Package 'praatpicture'

May 13, 2025

Title 'Praat Picture' Style Plots of Acoustic Data

Version 1.4.3

Description Quickly and easily generate plots of acoustic data aligned with transcriptions similar to those made in 'Praat' using either derived signals generated directly in R with 'wrassp' or imported derived signals from 'Praat'. Provides easy and fast out-of-the-box solutions but also a high extent of flexibility. Also provides options for embedding audio in figures and animating figures.

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Imports av (>= 0.9.0), bslib (>= 0.6.1), crayon (>= 1.5.2), emuR (>= 2.4.2), gifski (>= 1.12.0.2), graphics (>= 4.3.2), grDevices (>= 4.3.2), gsignal (>= 0.3.5), ipa (>= 0.1.0), phonTools (>= 0.2.2.2), reticulate, rPraat (>= 1.3.2.1), rstudioapi (>= 0.15.0), shiny (>= 1.8.1.1), shinyjs (>= 2.1.0), soundgen (>= 2.6.2), tuneR (>= 1.4.6), utils (>= 4.3.2), wrassp (>= 1.0.4), zoo (>= 1.8-12)

URL <https://github.com/rpuggaardrode/praatpicture>

BugReports <https://github.com/rpuggaardrode/praatpicture/issues>

NeedsCompilation no

Author Rasmus Puggaard-Rode [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-4522-9987>)

Maintainer Rasmus Puggaard-Rode <r.puggaard@phonetik.uni-muenchen.de>

Repository CRAN

Date/Publication 2025-05-13 08:40:05 UTC

Contents

conv2sc   *Convert capital letters to Unicode small caps*

Description

Helper function to convert capital letters into Unicode small caps. May not work for all font families. Note that there's no Unicode small cap 'X', so 'X' will just be converted to 'x'.

Usage

```
conv2sc(x)
```

Arguments

x   A string where all capital letters should be converted to small caps.

Value

A string where all capital letters have been converted to small caps.

Examples

```
my_string <- 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
conv2sc(my_string)
```

---

draw_arrow   *Draw arrow on praatpicture plot component*

Description

Helper function for drawing arrows on plot components made with praatpicture. Do not use directly, instead use [praatpicture](#) with the `draw_arrow` argument.

Usage

```
draw_arrow(plot_component, args)
```

Arguments

`plot_component`	String giving the name of the plot component to draw on.
`args`	A list of vectors giving arguments used for drawing arrows. See [praatpicture](#) documentation.

Value

No return values, called internally by [praatpicture](#) and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, draw_arrow=c('spectrogram', 0.1, 500, 0.4, 2000))
```

---

draw_lines   *Draw straight lines on plot component*

Description

Helper function for adding straight lines to plot components made with praatpicture. Do not use directly, instead use [praatpicture](#) with the `draw_lines` argument.

Usage

```
draw_lines(plot_component, args)
```

Arguments

`plot_component`	String giving the name of the plot component to draw on.
`args`	A list of vectors giving arguments used for drawing straight lines. See [praatpicture](#) documentation.

Value

No return values, called internally by praatpicture and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, draw_lines=c('spectrogram',
h=seq(0,5000,by=1000), lty='dashed'))
```

---

draw_rectangle   *Draw rectangle on praatpicture plot component*

Description

Helper function for drawing rectangles on plot components made with praatpicture. Do not use directly, instead use praatpicture with the draw_rectangle argument.

Usage

```
draw_rectangle(plot_component, args)
```

Arguments

plot_component String giving the name of the plot component to draw on.

args   A list of vectors giving arguments used for drawing rectangles. See praatpicture
                documentation.

Value

No return values, called internally by praatpicture and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, draw_rectangle=c('spectrogram', 0.1, 500, 0.4, 2000))
```

---

emupicture   *Make Praat Picture style plots in EMU*

Description

Generate plots in the style of Praat Pictures from sound files and annotations in an EMU database.

Usage

```
emupicture(
  db_handle,
  session = "0000",
  bundle,
  pitch_ssffExt = NULL,
  formant_ssffExt = NULL,
  intensity_ssffExt = NULL,
  talking = FALSE,
  ...
)
```

Arguments

db_handle	The handle of an EMU database loaded into R.
session	String giving the name of the session where the sound file to plot is located. Default is 0000.
bundle	String giving the name of the bundle with the sound file to plot.
pitch_ssffExt	String giving the file extension for an SSFF track with pitch data to plot. Default is NULL.
formant_ssffExt	
	String giving the file extension for an SSFF track with formant data to plot. Default is NULL.
intensity_ssffExt	
	String giving the file extension for an SSFF track with intensity data to plot. Default is NULL.
talking	Boolean; should a video be created with embedded audio, as when using [talking_praatpicture](#)? Default is FALSE.
...	Further arguments passed to [praatpicture](#) (or [talking_praatpicture](#) if talking = TRUE).

Value

No return value, produces a plot or a video.

See Also

See [praatpicture](#) for more details on how to customize plots and [talking_praatpicture](#) for more details on how to customize videos.

Examples

```
# Create demo data and load demo database
emuR::create_emuRdemoData(tempdir())
db_path <- paste0(tempdir(), '/emuR_demoData/ae_emuDB')
db <- emuR::load_emuDB(db_path)

emuR::list_bundles(db)
emupicture(db, bundle='msajc003', tg_tiers=c('Text', 'Tone'))

# Plot SSFF track data

emuR::list_ssffTrackDefinitions(db)
emupicture(db, bundle='msajc003', frames=c('sound', 'formant'),
proportion=c(30,70), formant_ssffExt='fms', formant_number=4)
```

---

formantplot   *Plot formant object*

Description

Function for plotting formant objects called by praatpicture. Instead of using this function directly,
just use praatpicture('my_sound_file', frames='formant').

Usage

