Package 'dynaSpec'

September 29, 2024

Type Package

Title Dynamic Spectrogram Visualizations

Version 1.0.2

Description A set of tools to generate dynamic spectrogram visualizations in video format.

License GPL (>= 2)

Imports utils, grDevices, graphics, seewave, tuneR, grid, png, ggplot2, viridis, scales, ari, gganimate, warbleR

Depends R (>= 3.2.1)

LazyData TRUE

SystemRequirements ffmpeg

URL https://github.com/maRce10/dynaSpec

BugReports https://github.com/maRce10/dynaSpec/issues

NeedsCompilation no

Suggests parallel, imager, fs

RoxygenNote 7.3.2

Repository CRAN

Language en-US

Encoding UTF-8

Author Marcelo Araya-Salas [aut, cre] (<https://orcid.org/0000-0003-3594-619X>)

Maintainer Marcelo Araya-Salas <marcelo.araya@ucr.ac.cr>

Date/Publication 2024-09-29 04:30:02 UTC

Contents

canyon_wren *Acoustic recording of a* Catherpes mexicanus *(canyon wren) song.*

Description

Acoustic recording of a *Catherpes mexicanus* (canyon wren) song.

Usage

```
data(canyon_wren)
```

Format

One Wave object:

canyon_wren *Catherpes mexicanus* recording

paged_spectro *Make a paged dynamic spectrogram similar to spectral display in Adobe Audition*

Description

This function works on an object generated with prep_static_ggspectro, an alias for prepStaticSpec(). Video generation is very time consuming, and all the desired spectrogram parameters should be set in the prep step. The output is an mp4 video of a dynamic spectrogram video. If the input sound file was segmented in the prep step, the resulting video will be a concatenation of multiple dynamic spectrogram "pages." Each page has a sliding window revealing the part of the static spectrogram being played. Temporal width of each page is defined by the xLim parameter in prep_static_ggspectro. You can also output temporary segmented files, if desired.

Usage

```
paged_spectro(
  specParams,
  destFolder,
  vidName,
  framerate = 30,
  highlightCol = "#4B0C6BFF",
  highlightAlpha = 0.6,
  cursorCol = "white",
  delete_temp_files = TRUE
)
```

Arguments

specParams	an object returned from prep_static_ggspectro
destFolder	destination of output video; this setting overwrites setting from specParams object
vidName	expects "FileName", .mp4 not necessary; if not supplied, will be named after the file you used in prep_static_ggspectro()
framerate	by default, set to 30 (currently this is not supported, as animate doesn't honor the setting)
highlightCol	default "#4B0C6BFF" (a purple color to match the default viridis 'inferno' palette)
highlightAlpha	opacity of the highlight box; default is 0.6
cursorCol	Color of the leading edge of the highlight box; default "white"
delete_temp_files	Default= TRUE, deletes temporary files (specs & WAV files used to create concatenated video)

Value

Nothing is returned, though progress and file save locations are output to user. Video should play after rendering.

Author(s)

Matthew R Wilkins (<matt@galacticpolymath.com>)

References

Araya-Salas M & Wilkins M R. (2020). *dynaSpec: dynamic spectrogram visualizations in R*. R package version 1.0.0.

See Also

prep_static_ggspectro

Examples

```
## Not run:
#show wav files included with dynaSpec
f <- list.files(pattern=".wav", full.names = TRUE,
     path = system.file(package="dynaSpec"))

femaleBarnSwallow<-prep_static_ggspectro(f[1],destFolder=tempdir(),
                   onlyPlotSpec = FALSE, bgFlood= TRUE)
paged_spectro(femaleBarnSwallow,destFolder=tempdir())

maleBarnSwallow<-prep_static_ggspectro(f[2],destFolder=tempdir(),
                 onlyPlotSpec = FALSE, bgFlood= TRUE,min_dB=-40)
```

```
paged_spectro(femaleBarnSwallow,destFolder=tempdir())

# Make a multipage dynamic spec of a humpback whale song
# Note, we're saving PNGs of our specs in the working directory; to add
# axis labels, we set onlyPlotSpec to F, and to make the same background
# color for the entire figure, we set bgFlood= TRUE;
# The yLim is set to only go to 0.7kHz, where the sounds are for these big whales;
#also applying an amplitude transform to boost signal.
#This is a longer file, so we're taking the first 12 seconds with crop=12
#xLim=3 means each "page" will be 3 seconds, so we'll have 4 dynamic spec pages that get combined

humpback <- prep_static_ggspectro(
"http://www.oceanmammalinst.org/songs/hmpback3.wav",destFolder=tempdir(),savePNG= FALSE,
onlyPlotSpec=FALSE,bgFlood= TRUE,yLim=c(0,.7),crop=12,xLim=3,ampTrans=3)

#to generate multipage dynamic spec (movie), run the following
paged_spectro(humpback,destFolder=tempdir())

# see more examples at https://marce10.github.io/dynaSpec/

## End(Not run)
```

`prep_static_ggspectro` *Generate ggplot2-based spectrogram(s), which can be passed to paged_spectro*

Description

Can be used to generate single or segmented static spectrograms. Works as standalone, but the returned object is also intended to feed into `paged_spectro`. Workflow: 1) use prep_static_ggspectro to crop, filter, segment and tweak all spectrogram parameters; 2) pass these settings to `paged_spectro` to generate dynamic spectrogram video.

