml = TRUE,
  show.prog = FALSE,
  ant.per = 0.1,
  tips = 0.02,
  smoothing = "loess",
  smooth = 0.25,
  smooth.points = 200,
  image.type = "orig",
  save = TRUE,
  out.dir = NULL,
  flip = TRUE,
  size.min = 0.02,
  search.for = "largest",
  edges = FALSE,
  border = 5
)
```

Arguments

`image.dir`	character, directory containing images to analyze.
`frames`	numeric, vector indicating which images to process.
`thr`	numeric or character ('otsu') threshold to determine binary image. See Details.
`plot.pml`	logical, value indicating if outputted images should include an overlay of the theoretical midline based on `ant.per`.
`show.prog`	logical value indicating if outputted image should be displayed during analysis.
`ant.per`	numeric; left-most percentage of ROI that establishes the horizontal reference for the midline displacement.

tips,	numeric, the proportion the the midline data to use in calculation of the head and tail position.
smoothing	character, the midline smoothing method, either 'loess' or "spline".
smooth	numeric; if smoothing is set to 'loess', smoothing parameter value for plotted midline.
smooth.points	numeric, number of equally spaced points along the ROI midline on which the smoothed midline is computed.
image.type	character; the type of image to be outputted, either 'orig' or 'bin' representing the original or binary images, respectively. Ignored if 'save=FALSE'.
save	logical, value indicating if images should be outputted with midline and predicted midline based on the ant.per lm() overlaying original or binary images.
out.dir	character, the directory to which outputted images should be saved.
flip	logical, indicating if binary should be flipped.
size.min	numeric, indicating the minimum size of ROIs as a proportion of the pixel field to be considered in analysis. May be useful if smaller unimportant ROIs appear in the frame. Default is 0.02.
search.for	character, the search parameter. See Details.
edges	logical, should ROIs on image edges be evaluated. See Details.
border	if edges=TRUE, size of border to add in pixels. Dee details.

Details

The algorithm assumes a left-right orientation, i.e., the head of the ROI is positioned left, the tail right. The ant.per value therefor establishes the reference line (theoretical straight midline) based on that portion of the head. The midline is calculated as the midpoints between the y extrema for each x position. Chooses ROIs based on relative ROI size or position.

Thresholding operations can be performed with an arbitrary (user defined) numeric value or with Otsu's method ('thr="otsu"'). The latter chooses a threshold value by minimizing the combined intra-class variance. See otsu.

If 'edges=TRUE', it is best to add an artificial border so that any part of the ROI in contact with the edge can be distinguished from it.

search.for determines how ROIs are chosen:

- "offset", the ROI with a centroid that is the shortest linear distance to the center of the field
- "offset.x", the ROI with a centroid x position that is closest to the x position of the center of the field
- "offset.y", the ROI with a centroid y position that is closest to the y position of the center of the field
- "largest", the largest ROI.

These choices will be made on ROI sets that are not on the edge of the field if 'edges=FALSE'.

edges Set by default to 'FALSE'. It is not advisable to include shapes that are on the edge of any frame and are therefore incomplete. Yet, if set to 'TRUE', the border adds a black border to the image so that the intended ROI may be distinguished from the edge.

image.type Can be set as "orig" or "bin". "orig" plots midline and reference lines over the original video frames, "bin" over binary images.

Value

A list with the following components:

`kin.dat` a data table consisting of frame-by-frame position parameters for the ROI determined by `search.for`.

- the frame number
- 'x' and "y': the position of the tail (rightmost or posteriormost)
- 'head.x' and 'head.y': the x and y position of the head (leftmost or anteriormost)
- 'amp': the amplitude (`amp`) of the tail relative to thr theoretical midline determined by the `lm()` predictions from `ant.per`
- 'head.pval': p values of the `lm()` fit that describes the position of the head as determined by `ant.per` (green points in the outputted images/video)
- 'roi': a character indicating the ROI ranked by size ('a' being the largest)
- 'edge': indicating whether ROI was on the edge of the image field
- 'size': size of the ROI in pixels^2
- 'offset.x': ROI distance from horizontal center
- 'offset.y': ROI distance from vertical center
- 'offset': linear distance of ROI's centroid to image center

`midline` A data table containing, for each frame described by `frames`, the following:

- 'x' and 'y.m': x and y positions of the midline of the ROI #'
- 'y.min' and 'y.max': min and max y positions ROI's contour used in y.m calculation
- 'mid.pred': the predicted linear midline based on the points/pixels defined by `head.per` (green points in the outputted images/video)
- 'y.pred': midline points fit to a smooth spline or loess model with spar or span equal to `smooth` (red curve in the outputted images/video)
- 'wave.y': midline points 'y.pred' relative to 'mid.pred'
- 'roi': a character indicating ROI size ('a' being the largest)

`cont` A data table containing x and y positions of the contours used to calculate the data in 'kin.dat'. Contains the following:

- 'frame': the frame
- 'x' and 'y': the x and y positions of the contours

`all.classes` A data table containing the following for all ROIs detected:

- 'frame': the frame
- 'roi': the name of each ROI found in a frame.
- 'edge': indicating whether ROI was on the edge of the image field
- 'size': size of the ROI in pixels^2
- 'offset.x': ROI distance from horizontal center
- 'offset.y': ROI distance from vertical center

- 'offset': linear distance of ROI's centroid to image center

`dim` the x and y dimensions of the images analyzed

A list with the following components:

`kin.dat` a data frame consisting of frame-by-frame position parameters for the ROI indicated by `n.blob`:

- the frame number
- 'head.x' and 'head.y': the x and y position of the head (leftmost or anteriormost)
- 'x' and 'y': the position of the tail (rightmost or posteriormost)
- 'amp': the amplitude (`amp`) of the tail
- 'cent.x' and 'cent.y': centroid coordinate of ROI
- 'roi': a character indicating ROI size ('a' being the largest)
- 'head.pval': p values of the `lm()` fit that describes the position of the head as determined by `ant.per` (green points in the outputted images/video)

`midline` A data frame containing, for each frame described by `frames`, the following:

- 'x' and 'y.m': x and y positions of the midline of the ROI
- 'mid.pred': the predicted linear midline based on the points/pixels defined by `head.per` (green points in the outputted images/video)
- 'y.pred': midline points fit to a smooth spline or loess model with spar or span equal to `smooth` (red curve in the outputted images/video)
- 'wave.y': midline points 'y.pred' normalized to 'mid.pred'
- 'roi': a character indicating ROI size ('a' being the largest)
- 'cent.x': x centroid of ROI
- 'cent.y': y centroid of ROI
- 'offset.x': ROI distance from horizontal center
- 'offset.y': ROI distance from vertical center
- 'offset.total': sum of ROI offset.x and offset.y
- 'ar': aspect ration of the ROI
- 'size': size of ROI in pixels

`dim` the x and y dimensions of the images analyzed

See Also

`kin.simple`

Examples