```
formantplot(
  fm,
  start,
  end,
  tfrom0 = TRUE,
  tgbool = FALSE,
  lines = NULL,
  focusTierColor = "black",
  focusTierLineType = "dotted",
  dynamicRange = 30,
  freqRange = c(0, 5500),
  plotType = "speckle",
  color = "black",
  ind = NULL,
  min_max_only = FALSE,
  highlight = NULL,
  axisLabel = "Frequency (Hz)",
  drawSize = 1,
  speckleSize = 1
)
```

Arguments

fm	Formant object loaded using rPraat::formant.read
start	Start time (in seconds) of desired plotted area.
end	End time (in seconds) of desired plotted area.
tfrom0	Logical; should time on the x-axis run from 0 or from the original time? Default is TRUE.
tgbool	Logical; should dotted lines be plotted corresponding to locations in a TextGrid? Default is FALSE.
lines	Numeric vector giving locations in seconds of locations from a TextGrid to be plotted with dotted lines. Default is NULL.
focusTierColor	String or vector of strings giving the color(s) to use for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth color. Default is 'black'.
focusTierLineType	
	String or vector of strings giving the line type(s) for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth line type. Default is 'dotted'.
dynamicRange	Dynamic range in dB for producing formant plots. When a formant plot of plotType='speckle' is drawn, no formants are shown in frames with intensity level dynamicRange below the maximum intensity. Default is 30. If set to 0, all formants are shown.
freqRange	Vector of two integers giving the frequency range to be used for producing formant plots. Default is c(0,5500).
plotType	String giving the type of formant plot to produce; default is speckle (a point plot), the only other option is draw (a line plot). Alternatively a vector c('draw','speckle') can be passed, in which case both are used.
color	String or vector of strings giving the name(s) of colors to be used for plotting formants. If one color is provided, all formants will be plotted in this color. If multiple colors are provided, different formants will be shown in different colors. Default is 'black'.
ind	Integer indexing waveform relative to other plot components. Default is NULL.
min_max_only	Logical; should only minimum and maximum values be given on the y-axis? Default is FALSE. Can also be a logical vector if some but not all plot components should have minimum and maximum values on the y-axis. Ignored for TextGrid component.
highlight	Named list giving parameters for differential highlighting of formants based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the start and end arguments which must be numbers or numeric vectors, or using the tier and label arguments to highlight based on information in a plotted TextGrid. Further contains the optional arguments color (string or vector of strings, see color), drawSize or speckleSize (both numeric), and background (a string specifying a background color).
axisLabel	String giving the name of the label to print along the y-axis when plotting formants. Default is Frequency (Hz).

drawSize	Number indicating the line width if plotType is 'draw'. Default is 1. Controls the lwd argument of graphics::lines.
speckleSize	Number indicating the point size of if _plotType is 'speckle'. Default is 1. Controls the cex arguments of graphics::points.

Value

No return values, called internally by praatpicture and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, frames='formant')
```

---

intensityplot   *Plot intensity object*

Description

Function for plotting intensity objects called by praatpicture. Instead of using this function directly, just use praatpicture('my_sound_file', frames='intensity').

Usage

```
intensityplot(
  it,
  start,
  end,
  tfrom0 = TRUE,
  tgbool = FALSE,
  lines = NULL,
  focusTierColor = "black",
  focusTierLineType = "dotted",
  range = NULL,
  color = "black",
  ind = NULL,
  min_max_only = TRUE,
  highlight = NULL,
  axisLabel = "Intensity (dB)",
  drawSize = 1
)
```

Arguments

it	IntensityTier object loaded using [rPraat::it.read](#)
start	Start time (in seconds) of desired plotted area.
end	End time (in seconds) of desired plotted area.
tfrom0	Logical; should time on the x-axis run from 0 or from the original time? Default is TRUE.
tgbool	Logical; should dotted lines be plotted corresponding to locations in a TextGrid? Default is FALSE.
lines	Numeric vector giving locations in seconds of locations from a TextGrid to be plotted with dotted lines. Default is NULL.
focusTierColor	String or vector of strings giving the color(s) to use for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth color. Default is 'black'.
focusTierLineType	
	String or vector of strings giving the line type(s) for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth line type. Default is 'dotted'.
range	Vector of two integers giving the intensity range to be used for producing intensity plots. Default is NULL, in which case the range is simply the minimum and maximum levels in the curve.
color	String giving the name of the color to be used for plotting intensity. Default is 'black'.
ind	Integer indexing waveform relative to other plot components. Default is NULL.
min_max_only	Logical; should only minimum and maximum values be given on the y-axis? Default is TRUE. Can also be a logical vector if some but not all plot components should have minimum and maximum values on the y-axis. Ignored for TextGrid component.
highlight	Named list giving parameters for differential highlighting of the intensity contour based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the start and end arguments which must be numbers or numeric vectors, or using the tier and label arguments to highlight based on information in a plotted TextGrid. Further contains the optional arguments color (string or vector of strings, see color) and drawSize (integer), and background (a string specifying a background color).
axisLabel	String giving the name of the label to print along the y-axis when plotting intensity. Default is Intensity (dB).
drawSize	Number indicating the line width of the intensity contour. Default is 1. Controls the lwd argument of [graphics::lines](#).

Value

No return values, called internally by [praatpicture](#) and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, frames='intensity')
```

---

make_annot   *Annotate praatpicture plot component*

Description

Helper function for annotating plot components made with praatpicture. Do not use directly, instead use praatpicture with the annotate argument.

Usage

```
make_annot(plot_component, args)
```

Arguments

plot_component	String giving the name of the plot component to annotate.
args	A list of vectors giving arguments used for annotating. See praatpicture documentation.

Value

No return values, called internally by praatpicture and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, annotate=c('spectrogram', 0.25, 1500,
'An annotation'))
```

---

make_TextGrid   *Interactively create a TextGrid object*

Description

Annotate a sound file by interacting with waveform or spectrogram plots, resulting in a TextGrid object which can be used for creating various acoustic plots with time-aligned annotations with praatpicture().

Usage

```
make_TextGrid(
  sound,
  tierNames,
  start = 0,
  end = 0,
  audioInViewer = TRUE,
  show = "wave",
  channel = 1,
  sampa2ipa = FALSE
)
```

Arguments

sound	String giving the file name of a sound file with the .wav extension.
tierNames	String or vector of strings giving the name(s) of tiers in the new TextGrid object.
start	Start time (in seconds) of desired plotted area. Default is 0.
end	End time (in seconds) of desired plotted area. Default is 0 (= the entire file).
audioInViewer	Logical; should audio be playable from the Viewer pane in RStudio?
show	String giving the type of plot to show. Default is wave, another option is spectrogram. Note that spectrogram plotting is relatively slow within this function.
channel	Number indicating which audio channel to show. Default is 1.
sampa2ipa	Logical; should SAMPA transcriptions be converted to IPA? Default is FALSE.

Details

Running this function will show either a waveform or a spectrogram in a separate X11 graphics device window. Click on this figure in the locations where you want to add boundaries to your TextGrid objects. This should be done sequentially, starting with the first boundary along the time axis and ending with the last. It does not matter where on the y-axis you click.

Once you have indicated all the desired boundaries, you will be prompted in the R console to say whether the tier is an interval tier or a point tier by typing y (for interval tier) or n (for point tier). Subsequently you will be prompted in the console to write labels corresponding to each interval or point.

If you are creating a TextGrid with multiple tiers (i.e., if tierNames is longer than 1), this process will be repeated for all tiers.

Value

A list object identical to those created by [rPraat::tg.read()](#) when loading TextGrid objects into
R. This object can be passed to the tg_obj argument when using praatpicture.