Usage

```
prep_static_ggspectro(
  soundFile,
  destFolder,
  outFilename = NULL,
  savePNG = FALSE,
  colPal = "inferno",
  crop = NULL,
  bg = NULL,
  filter = NULL,
  xLim = NULL,
  yLim = c(0, 10),
  title = NULL,
  plotLegend = FALSE,
  onlyPlotSpec = TRUE,
```

```
  ampTrans = 1,
  resampleRate = 15000,
  min_dB = -30,
  wl = 512,
  ovlp = 90,
  wn = "blackman",
  specWidth = 9,
  specHeight = 3,
  colbins = 30,
  ampThresh = 0,
  bgFlood = FALSE,
  fontAndAxisCol = NULL,
  optim = NULL,
  ...
)
```

Arguments

soundFile	should work with URLs, full and relative paths; handles .mp3 and .wav
destFolder	path to directory to save output. Needs to be like "figures/spectrograms/" to be relative to working directory. Default=parent folder of soundFile. Specify "wd" to output to the working directory, gotten from [get_wd()]
outFilename	name for output PNG. default=NULL will use input name in output filename.
savePNG	logical; Save static spectrograms as PNGs? They will be exported to destFolder.
colPal	color palette; one of "viridis","magma","plasma","inferno","cividis" from the viridis package OR a 2 value vector (e.g. c("white","black")), defining the start and end of a custom color gradient
crop	subset of recording to include; default crop=NULL will use whole file, up to 10 sec; if a number, interpreted as crop first X.X sec; if c(X1,X2), interpreted as trimming out a specific time interval in sec; if crop=FALSE, will not crop at all, even for recordings over 10 sec.
bg	background color (defaults to 1st value of chosen palette)
filter	apply a bandpass filter? Defaults to none (NULL). Expects 'c(0,2)' where sound from 0 to 2kHz would be filtered out
xLim	the time limit (x-axis width) in seconds for all spectrograms; i.e. page width in seconds for multi-page dynamic spectrograms (defaults to WAV file length, unless file duration >5s). To override the 5s limit, put xLim=Inf or specify the desired spectrogram x-axis limit.
yLim	the frequency limits (y-axis); default is c(0,10) aka 0-10kHz
title	string for title of plots; default=NULL
plotLegend	logical; include a legend showing amplitude colors? default=FALSE
onlyPlotSpec	logical; do you want to just plot the spec and leave out the legend, axes, and axis labels? default= TRUE

ampTrans	amplitude transform for boosting spectrum contrast; default=1 (actual dB values); specify a decimal number for the lambda value of scales::modulus_trans(); 2.5 is a good place to start. (This amplifies your loud values the most, while not increasing background noise much at all)
resampleRate	a number in Hz to downsample audio for spectrogram only. This will simplify audio data and speed up generation of spectrogram. Passed to [tuneR::downsample()]. Default=15000 shaves off a few seconds without losing much quality. Put NULL to keep original sample rate for spectrogram. Audiofile will not be resampled for MP4.
min_dB	the minimum decibel (quietest sound) to include in the spec; defaults to -30 (-40 would include quieter sounds; -20 would cut out all but very loud sounds)
wl	window length for the spectrogram (low values= higher temporal res; high values= higher freq. res). Default 512 is a good tradeoff; human speech would look better at 1024 or higher, giving higher frequency resolution.
ovlp	how much overlap (as percent) between sliding windows to generate spec? Default 90 looks good, but takes longer
wn	window name (slight tweaks on algorithm that affect smoothness of output) see spectro
specWidth	what width (in inches) would you like to make your PNG output be, if saving a static spec?
specHeight	what height (in inches) would you like to make your PNG output be, if saving a static spec?
colbins	default 30: increasing can smooth the color contours, but take longer to generate spec
ampThresh	amplitude threshold as a percent to cut out of recording (try 5 to start); default= no filtering (high data loss with this; not recommended; play with min_dB and ampTrans first)
bgFlood	do you want the background color to spill into the axis margins? Default=FALSE (i.e. white margins)
fontAndAxisCol	the color of legend text if onlyPlotSpec=TRUE (since margins will be white, with black text); if bgFlood=TRUE, this will be the color of axis margins, labels and legend text. If you don't supply this, it will be picked automatically to be white or black based on supplied bg color
optim	NULL by default; this is an experimental feature to simplify the dataframe of the FFT-processed waveform used to generate the spectrogram (currently does nothing)
...	Other arguments to be passed for rendering the spec (i.e. to seewave::spectro)