```
#### plot lot caudal amplitude and produce a classic midline waveform plot of swimming fish

##A very long example.
## Not run:

#download example images and place in 'example' subdirectory
f <- "https://github.com/ckenaley/exampledata/blob/master/example.zip?raw=true"

download.file(f, paste0(tempdir(),"/temp.zip"))
unzip(paste0(tempdir(),"/temp.zip"), exdir=tempdir())
unlink(paste0(tempdir(),"/temp.zip"))

dir.create(paste0(tempdir(),"/processed_images"))
kin <- kin.search(image.dir =paste0(tempdir(),"/example"),
       search.for = "largest",
      smoothing = "loess",frames=1:50,
      out.dir=paste0(tempdir(),"/processed_images"),
      show.prog = FALSE,thr = "otsu",
      image.type="bin",smooth=0.4)

#plot instantaneous amplitude of tail (last/rightmost point) over frames
p <- ggplot(dat=kin$kin.dat,aes(x=frame,y=amp))+geom_line()+geom_point()+theme_classic(15)
print(p)

# midline plot
ml <- kin$midline
#leftmost x starts at 0
ml <- ml[,x2:=x-x[1],by=frame]

ml <- merge(ml,kin$kin.dat[,list(frame,amp)],by="frame") #merge these

pal <- wes_palette("Zissou1", 100, type = "continuous") #"Zissou" color palette

p <- ggplot(dat=ml,aes(x=x2,y=wave.y))+theme_classic(15)+scale_color_gradientn(colours = pal)

p <- p+geom_line(aes(group=frame,color=amp),stat="smooth",method = "loess", size = 1.5)
print(p)

#Make a video of processed frames

images.to.video2(image.dir=paste0(tempdir(),"/processed_images"),
vid.name="trout_test",out.dir=tempdir(),frame.rate=5,qual=100,raw=FALSE)
file.exists(paste0(tempdir(),"/trout_test_red.mp4"))


## End(Not run)

## A very short example.

#retrieve image
```

```
i <- EBImage::readImage(system.file("extdata/img", "sunfish_BCF.jpg", package = "trackter"))
#create directory and write image to it
t <- tempdir()


dir.create(paste0(t,"/images"))
EBImage::writeImage(i,paste0(t,"/images/sunfish001.jpg"),type = "jpeg")

list.files(paste0(t,"/images"))
#run kin.search and save output image to directory
kin.i<- kin.search(image.dir = paste0(t,"/images"),smooth=0.7,save = TRUE,out.dir = t)

#plot midline over original image
with(kin.i$midline,plot(x,wave.y))

i2 <- EBImage::readImage(paste0(t,"/sunfish001_000.jpg"))
EBImage::display(i2,method="raster")

#clean up
unlink(paste0(t,"/images"),recursive=TRUE)
```

`kin.simple` *Simplified midline tracking over image sequences*

Description

Automatically retrieves the midline of a detected ROI based on size. Assumes the ROI of interest is the largest detected and not intersecting the edges of the image frame, conditions often met in kinematic studies. For each ROI of interest, finds the y-value midpoint along the x-value array of the ROI and fits a midline according to a chosen smoothing method (loess or spline). Also outputs the midline amplitude relative to a reference line determined by an anterior section of the ROI and outputs contours ROIs in each frame for subsequent analysis. Supported image formats are jpeg, png, and tiff.

Usage

```
kin.simple(
  image.dir = NULL,
  frames = NULL,
  thr = 0.7,
  size.min = 0.05,
  ant.per = 0.2,
  tips = 0.02,
  smoothing = "loess",
  smooth = 0.25,
  smooth.points = 200,
  save = TRUE,
  out.dir = NULL,
```