See Also

make_TextGrid() is largely a wrapper around the function [tg_createTier()](#) which does most of
the work.

Examples

```
## Not run:
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/2.wav')
tg <- make_TextGrid(soundFile, tierNames=c('Mary', 'John', 'Bell'))
# Follow the steps shown in the console

praatpicture(soundFile, tg_obj=tg)

## End(Not run)
```

---

pitchplot   *Plot pitch object*

Description

Function for plotting pitch objects called by [praatpicture.](#) Instead of using this function directly,
just use praatpicture('my_sound_file', frames='pitch').

Usage

```
pitchplot(
  pt,
  start,
  end,
  tfrom0 = TRUE,
  tgbool = FALSE,
  lines = NULL,
  focusTierColor = "black",
  focusTierLineType = "dotted",
  plotType = "draw",
  scale = "hz",
  freqRange = NULL,
  semitonesRe = 100,
  color = "black",
  ind = NULL,
  min_max_only = TRUE,
  highlight = NULL,
```

```
    axisLabel = NULL,
    drawSize = 1,
    speckleSize = 1
)
```

Arguments

pt	PitchTier object loaded using rPraat::pt.read
start	Start time (in seconds) of desired plotted area.
end	End time (in seconds) of desired plotted area.
tfrom0	Logical; should time on the x-axis run from 0 or from the original time? Default is TRUE.
tgbool	Logical; should dotted lines be plotted corresponding to locations in a TextGrid? Default is FALSE.
lines	Numeric vector giving locations in seconds of locations from a TextGrid to be plotted with dotted lines. Default is NULL.
focusTierColor	String or vector of strings giving the color(s) to use for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth color. Default is 'black'.
focusTierLineType	
	String or vector of strings giving the line type(s) for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth line type. Default is 'dotted'.
plotType	String giving the type of pitch plot to produce; default is draw (a line plot), the only other option is speckle (a point plot). Alternatively a vector c('draw','speckle') can be passed, in which case both are used.
scale	String giving the frequency scale to use when producing pitch plots. Default is hz; other options are logarithmic (also in Hz), semitones, erb, and mel.
freqRange	Vector of two integers giving the frequency range to be used for producing pitch plots. Default is NULL, in which case the pitch range is automatically reset to c(-12,30) for the semitones scale, c(0,10) for the erb scale, and c(50,500) for the Hz-based scales, following Praat defaults.
semitonesRe	Frequency in Hz giving the reference level for converting pitch frequency to semitones. Default is 100.
color	String giving the name of the color to be used for plotting pitch. Default is 'black'.
ind	Integer indexing waveform relative to other plot components. Default is NULL.
min_max_only	Logical; should only minimum and maximum values be given on the y-axis? Default is TRUE. Can also be a logical vector if some but not all plot components should have minimum and maximum values on the y-axis. Ignored for TextGrid component.
highlight	Named list giving parameters for differential highlighting of pitch based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the start and end arguments which must be numbers or numeric vectors, or using the tier and label arguments to highlight

based on information in a plotted TextGrid. Further contains the optional arguments `color` (string or vector of strings, see `color`), `drawSize` or `speckleSize` (both numeric), and `background` (a string specifying a background color).

axisLabel	String giving the name of the label to print along the y-axis when printing a pitch track. Default is NULL, in which case the axis label will depend on the scale.
drawSize	Number indicating the line width if `plotType` is `'draw'`. Default is 1. Controls the `lwd` argument of [graphics::lines](#).
speckleSize	Number indicating the point size of if `_plotType` is `'speckle'`. Default is 1. Controls the `cex` arguments of [graphics::points](#).

Value

No return values, called internally by [praatpicture](#) and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, frames='pitch')
```

---

praatanimation	*Make animations from Praat Picture-style plots of acoustic data*

Description

Animate some aspect of a Praat Picture-style plot of acoustic data, potentially aligned with transcriptions.

Usage

```
praatanimation(
  sound,
  width = 1080,
  height = 720,
  frameRate = 24,
  n_frames = 50,
  loop = TRUE,
  outputFile = NULL,
  outputFormat = "gif",
  useViewer = TRUE,
  verbose = TRUE,
  pointsize = 25,
  start = 0,
  end = 0,
  spec_freqRange = c(0, 5000),
  spec_windowLength = 0.005,
```