Value

a list with all spectrogram parameters, segmented WAV files (segWavs) and spectrograms spec; importantly, spec is a list of n=number of "pages"/segments; the first page is displayed by default

Author(s)

Matthew R Wilkins (<matt@galacticpolymath.com>)

References

Araya-Salas M & Wilkins M R. (2020). *dynaSpec: dynamic spectrogram visualizations in R*. R package version 1.0.0.

See Also

paged_spectro

Examples

```
## Not run:
require(dynaSpec)
f <- list.files(pattern=".wav", full.names = TRUE, path = system.file(package="dynaSpec"))

# default behavior should be a decent start for good recordings; doesn't save anything, just plots
prep_static_ggspectro(f[1])

# to use with paged_spectro or to do other stuff, you need to assign the
# resulting object, but it will still always plot the first spec
# let's add axes and boost the signal a smidge
femaleBarnSwallow <- prep_static_ggspectro(f[1],destFolder="wd",
onlyPlotSpec = FALSE, bgFlood=TRUE,ampTrans=2)

# feels like we're missing a little bit of the quieter signals; let's lower
# the minimum amplitude threshold a bit
femaleBarnSwallow<-prep_static_ggspectro(f[1],destFolder="wd",
onlyPlotSpec = FALSE, bgFlood=TRUE,ampTrans=2,min_dB=-35)

#now for a male song
maleBarnSwallow<-prep_static_ggspectro(f[2],destFolder="wd",onlyPlotSpec = FALSE,
bgFlood=TRUE)

#Nice, but the trill is fading out; I'm gonna signal boost and lower the min_dB
maleBarnSwallow<-prep_static_ggspectro(f[2],destFolder="wd",onlyPlotSpec = FALSE,
bgFlood=TRUE,ampTrans=2,min_dB=-40)

#much stronger, now let's combine them
(you need the patchwork package to use the / operator to stack plots)
library(patchwork)
(femaleBarnSwallow$spec[[1]]+ggplot2::xlim(0,5)) /
(maleBarnSwallow$spec[[1]]+ggplot2::xlim(0,5))  +
patchwork::plot_annotation(title="Female and Male barn swallow songs",
caption="Female song (top) is much shorter, but similar
complexity to males. See: MR Wilkins et al. (2020) Animal
 Behaviour 168")

# ggplot2::ggsave("M&F_barn_swallow_song_specs.jpeg",width=11,height=7)

# see more examples at https://marce10.github.io/dynaSpec/

## End(Not run)
```

scrolling_spectro *Create scrolling dynamic spectrograms*

Description

scrolling_spectro create videos of single row spectrograms scrolling from right to left sync'ed with sound.

Usage

```
scrolling_spectro(wave, file.name = "scroll.spectro.mp4", hop.size = 11.6, wl = NULL,
ovlp = 70, flim = NULL, pal = seewave::reverse.gray.colors.1, speed = 1, fps = 50,
t.display = 1.5, fix.time = TRUE, res = 70,
width = 700, height = 400, parallel = 1, pb = TRUE,
 play = TRUE, loop = 1, lcol = "#07889B99",
 lty = 2, lwd = 2, axis.type = "standard", buffer = 1,
 ggspectro = FALSE, lower.spectro = TRUE, height.prop = c(5, 1), derivative = FALSE,
 osc = FALSE, colwave = "black", colbg = "white",
 spectro.call = NULL, annotation.call = NULL, ...)
```

Arguments

wave	object of class 'Wave'.
file.name	Character string with the name of the output video file. Must include the .mp4 extension. Default is 'scroll.spectro.mp4'.
hop.size	A numeric vector of length 1 specifying the time window duration (in ms). Default is 11.6 ms, which is equivalent to 512 wl for a 44.1 kHz sampling rate. Ignored if 'wl' is supplied.
wl	A numeric vector of length 1 specifying the window length of the spectrogram, default is NULL. If supplied, 'hop.size' is ignored.
ovlp	Numeric vector of length 1 specifying the percent overlap between two consecutive windows, as in spectro. Default is 70.
flim	A numeric vector of length 2 specifying limits in the frequency axis (in kHz). Default is NULL (which means from 0 to Nyquist frequency).
pal	Character string with the color palette to be used. Default is 'reverse.gray.colors.1'.
speed	Numeric vector of length 1 indicating the speed at which the sound file will be reproduced (default is 1, normal speed). Values < 1 (but higher than 0) slow down while values > 1 speed up. Note that changes in speed are achieved by modifying the number of frames per second in the output video. Hence, you may want to adjust 'fps' if video quality is considerably affected.
fps	Numeric vector of length 1 specifying the number of frames per second.
t.display	Numeric vector of length 1 specifying the time range displayed in the spectrogram.