```
  plot.pml = TRUE,
  image.type = "orig",
  flip = TRUE,
  show.prog = FALSE
)
```

Arguments

image.dir	character, directory containing images to analyze.
frames	numeric, vector indicating which images to process.
thr	numeric or character ('otsu') threshold to determine binary image. See Details.
size.min	numeric, indicating the minimum size of ROIs as a proportion of the pixel field to be considered in analysis. May be useful if smaller unimportant ROIs appear in the frame. Default is 0.05.
ant.per	numeric; left-most proportion of ROI that establishes the horizontal reference for the midline displacement.
tips,	numeric, the proportion the the midline data to use in calculation of the head and tail position.
smoothing	character, the midline smoothing method, either 'loess' or 'spline'.
smooth	numeric; if smoothing is set to 'loess', passed to 'span' parameter of loess. If smoothing is set to 'spline', passed to 'spar' parameter of smooth.spline
smooth.points	numeric, number of equally spaced points along the ROI midline on which the smoothed midline is computed.
save	logical, value indicating if images should be outputted with midline and predicted midline based on the lm() predictions from ant.peroverlaying original or binary images.
out.dir	character, the directory to which outputted images should be saved.
plot.pml	logical, value indicating if outputted images should include the predicted midline (in blue) and the points according to ant.per used to construct the predicted midline (in green).
image.type	character; the type of image to be outputted, either 'orig' or 'bin' representing the original or binary images, respectively. Ignored if 'save=FALSE'.
flip	logical, indicating if binary image should be flipped.
show.prog	logical, indicating if outputted image should be displayed during analysis.

Details

The algorithm assumes a left-right orientation, i.e., the head of the ROI is positioned left, the tail right. ffmpeg operations or even imageJ can rotate images not in this orientation. The ant.per value therefore establishes the reference line (theoretical straight midline) based on that portion of the head. The midline is calculated as the midpoints between the y extrema for each x position.

If 'save=TRUE', images are saved as binary or the original with a body midline overlay and, if chosen, with the theoretical midline (based on ant.per).

Thresholding operations can be performed with an arbitrary (user defined) numeric value or with Otsu's method ('thr="otsu"'). The latter chooses a threshold value by minimizing the combined intra-class variance. See otsu.

Value

A list with the following components:

`kin.dat` a data table consisting of frame-by-frame position parameters for the ROI determined by LDA analysis.

- the frame number
- 'x' and 'y': the position of the tail (rightmost or posteriormost)
- 'head.x' and 'head.y': the x and y position of the head (leftmost or anteriormost)
- 'amp': the amplitude (`amp`) of the tail relative to the theoretical midline determined by the `lm()` predictions from `ant.per`
- 'roi': a character indicating the ROI ranked by size ('a' being the largest)
- 'head.pval': p values of the `lm()` fit that describes the position of the head as determined by `ant.per` (green points in the outputted images/video)

`midline` A data table containing, for each frame described by `frames`, the following:

- 'x' and 'y.m': x and y positions of the midline of the ROI #'
- 'y.min' and 'y.max': min and max y positions ROI's contour used in y.m calculation
- 'mid.pred': the predicted linear midline based on the points/pixels defined by `ant.per` (green points in the outputted images/video if 'plot.pml=TRUE')
- 'y.pred': midline points fit to a smooth spline or loess model with spar or span equal to `smooth` (red curve in the outputted images/video)
- 'wave.y': midline points 'y.pred' relative to 'mid.pred'
- 'roi': a character indicating ROI size ('a' being the largest)

`cont` A data table containing x and y positions of the contours used to calculate the data in 'kin.dat'. Contains the following:

- 'frame': the frame
- 'x' and 'y': the x and y positions of the contours

`all.classes` A data table containing the following for all ROIs detected:

- 'frame': the frame
- 'roi': the name of each ROI found in a frame.
- 'size': the size of each ROI

`dim` the x and y dimensions of the images analyzed

See Also

`kin.search`

Examples

```
#### plot caudal amplitude and produce a classic midline waveform plot of swimming fish
##A very long example.
## Not run:

#download example images and place in 'example' subdirectory
f <- "https://github.com/ckenaley/exampledata/blob/master/example.zip?raw=true"

download.file(f, paste0(tempdir(),"/temp.zip"))
unzip(paste0(tempdir(),"/temp.zip"), exdir=tempdir())
unlink(paste0(tempdir(),"/temp.zip"))

dir.create(paste0(tempdir(),"/processed_images"))
kin <- kin.simple(image.dir =paste0(tempdir(),"/example"),
      smoothing = "loess",frames=1:50,
      out.dir=paste0(tempdir(),"/processed_images"),
      show.prog = FALSE,thr = "otsu",
      image.type="bin",smooth=0.4)

#plot instantaneous amplitude of tail (last/rightmost point) over frames
p <- ggplot(dat=kin$kin.dat,aes(x=frame,y=amp))+geom_line()+geom_point()+theme_classic(15)

print(p)

# midline plot
ml <- kin$midline
#leftmost x starts at 0
ml <- ml[,x2:=x-x[1],by=frame]

ml <- merge(ml,kin$kin.dat[,list(frame,amp)],by="frame") #merge these

pal <- wes_palette("Zissou1", 100, type = "continuous") #"Zissou" color palette

p <- ggplot(dat=ml,aes(x=x2,y=wave.y))+theme_classic(15)+scale_color_gradientn(colours = pal)

p <- p+geom_line(aes(group=frame,color=amp),stat="smooth",method = "loess", size = 1.5)
print(p)

#Make a video of processed frames

images.to.video2(image.dir=paste0(tempdir(),"/processed_images"),
vid.name="trout_test",out.dir=tempdir(),frame.rate=5,qual=100,raw=FALSE)
file.exists(paste0(tempdir(),"/trout_test_red.mp4"))


## End(Not run)

## A very short example.

#retrieve image
i <- EBImage::readImage(system.file("extdata/img", "sunfish_BCF.jpg", package = "trackter"))
#create directory and write image to it
```