```
    spec_dynamicRange = 50,
    spec_timeStep = 1000,
    pitch_timeStep = NULL,
    pitch_floor = 50,
    pitch_ceiling = 600,
    pitch_freqRange = c(50, 500),
    pitch_semitonesRe = 100,
    formant_timeStep = NULL,
    formant_windowLength = 0.025,
    formant_dynamicRange = 30,
    formant_freqRange = c(50, 5500),
    intensity_timeStep = NULL,
    intensity_minPitch = 100,
    intensity_range = NULL,
    ...
)
```

Arguments

sound	String giving the file name of a sound file with the .wav extension.
width	Number giving the desired width of the resulting animation in pixels; default is 1080.
height	Number giving the desired height of the resulting animation in pixels; default is 720.
frameRate	Number giving the desired frame rate of the resulting animation in Hz; default is 24, i.e. 24 frames per second.
n_frames	Number giving the desired number of frames of the resulting animation; default is 50.
loop	Logical; should the animation be looped? Default is TRUE. Ignored when outputType is mp4.
outputFile	String giving the desired file name of the animation. Default is NULL, in which case GIF files are named praatgif.gif and MP4 files are named praatvid.mp4. If you choose a different name, make sure that the file extension matches the selected outputType.
outputFormat	String giving the desired file type; default is gif, the only other option is mp4.S
useViewer	Logical; should the animation be shown in the Viewer pane in RStudio? Default is TRUE; if true, the animation is only saved in a temporary directory, but can be downloaded from a browser.
verbose	Logical; should status messages be printed in the console as figures are being generated? Default is TRUE.
pointsize	Number; which point size should be used for text in the animation? Default is 25. See grDevices::png() for more details.
start	Start time (in seconds) of desired plotted area. Default is 0. Alternatively, a vector giving the first and last start time in the animation.
end	End time (in seconds) of desired plotted area. Default is 0 (= the entire file). Alternatively, a vector giving the first and last end time in the animation.

spec_freqRange   Vector of two integers giving the frequency range to be used for plotting spectro-
                 grams. Default is `c(0,5000)`. Alternatively, a vector of four integers giving the
                 first and last lowest frequency, followed by the first and last highest frequency in
                 the animation; i.e., `c(0,0,5000,10000)` will produce an animation where the
                 upper frequency boundary gradually increases from 5000 Hz to 10,000 Hz.

spec_windowLength
                 Window length in seconds for generating spectrograms. Default is `0.005`. Al-
                 ternatively, a vector giving the first and last window lengths in the animation.

spec_dynamicRange
                 Dynamic range in dB for generating spectrograms. The maximum intensity
                 minus `spec_dynamicRange` will all be printed in white. Default is 50. Alterna-
                 tively, a vector giving the first and last dynamic range values in the animation.

spec_timeStep   How many time steps should be calculated for spectrograms? Default is `1000`.
                 Alternatively, a vector giving the first and last time step values in the animation.

pitch_timeStep   Measurement interval in seconds for tracking pitch. Default is `NULL`, in which
                 case the measurement interval is equal to 0.75 / `pitch_floor`. Alternatively, a
                 vector giving the first and last measurement intervals in the animation.

pitch_floor   Frequency in Hz; no pitch candidates considered below this frequency. Default
                 is 75. Alternatively, a vector giving the first and last pitch floors to be used in
                 the animation.

pitch_ceiling   Frequency in Hz; no pitch candidates considered above this frequency. Default
                 is 600. Alternatively, a vector giving the first and last pitch ceilings to be used
                 in the animation.

pitch_freqRange
                 Vector of two integers giving the frequency range to be used for producing pitch
                 plots. Default is `c(50,500)`. If the frequency scales `semitones` or `erb` are
                 used, the pitch range is automatically reset to the Praat defaults for these scales
                 (`c(-12,30)` and `c(0,10)`, respectively). Alternatively, a vector of four integers
                 giving the first and last lowest frequency, followed by the first and last highest
                 frequency in the animation (see `spec_freqRange` for usage details).

pitch_semitonesRe
                 Frequency in Hz giving the reference level for converting pitch frequency to
                 semitones. Default is 100. Alternatively, a vector giving the first and last semi-
                 tone reference levels to be used in the animation.

formant_timeStep
                 Measurement interval in seconds for tracking formants. Default is `NULL`, in
                 which case the measurement interval is equal to `formant_windowLength / 4`.
                 Alternatively, a vector giving the first and last measurement intervals to be used
                 in the animation.

formant_windowLength
                 The effective duration of the analysis window used for tracking formants in
                 seconds; the actual duration of the analysis window is twice this value. Al-
                 ternatively, a vector giving the first and last window lengths to be used in the
                 animation.

formant_dynamicRange
                 Dynamic range in dB for producing formant plots. When a formant plot of
                 `formant_plotType='speckle'` is drawn, no formants are shown in frames with

> intensity level `formant_dynamicRange` below the maximum intensity. Default is `30`. If set to `0`, all formants are shown. Alternatively, a vector giving the first and last dynamic range levels to be used in the animation.

`formant_freqRange`

> Vector of two integers giving the frequency range to be used for producing formant plots. Default is `c(0,5500)`. Alternatively, a vector of four integers giving the first and last lowest frequency, followed by the first and last highest frequency in the animation (see `spec_freqRange` for usage details).

`intensity_timeStep`

> Measurement interval in seconds for tracking intensity. Default is `NULL`, in which case the measurement interval is equal to 0.8 * `intensity_minPitch`. Alternatively, a vector giving the first and last measurement intervals to be used in the animation.

`intensity_minPitch`

> Lowest pitch in Hz used when calculating intensity; default is `100`. Alternatively, a vector giving the first and last minimum pitch levels to be used in the animation.

`intensity_range`

> Vector of two integers giving the intensity range to be used for producing intensity plots. Default is `NULL`, in which case the range is simply the minimum and maximum levels in the curve. Alternatively, a vector of four integers giving the first and last lowest level, followed by the first and last highest level in the animation (see `spec_freqRange` for usage details).

`...`   Further arguments passed to `praatpicture`.

Value

No return value, produces an animated figure.

See Also

This function is a wrapper for either `gifski::save_gif()` or `av::av_capture_graphics()` used to produce animations based on `praatpicture()`. For more detail on your options, see the `praatpicture()` help file.

Examples

```
## Not run:
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')

# Show increasing frequency range
praatanimation(soundFile, spec_freqRange=c(0,0,4000,12000))

# Transition from narrowband to broadband spectrogram
praatanimation(soundFile, spec_windowLength=c(0.005,0.03))

# Etc.

## End(Not run)
```

---

praatpicture   *Make Praat Picture style plots of acoustic data*

Description

Generate plots of acoustic data aligned with transcriptions similar to those made with Praat Picture. The default is to produce a plot with a relatively small waveform, somewhat larger spectrogram, and the first tier of a TextGrid.

Usage

```
praatpicture(
  sound,
  start = 0,
  end = 0,
  tfrom0 = TRUE,
  tUnit = "s",
  frames = c("sound", "spectrogram", "TextGrid"),
  proportion = c(30, 50, 20),
  mainTitle = "",
  mainTitleAlignment = 0,
  start_end_only = TRUE,
  min_max_only = TRUE,
  drawSize = 1,
  speckleSize = 1,
  wave_channels = "all",
  wave_channelNames = FALSE,
  wave_color = "black",
  wave_lineWidth = 1,
  wave_highlight = NULL,
  tg_obj = NULL,
  tg_file = NULL,
  tg_tiers = "all",
  tg_focusTier = tg_tiers[1],
  tg_focusTierColor = "black",
  tg_focusTierLineType = "dotted",
  tg_tierNames = TRUE,
  tg_alignment = "central",
  tg_edgeLabels = "keep",
  tg_specialChar = FALSE,
  tg_color = "black",
  tg_highlight = NULL,
  spec_channel = NULL,
  spec_freqRange = c(0, 5000),
  spec_windowLength = 0.005,
  spec_dynamicRange = 50,
  spec_timeStep = 1000,
```