`fix.time` Logical argument to control if the time axis moves along with the spectrogram or remains fixed. Default is `TRUE` (fixed).

`res` Numeric vector of length 1 specifying the resolution of the image files (see png).

`width` Numeric vector of length 1 specifying width of the video frame in pixels (see png). Default is 700.

`height` Numeric vector of length 1 specifying height of the video frame in pixels (see png). Default is 400.

`parallel` Numeric vector of length 1. Controls whether parallel computing is applied by specifying the number of cores to be used. Default is 1 (i.e. no parallel computing).

`pb` Logical argument to control if progress bar is shown. Default is `TRUE`.

`play` Logical argument to control if the video is played after generated. Default is `TRUE`.

`loop` Logical argument to control if the video is formatted to be played in a loop (i.e. if ends at the start of the clip).

`lcol` Character string with the color to be used for the vertical line at which sounds are played. Default is `"#07889B99"`.

`lty` Character string to control the type of the line at which sounds are played. 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).Default is 2.

`lwd` Character string to control the width of the line at which sounds are played. Default is 2.

`axis.type` Character string to control the style of spectrogram axes. Currently there are 3 options:

- `standard`: Both Y and X axes are printed as in the default `spectro` view.
- `minimal`: Single lines are used to denote the range defined by 1 s and 1 kHz for the X and Y axes respectively.
- `none`: No axis is printed (also removes ticks, tick labels, and axis labels).

`buffer` Numeric vector of length 1 (> 0) specifying the time to delay the start of the spectrogram scrolling (in seconds). Default is 1. Not available when loop is > 1.

`ggspectro` Logical argument to control if a ggspectro (`ggspectro`) is used instead. Note that there is much less control on display parameters when `ggpsectro = TRUE`. Default is `FALSE`.

`lower.spectro` Logical argument to control if a spectrogram of the full wave object is plotted at the bottom of the graph. Default is `TRUE`.

`height.prop` Numeric vector of length 2 to control the relative height of the scrolling and lower spectro, respectively. Default is `c(5, 1)`. Ignored if `lower.spectro = FALSE`.

`derivative` Logical argument to control if spectral derivative is used instead of spectrogram (as in Sound Analysis Pro, see `deriche`). Default is `FALSE`.

osc	Logical argument to control if the oscillogram is plotted at the bottom of the spectrogram. Default is FALSE. Note that 'osc' and 'lower.spectro' are mutually exclusive.
colwave	Character string to control the color of the oscillogram. Default is 'black'.
colbg	Character string to control the background color. Default is 'white'.
spectro.call	A call from a spectrogram creating function (i.e. spectro, color_spectro) generated by the function call. This call will replace the internal spectrogram creating call. Default is NULL.
annotation.call	A call from text generated by the function call. The call should also include the argmuents 'start' and 'end' to indicate the time at which the labels are displayed (in s).'fading' is optional and allows fade-in and fade-out effects on labels (in s as well). The position ('x' and 'y' arguments) should be between 0 and 1: x = 0, y = 0 corresponds to the bottom left and x = 1, y = 1 corresponds to the top right position.
...	Additional arguments to be passed to spectro for customizing spectrograms. Note that 'scale' cannot be included.

Details

The function creates videos (mp4 format) of single row spectrograms scrolling from right to left. The audio is sync'ed with the spectrograms. Sound files with a sampling rate other than 44.1 kHz will be resampled to 44.1 kHz as required by ffmpeg when embeding audio to video files.

Value

A video file in mp4 format in the working directory with the scrolling spectrogram.

Author(s)

Marcelo Araya-Salas (<marcelo.araya@ucr.ac.cr>)

References

Araya-Salas M & Wilkins M R. (2020). dynaSpec: dynamic spectrogram visualizations in R. R package version 1.0.0.

See Also

spectro

Examples

```
## Not run:
# load example data
data(list = c("Phae.long1"))

# run function
scrolling_spectro(wave = Phae.long1, wl = 300, ovlp = 90,
```

```
fps = 50, t.display = 1.5, collevels = seq(-40, 0, 5),
 pal = reverse.heat.colors, grid = FALSE, flim = c(1, 10),
 res = 120)

## End(Not run)
```

Index