```
t <-tempdir()
dir.create(paste0(t,"/images"))
EBImage::writeImage(i,paste0(t,"/images/sunfish001.jpg"),type = "jpeg")

#run kin.search and save output image to directory
kin.i<- kin.simple(image.dir = paste0(t,"/images"),save = TRUE,out.dir = t)

#plot midline
with(kin.i$midline,plot(x,wave.y))
i2 <- EBImage::readImage(paste0(t,"/sunfish001_000.jpg"))
EBImage::display(i2,method="raster")
#clean up
unlink(paste0(t,"/images"),recursive=TRUE)
```

---

rad *convert degrees to radians*

Description

convert degrees to radians

Usage

```
rad(x)
```

Arguments

x	Numeric; value in degrees

Value

A single value

See Also

deg

---

vid.to.images *Extracts images from a video file with ffmpeg*

Description

Uses ffmpeg systems calls to extract images from a video.

Usage

```
vid.to.images(vid.path = NULL, out.dir = NULL, overwrite = FALSE, qual = 50)
```

Arguments

vid.path	Character; path of video file to be processed.
out.dir	character; directory path in which to store images.
overwrite	logical; should path described by 'out.dir' be overwritten if it exhists.
qual	numeric; the quality of the jpeg images to be rendered from 1-100%. Defaults to 50%.

Value

Extracts all the images of the video and saves them to an "images" directory with appended number sequence

See Also

images.to.video

Examples

```
#make a video with animation package

require(animation)
fun <- function(){
y <- sin(1:50)
x <- 1:50
for(i in 1:50) {
  plot(x[i],y[i],col="red",xlim=c(0,50),ylim=range(y))
  animation::ani.pause()
  }
}
animation::saveVideo(fun(),video.name=paste0(tempdir(),"/wave.mp4"),interval = 0.2)

#create directory in which to store images
dir.create(paste0(tempdir(),"/images"))
vid.to.images(vid.path=paste0(tempdir(),"/wave.mp4"),
out.dir= paste0(tempdir(),"/images"),qual=100)

#see the images in the "images" subdirectory
list.files( paste0(tempdir(),"/images"))

#clean up
unlink(paste0(tempdir(),"/images"),recursive=TRUE)
```

`vid.to.images2` *Extracts images from a video file with ffmpeg*

Description

Extract images from video file using ffmpegs flexible video filters and codecs

Usage

```
vid.to.images2(
  vid.path = NULL,
  out.dir = NULL,
  overwrite = FALSE,
  filt = NULL,
  codec = NULL,
  silent = TRUE
)
```

Arguments

`vid.path`	character; path of video file to be processed.
`out.dir`	character; directory path in which to store images.
`overwrite`	logical; should path described by 'out.dir' be overwritten if it exhists.
`filt`	character; video filter that should be applied to ffmpeg operation. See https://ffmpeg.org/ffmpeg-filters.html
`codec`	character; video codec to apply in ffmpeg operation
`silent`	logical; should output of `system` call for ffmpeg operation be suppressed.

Details

Particularly useful for resizing images

Value

Extracts all the images of the video and saves them to an "images" directory with appended number sequence

See Also

`images.to.video`

Examples

```
#make a video with animation package

fun <- function(){
y <- sin(1:50)
x <- 1:50
for(i in 1:50) {
  plot(x[i],y[i],col="red",xlim=c(0,50),ylim=range(y))
  animation::ani.pause()
  }
}
animation::saveVideo(fun(),video.name=paste0(tempdir(),"/wave.mp4"),interval = 0.2)

#reduce the image images to 200 px wide maintaining aspect ratio
#notice the spaces at the beginning/end of string
filt.red <- " -vf scale=200:-1 "
c <- " -c:v libx264 "
dir.create(paste0(tempdir(),"/images"))
vid.to.images2(vid.path=paste0(tempdir(),"/wave.mp4"),
out.dir=paste0(tempdir(),"/images"),filt=filt.red,codec=NULL)

#see the images in the "images" directory
list.files( paste0(tempdir(),"/images"))