```
          spec_windowShape = "Gaussian",
          spec_colors = c("white", "black"),
          spec_axisLabel = "Frequency (Hz)",
          spec_highlight = NULL,
          pitch_timeStep = NULL,
          pitch_floor = 75,
          pitch_ceiling = 600,
          pitch_plotType = "draw",
          pitch_scale = "hz",
          pitch_freqRange = NULL,
          pitch_semitonesRe = 100,
          pitch_color = "black",
          pitch_plotOnSpec = FALSE,
          pitch_ssff = NULL,
          pitch_axisLabel = NULL,
          pitch_highlight = NULL,
          formant_timeStep = NULL,
          formant_maxN = 5,
          formant_windowLength = 0.025,
          formant_dynamicRange = 30,
          formant_freqRange = c(50, 5500),
          formant_number = NULL,
          formant_plotType = "speckle",
          formant_color = "black",
          formant_plotOnSpec = FALSE,
          formant_ssff = NULL,
          formant_axisLabel = "Frequency (Hz)",
          formant_highlight = NULL,
          intensity_timeStep = NULL,
          intensity_minPitch = 100,
          intensity_range = NULL,
          intensity_color = "black",
          intensity_plotOnSpec = FALSE,
          intensity_ssff = NULL,
          intensity_axisLabel = "Intensity (dB)",
          intensity_highlight = NULL,
          time_axisLabel = NULL,
          highlight = NULL,
          draw_lines = list("formant", h = seq(0, 10000, by = 1000), lty = "dotted"),
          draw_rectangle = NULL,
          draw_arrow = NULL,
          annotate = NULL,
          gender = "u",
          ...
)
```

Arguments

sound   String giving the file name of a sound file with the .wav extension.

start	Start time (in seconds) of desired plotted area. Default is 0.
end	End time (in seconds) of desired plotted area. Default is 0 (= the entire file).
tfrom0	Logical; should time on the x-axis run from 0 or from the original time? Default is TRUE.
tUnit	String giving the unit of time to print along the x-axis. Possible options are 's' (default) for seconds and 'ms' for milliseconds.
frames	String or vector of strings giving the frames that the plot should consist of. Default is sound, spectrogram, TextGrid. This requires a file with the extension .TextGrid and the same base name as the sound file. Other options are pitch, formant, and intensity. See details for more information.
proportion	Integer or vector of integers of the same size as frames giving the proportion in percents of the plotting area to be taken up by the individual frames. Default is c(30,50,20). If more or less than three frames are plotted and no proportions are given, frames will be of equal size.
mainTitle	String giving a title to print at the top of the plot. The default is an empty string, i.e. no title.
mainTitleAlignment	
	Number indicating the vertical alignment of the plot title, where 0 (default) indicates left-alignment, 1 indicates right-alignment, 0.5 indicates central alignment, etc, following the conventions of the adj argument of graphics::mtext.
start_end_only	Logical; should there only be ticks on the x-axis for start and end times? Default is TRUE.
min_max_only	Logical; should only minimum and maximum values be given on the y-axis? Default is TRUE. Can also be a logical vector if some but not all plot components should have minimum and maximum values on the y-axis. Ignored for TextGrid component.
drawSize	Number indicating the line width of plot components where the _plotType is 'draw' (i.e., pitch, formants, or intensity rendered as line plots). Default is 1. Controls the lwd argument of graphics::lines.
speckleSize	Number indicating the point size of plot components where the _plotType is 'speckle' (i.e. pitch or formants rendered as point plots). Default is 1. Controls the cex arguments of graphics::points.
wave_channels	Vector of numbers or strings giving either numeric identifiers of audio channels to plot of the names of audio channels to plot. Also understands 'all', which plots all channels and is the default.
wave_channelNames	
	Should names of audio channels be printed on the y-axis? If TRUE, names will be grabbed from the audio metadata if available. Alternatively, if two channels are available, they will be named left and right. If more or less than two channels are available, channels are named Cn, where n is the number of the channel. Alternatvely, a vector of strings can be provided with channel names. Default is FALSE.
wave_color	String giving the name of the color to be used for plotting the waveform. Default is 'black'. Alternatively a vector of strings, if different colors should be used for different channels.

wave_lineWidth	Number giving the line width to use for plotting the waveform. Default is 1.
wave_highlight	Named list giving parameters for differential highlighting of the waveform based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the start and end arguments which must be numbers or numeric vectors, or using the tier and label arguments to highlight based on information in a plotted TextGrid. Further contains the argument color (string, see wave_color), and background (a string specifying a background color).
tg_obj	A TextGrid object returned by the make_TextGrid() function.
tg_file	Path of file to be used for plotting TextGrid. Default is NULL, in which case the function searches for a TextGrid sharing the same base name as sound with the .TextGrid extension.
tg_tiers	Vector of numbers or strings giving either numeric identifiers of TextGrid tiers to plot or the names of TextGrid tiers to plot. Also understands 'all', which plots all tiers and is the default.
tg_focusTier	For which tier(s) should lines be shown on all acoustic plots giving the locations of boundaries? Vector of number or strings giving either numeric identifiers of TextGrid tiers or the names of TextGrid tiers to plot. Default is tg_tiers[1], i.e. the first tier given in the tg_tiers argument. Additionally accepts the string none, in which case no lines are shown on acoustic plots, and all, in which case lines from all tiers are shown on acoustic plots.
tg_focusTierColor	
	String or vector of strings giving the color(s) to use for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth color. Default is 'black'.
tg_focusTierLineType	
	String or vector of strings giving the line type(s) for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth line type. Default is 'dotted'.
tg_tierNames	Logical; should TextGrid tier names be printed along the y-axis? Default is TRUE.
tg_alignment	String giving the desired alignment of text in the TextGrids. Default is central; other options are left and right. Alternatively, a vector of strings if different alignments are needed.
tg_edgeLabels	String specifying how to handle TextGrid labels in interval tiers that fall partially before start or partially after end. Default is 'keep', where labels are kept at the center of the interval. Other options are 'center', where labels are recentered to the visible part of the interval, or 'discard', where such labels are ignored.
tg_specialChar	Logical; should Praat typesetting for special font types such as italic, bold, and small caps be converted into corresponding R-readable special font types. Default is FALSE, since special characters can create unfortunate text alignment artefacts. See https://www.fon.hum.uva.nl/praat/manual/Text_styles.html.
tg_color	String or vector of strings giving the name of the color(s) to be used for the text in TextGrids. Default is 'black'. If a vector is provided, different colors are used for different tiers.

tg_highlight	Named list giving parameters for differential highlighting of TextGrid intervals. This list should contain information about which intervals to highlight, using the `tier` and `label`. Further contains the argument `color`, and `background` (a string specifying a background color).
spec_channel	Numeric giving the channel that should be used to generate the spectrogram. Default is `1`. Generating spectrograms from multiple channels is not currently possible with `praatpicture`.
spec_freqRange	Vector of two integers giving the frequency range to be used for plotting spectrograms. Default is `c(0,5000)`.
spec_windowLength	
	Window length in seconds for generating spectrograms. Default is `0.005`.
spec_dynamicRange	
	Dynamic range in dB for generating spectrograms. The maximum intensity minus `spec_dynamicRange` will all be printed in white. Default is `50`.
spec_timeStep	How many time steps should be calculated for spectrograms? Default is `1000`.
spec_windowShape	
	String giving the name of the window shape to be applied to the signal when generating spectrograms. Default is `Gaussian`; other options are `square`, `Hamming`, `Bartlett`, `Hanning`, or `Blackman`. Note that the Gaussian window function provided by the `phonTools` package and used in `praatpicture()` does not have the same properties as the Gaussian window function used for spectral estimation in Praat; plotting a simple sine wave with high dynamic range will produce sidelobes in `praatpicture()` but not in Praat. It's recommended to use Blackman windows instead if you have this problem.
spec_colors	Vector of strings giving the names of colors to be used for plotting the spectrogram; default is `c('white', 'black')`. The first value is used for plotting the lowest visible amplitude, and the last for plotting the highest visible amplitude. Vectors with more than two color names can be used for plotting values in between in different colors.
spec_axisLabel	String giving the name of the label to print along the y-axis when plotting a spectrogram. Default is `Frequency (Hz)`.
spec_highlight	Named list giving parameters for differential highlighting of the spectrogram based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the `start` and `end` arguments which must be numbers or numeric vectors, or using the `tier` and `label` arguments to highlight based on information in a plotted TextGrid. Further contains the argument `colors` (vector of strings, see `spec_colors`).
pitch_timeStep	Measurement interval in seconds for tracking pitch. Default is `NULL`, in which case the measurement interval is equal to `0.75 / pitch_floor`.
pitch_floor	Frequency in Hz; no pitch candidates considered below this frequency. Default is `75`.
pitch_ceiling	Frequency in Hz; no pitch candidates considered above this frequency. Default is `600`.
pitch_plotType	String giving the type of pitch plot to produce; default is `draw` (a line plot), the only other option is `speckle` (a point plot). Alternatively a vector `c('draw','speckle')` can be passed, in which case both are used.

pitch_scale   String giving the frequency scale to use when producing pitch plots. Default is
               hz; other options are logarithmic (also in Hz), semitones, erb, and mel.

pitch_freqRange

               Vector of two integers giving the frequency range to be used for producing pitch
               plots. Default is NULL, in which case the pitch range is automatically reset to
               c(-12,30) for the semitones scale, c(0,10) for the erb scale, and c(50,500)
               for the Hz-based scales, following Praat defaults.

pitch_semitonesRe

               Frequency in Hz giving the reference level for converting pitch frequency to