#clean up
unlink(paste0(tempdir(),"/images"),recursive=TRUE)
```

---

wave *Compute wavelengths from a sine-like waveform*

Description

Computes full wavelengths and their positions and amplitude from a sine-like waveform based on either peak-to-peak, trough-to-trough, or internodal distance.

Usage

```
wave(
  x,
  y,
  method = "zeros",
  zero.begin = TRUE,
  fit = TRUE,
  dens = 10,
  smooth = 0.1,
  smoothing = "loess"
)
```

Arguments

x	numeric; x position
y	numeric; y position
method	character; how waves should be found and classified, where it crosses zero/the internodal length ("zeros"), peak to peak ("p2p") or trough to trough ("t2t"). See Details.
zero.begin	logical; does wave begin at zero? Default is 'TRUE' and will help find waves beginning at first x,y values if y=0
fit	logical; if 'method="zeros"', should zeros be detected by a fitting operation. See Details.
dens	numeric; factor by which to increase the sample density used in fitting when 'method="zeros"'. See Details.
smooth	numeric; if smoothing is set to 'loess', 'span' parameter value for loess. If smoothing is set to 'spline' 'spar' parameter value for smooth.spline
smoothing	character; the smoothing method when 'fit=TRUE', either 'loess' or 'spline'. See Details.

Details

If 'method="p2p"' or 'method="t2t"', full waves are found using critical points (i.e., local maxima, the peaks or minima, the troughs) with features.

If 'method="zeros"' and 'fit=TRUE', zero crossings are determined by first increasing the sample density by a factor determined by dens. A more dense loess or smooth.spline model is then fit to the data and new y values predicted. Wave positions and lengths are determined based on these predicted values. This option should be useful when the sampling density of the waveform is relatively low and therefor detected wave positions and zero crossings (the internodes) may be rather coarse.

Value

A list with the following components:

method the method chosen to find full waves

names a data table with columns 'x', 'y', and 'wave' describing the x and y positions of the wave and a numeric name of each wave detected, respectively. If 'method="zeros"' and 'fit=TRUE', these values reflect the predicted, more dense data as determined by smoothing, smooth, and dens.

dat a data table describing each wave detected.

- 'zeros': x value where y crosses zero. Returns NA if method is 'p2p' or 't2t', value is NA.
- 'wave.begin': x value where each wave begins.
- 'wave.end': x value where each wave ends.
- 'begin.index': x index of where each wave begins.
- 'end.index': x index of where each wave ends.
- 'wave': numeric name of each wave.

- 'l': the length of each wave.
- 'amp1': the peak amplitude of each wave. If method is set to 'p2p' or 't2t' this is the begin amplitude. If "method='zeros'" this is the peak amplitude between internodes.
- 'amp2': If method is set to 'p2p' or 't2t' this is the end amplitude. If "method='zeros'" this is the minimum amplitude between internodes.
- 'pos1': If method is set to 'p2p' or 't2t' the x position of begin amplitude for each half wave and identical to 'begin'. If "method='zeros'", the position of peak amplitude between the internodes.
- 'pos2': If method is set to 'p2p' or 't2t' the x position of end amplitude for each half wave and identical to 'end'. If "method='zeros'", the position of minimum amplitude between the internodes.

If 'method="zeros"' and 'fit=TRUE', these values reflect the predicted, more dense data as determined by smoothing,smooth, and dens.

See Also

features, loess, smooth.spline

Examples

```
require(ggplot2)
#Find length of the full waves
x <- seq(0,pi,0.01)
y <- sin(x^2*pi)

#zero method
w.z <- wave(x,y,method="zeros",smoothing="spline",smooth=0.1)

#plot wave with detected full waves using fitted 'zeros' method
p <- ggplot()+geom_point(aes(x=x,y=y))
p <- p+geom_line(data=w.z$names,aes(x=x,y=y,col=wave),alpha=0.4,size=3,inherit.aes=FALSE)
p+theme_classic()

#plot lambda as it varies with position

qplot(data=w.z$dat,x=pos1,y=l)

#trough-to-trough method
w.p <- wave(x,y,method="t2t")

qplot(data=w.p$names,x=x,y=y,col=wave)
```

Index