               semitones. Default is 100.

pitch_color   String giving the name of the color to be used for plotting pitch. Default is
               'black'. If pitch_plotOnSpec=TRUE, axes will follow the same color scheme.
               Also if pitch_plotOnSpec=TRUE, a vector of two strings can be passed, in
               which case the second color is used for background highlighting.

pitch_plotOnSpec

               Boolean; should pitch be plotted on top of spectrogram? Default is FALSE.

pitch_ssff   An object of class AsspDataObj containing a pitch track. Default is NULL.

pitch_axisLabel

               String giving the name of the label to print along the y-axis when printing a
               pitch track. Default is NULL, in which case the axis label will depend on the
               scale. If pitch_plotOnSpec=TRUE, this label will be printed on the right-hand
               y-axis label.

pitch_highlight

               Named list giving parameters for differential highlighting of pitch based on the
               time domain. This list should contain information about which parts of the plot
               to highlight, either done with the start and end arguments which must be num-
               bers or numeric vectors, or using the tier and label arguments to highlight
               based on information in a plotted TextGrid. Further contains the optional ar-
               guments color (string or vector of strings, see pitch_color), drawSize or
               speckleSize (both numeric), and background (a string specifying a back-
               ground color).

formant_timeStep

               Measurement interval in seconds for tracking formants. Default is NULL, in
               which case the measurement interval is equal to formant_windowLength / 4.

formant_maxN   Integer giving the maximum number of formants to track. Default is 5.

formant_windowLength

               The effective duration of the analysis window used for tracking formants in
               seconds; the actual duration of the analysis window is twice this value.

formant_dynamicRange

               Dynamic range in dB for producing formant plots. When a formant plot of
               formant_plotType='speckle' is drawn, no formants are shown in frames with
               intensity level formant_dynamicRange below the maximum intensity. Default
               is 30. If set to 0, all formants are shown.

formant_freqRange

               Vector of two integers giving the frequency range to be used for producing for-
               mant plots. Default is c(0,5500).

formant_number   Number of formants to plot. Default is NULL, in which case all available formants
                 are plotted.

formant_plotType

                String giving the type of formant plot to produce; default is speckle (a point
                 plot), the only other option is draw (a line plot). Alternatively a vector c('draw','speckle')
                 can be passed, in which case both are used.

formant_color   String or vector of strings giving the name(s) of colors to be used for plotting
                 formants. If one color is provided, all formants will be plotted in this color.
                 If multiple colors are provided, different formants will be shown in different
                 colors. Default is 'black'. If formant_plotOnSpec=TRUE and the length of
                 this vector twice the number of formants plotted, the first half of strings will
                 be used for the formants' primary colors and the second half will be used for
                 background highlighting. If the length of this vector is one more than the number
                 of formants plotted, the last string will be used for background highlighting.

formant_plotOnSpec

                Boolean; should formants be plotted on top of spectrogram? Default is FALSE.

formant_ssff   An object of class AsspDataObj containing formant tracks. Default is NULL.

formant_axisLabel

                String giving the name of the label to print along the y-axis when plotting for-
                 mants. Default is Frequency (Hz).

formant_highlight

                Named list giving parameters for differential highlighting of formants based on
                 the time domain. This list should contain information about which parts of the
                 plot to highlight, either done with the start and end arguments which must be
                 numbers or numeric vectors, or using the tier and label arguments to high-
                 light based on information in a plotted TextGrid. Further contains the optional
                 arguments color (string or vector of strings, see formant_color), drawSize
                 or speckleSize (both numeric), and background (a string specifying a back-
                 ground color).

intensity_timeStep

                Measurement interval in seconds for tracking intensity. Default is NULL, in
                 which case the measurement interval is equal to 0.8 * intensity_minPitch.

intensity_minPitch

                Lowest pitch in Hz used when calculating intensity; default is 100

intensity_range

                Vector of two integers giving the intensity range to be used for producing inten-
                 sity plots. Default is NULL, in which case the range is simply the minimum and
                 maximum levels in the curve.

intensity_color

                String giving the name of the color to be used for plotting intensity. Default
                 is 'black'. If intensity_plotOnSpec=TRUE, axes will follow the same color
                 scheme. Also if intensity_plotOnSpec=TRUE, a vector of two strings can be
                 passed, in which case the second color is used for background highlighting.

intensity_plotOnSpec

                Boolean; should intensity be plotted on top of spectrogram? Default is FALSE.

intensity_ssff   An object of class AsspDataObj containing intensity tracks. Default is NULL.

intensity_axisLabel

String giving the name of the label to print along the y-axis when plotting intensity. Default is Intensity (dB). If intensity_plotOnSpec=TRUE, this label will be printed on the right-hand y-axis label.

intensity_highlight

Named list giving parameters for differential highlighting of the intensity contour based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the start and end arguments which must be numbers or numeric vectors, or using the tier and label arguments to highlight based on information in a plotted TextGrid. Further contains the optional arguments color (string or vector of strings, see intensity_color) and drawSize (integer), and background (a string specifying a background color).

time_axisLabel String giving the name of the label to print along the x-axis. Default is NULL, in which case Time (s) is printed if tUnit = 's' and Time (ms) is printed if tUnit = 'ms'.

highlight Named list giving parameters for differential highlighting of part of the plot based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the start and end arguments which must be numbers or numeric vectors, or using the tier and label arguments to highlight based on information in a plotted TextGrid. Further contains the optional arguments color (a string), drawSize and speckleSize (both numeric), and background (a string specifying a background color). This argument is used to highlight all plot components, use the *_highlight arguments for highlighting individuals plot components.

draw_lines Use for drawing straight lines on plot components. Takes an argument of type list which should contain a) a string giving the plot component to draw straight lines on, and b) arguments to pass on to graphics::abline. Should have a named argument h for horizontal lines, or v for vertical lines, or a,b for the intercept and slope of the line otherwise. Alternatively a nested list can be passed if more (sets of) lines should be drawn. If multiple audio channels are plotted and lines should be added to one of these, use the channel identifier instead of a string giving the frame to draw on. The default value is list('formant', h=seq(0,10000,by=1000), lty='dotted'). According to Praat defaults, this means that if formants are plotted in a separate frame, horizontal dotted lines (lty) are shown at 1000 Hz intervals. To override this behavior, simply pass draw_lines=NULL.

draw_rectangle Use for drawing rectangles on plot components. A vector containing a) a string giving the plot component to draw a rectangle on, and b) arguments to pass on to graphics::rect. Alternatively a list of such vectors, if more rectangles should be drawn. If multiple audio channels are plotted and a rectangle should be added to one of these, use the channel identifier instead of a string giving the frame to draw on.

draw_arrow Use for drawing arrows on plot components. A vector containing a) a string giving the plot component to draw an arrow on, and b) arguments to pass on to graphics::arrows. Alternatively a list of such vectors, if more arrows should be drawn. If multiple audio channels are plotted and an arrow should be added to

one of these, use the channel identifier instead of a string giving the frame to draw on.

annotate   Use for annotating plot components. A vector containing a) a string giving the plot component to annotate, and b) arguments to pass on to graphics::text. Alternatively a list of such vectors, if more annotations should be made. If multiple audio channels are plotted and annotations should be added to one of these, use the channel identifier instead of a string giving the frame to draw on.

gender   String indicating the gender of the speaker; default is u for unknown, other legal values are m and f. Used to tweak pitch and formant tracking parameters.

...   Further global plotting arguments passed on to par().

Details

When available, pitch, formant, and intensity tracks are loaded from Praat files with the same base name as sound; i.e., if your sound file is called 1.wav and there is a Praat file called 1.Formant in the same directory, this file is used for plotting formants. Pitch files should have either the PitchTier or Pitch extension, and intensity files should have the IntensityTier extension.

If no such files are available, the signal processing tools in the wrassp package are used; pitch is tracked with the function wrassp::ksvF0, formants are tracked with wrassp::forest, and intensity is tracked with wrassp::rmsana. Parameters are set to mimic Praat as closely as possible, e.g. using a Gaussian-like window shape KAISER2_0, but results will differ from Praat simply because the tracking algorithms differ; as far as I know, the Burg algorithm used by Praat for tracking formants isn't implemented in R, nor is the autocorrelation method for tracking pitch.

Spectrograms are generated with the function phonTools::spectrogram. The code portion that actually adds the spectrogram to a plot is based on phonTools::plot.spectrogram but rewritten to use a bitmap raster for rendering the image if the graphics device allows for it, which significantly speeds up rendering the spectrogram.

Value

No return value, produces a figure.

Examples

```
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile)
```

---

praat_pitchRawAC   *Estimate pitch using Praat's raw autocorrelation function*

Description

Estimate pitch using Praat's raw autocorrelation function

Usage

```
praat_pitchRawAC(
  sound,
  output = "ssff",
  timeStep = NULL,
  floor = 75,
  ceiling = 600,
  maxNoCandidates = 15,
  gaussianWindow = FALSE,
  silenceThreshold = 0.03,
  voicingThreshold = 0.45,
  octaveCost = 0.01,
  octaveJumpCost = 0.35,
  voicedUnvoicedCost = 0.14
)
```

Arguments

sound	String giving the file name of a sound file with the .wav extension.
output	String giving the output format. Should be either `ssff` for the SSFF-style format used by wrassp (default) or `df` to output a data frame.
timeStep	Measurement interval in seconds. Default is NULL, in which case the measurement interval is equal to 0.75 / `pitch_floor`.
floor	Frequency in Hz; no pitch candidates considered below this frequency. Default is 75.
ceiling	Frequency in Hz; no pitch candidates considered above this frequency. Default is 600.
maxNoCandidates	
	Integer giving the maximum number of pitch candidates to estimate. Default is 15.
gaussianWindow	Logical; should Gaussian windows be used for estimating pitch? Default is FALSE, in which case Hanning windows are used.
silenceThreshold	
	Numeric giving the silence threshold when estimating pitch in terms of amplitude relative to the global maximum. Default is 0.01.
voicingThreshold	
	Numeric giving the voicing threshold when estimating pitch in terms of fractional strength in the autocorrelation function. Default is 0.14.
octaveCost	Numeric specifying how much high frequency pitch candidates should be favored in terms of fractional strength in the autocorrelation function. Default is 0.01.
octaveJumpCost	Numeric specifying how much pitch changes should be disfavored in terms of fractional strength in the autocorrelation function. Default is 0.35.
voicedUnvoicedCost	
	Numeric specifying how much transitions in voicing value should be disfavored in terms of fractional strength in the autocorelation function. Default is 0.14.

Value

List in the SSFF format or data frame with estimated pitch.

Examples

```
## Not run:
1+1

## End(Not run)
```

---

shiny_praatpicture   *Run praatpicture as Shiny app*

Description

Interactive version of [praatpicture](#)

Usage

```
shiny_praatpicture()
```

Value

No return values

Examples

```
## Not run:
shiny_praatpicture()

## End(Not run)
```

---

specplot   *Plot spectrogram*

Description

Function for plotting spectrograms called by [praatpicture.](#) Instead of using this function directly, just use praatpicture('my_sound_file', frames='spectrogram').

Usage

```
specplot(
  sig,
  sr,
  t,
  start,
  end,
  tfrom0 = TRUE,
  freqRange = c(0, 5000),
  windowLength = 0.005,
  dynamicRange = 60,
  timeStep = 1000,
  windowShape = "Gaussian",
  colors = c("white", "black"),
  pitch_plotOnSpec = FALSE,
  pt = NULL,
  pitch_plotType = "draw",
  pitch_scale = "hz",
  pitch_freqRange = NULL,
  pitch_axisLabel = NULL,
  pitch_color = "black",
  pitch_highlight = NULL,
  formant_plotOnSpec = FALSE,
  fm = NULL,
  formant_plotType = "speckle",
  formant_dynamicRange = 30,
  formant_color = "black",
  formant_highlight = NULL,
  intensity_plotOnSpec = FALSE,
  it = NULL,
  intensity_range = NULL,
  intensity_axisLabel = "Intensity (dB)",
  intensity_color = "black",
  intensity_highlight = NULL,
  tgbool = FALSE,
  lines = NULL,
  focusTierColor = "black",
  focusTierLineType = "dotted",
  ind = NULL,
  min_max_only = TRUE,
  highlight = NULL,
  axisLabel = "Frequency (Hz)",
  drawSize = 1,
  speckleSize = 1
)
```

Arguments

sig	Numeric vector corresponding to a sound signal.
sr	Integer giving the sampling rate of the signal.
t	Numeric vector giving times corresponding to the signal.
start	Start time (in seconds) of desired plotted area.
end	End time (in seconds) of desired plotted area.
tfrom0	Logical; should time on the x-axis run from 0 or from the original time? Default is TRUE.
freqRange	Vector of two integers giving the frequency range to be used for plotting spectrograms. Default is c(0,5000).
windowLength	Window length in seconds for generating spectrograms. Default is 0.005.
dynamicRange	Dynamic range in dB for generating spectrograms. The maximum intensity minus dynamicRange will all be printed in white. Default is 50.
timeStep	How many time steps should be calculated for spectrograms? Default is 1000. Note that this takes a while to plot, so for fiddling with plotting parameters it is a good idea to choose a smaller value.
windowShape	String giving the name of the window shape to be applied to the signal when generating spectrograms. Default is Gaussian; other options are square, Hamming, Bartlett, Hanning, or Blackman. Note that the Gaussian window function provided by the phonTools package and used in praatpicture() does not have the same properties as the Gaussian window function used for spectral estimation in Praat; plotting a simple sine wave with high dynamic range will produce sidelobes in praatpicture() but not in Praat. It's recommended to use Blackman windows instead if you have this problem.
colors	Vector of strings giving the names of colors to be used for plotting the spectrogram; default is c('white', 'black'). The first value is used for plotting the lowest visible amplitude, and the last for plotting the highest visible amplitude. Vectors with more than two color names can be used for plotting values in between in different colors.
pitch_plotOnSpec	
	Boolean; should pitch be plotted on top of spectrogram? Default is FALSE.
pt	Pitch object loaded using rPraat::pt.read or similar object.
pitch_plotType	String giving the type of pitch plot to produce; default is draw (a line plot), the only other option is speckle (a point plot). Alternatively a vector c('draw','speckle') can be passed, in which case both are used.
pitch_scale	String giving the frequency scale to use when producing pitch plots. Default is hz; other options are logarithmic (also in Hz), semitones, erb, and mel.
pitch_freqRange	
	Vector of two integers giving the frequency range to be used for producing pitch plots. Default is NULL, in which case the pitch range is automatically reset to c(-12,30) for the semitones scale, c(0,10) for the erb scale, and c(50,500) for the Hz-based scales, following Praat defaults.

pitch_axisLabel

    String giving the name of the label to print along the y-axis when printing a pitch track. Default is `NULL`, in which case the axis label will depend on the scale.

pitch_color   String or vector of strings giving the name of the color to be used for plotting pitch. Default is `'black'`. If a vector of two strings is passed, the second color will be used for background highlighting.

pitch_highlight

    Named list giving parameters for differential highlighting of pitch based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the `start` and `end` arguments which must be numbers or numeric vectors, or using the `tier` and `label` arguments to highlight based on information in a plotted TextGrid. Further contains the optional arguments `color` (string or vector of strings, see `pitch_color`), `drawSize` or `speckleSize` (both numeric).

formant_plotOnSpec

    Boolean; should formants be plotted on top of spectrogram? Default is `FALSE`.

fm   Formant object loaded using [rPraat::formant.read](#) or similar object.

formant_plotType

    String giving the type of formant plot to produce; default is `speckle` (a point plot), the only other option is `draw` (a line plot). Alternatively a vector `c('draw','speckle')` can be passed, in which case both are used.

formant_dynamicRange

    Dynamic range in dB for producing formant plots. When a formant plot of `formant_plotType='speckle'` is drawn, no formants are shown in frames with intensity level `formant_dynamicRange` below the maximum intensity. Default is `30`. If set to `0`, all formants are shown.

formant_color   String or vector of strings giving the name(s) of colors to be used for plotting formants. If one color is provided, all formants will be plotted in this color. If multiple colors are provided, different formants will be shown in different colors. Default is `'black'`. If the length of this vector twice the number of formants plotted, the first half of strings will be used for the formants' primary colors and the second half will be used for background highlighting. If the length of this vector is one more than the number of formants plotted, the last string will be used for background highlighting.

formant_highlight

    Named list giving parameters for differential highlighting of formants based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the `start` and `end` arguments which must be numbers or numeric vectors, or using the `tier` and `label` arguments to highlight based on information in a plotted TextGrid. Further contains the optional arguments `color` (string or vector of strings, see `formant_color`), `drawSize` or `speckleSize` (both numeric).

intensity_plotOnSpec

    Boolean; should intensity be plotted on top of spectrogram? Default is `FALSE`.

it   Intensity object loaded using [rPraat::it.read](#) or similar object.

intensity_range

> Vector of two integers giving the intensity range to be used for producing intensity plots. Default is NULL, in which case the range is simply the minimum and maximum levels in the curve.

intensity_axisLabel

> String giving the name of the label to print along the y-axis when plotting intensity. Default is Intensity (dB).

intensity_color

> String or vector of strings giving the name of the color to be used for plotting intensity. Default is 'black'. If a vector of two strings is passed, the second color will be used for background highlighting.

intensity_highlight

> Named list giving parameters for differential highlighting of the intensity contour based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the start and end arguments which must be numbers or numeric vectors, or using the tier and label arguments to highlight based on information in a plotted TextGrid. Further contains the optional arguments color (string or vector of strings, see intensity_color) and drawSize (integer).

tgbool   Logical; should dotted lines be plotted corresponding to locations in a TextGrid? Default is FALSE.

lines   Numeric vector giving locations in seconds of locations from a TextGrid to be plotted with dotted lines. Default is NULL.

focusTierColor   String or vector of strings giving the color(s) to use for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth color. Default is 'black'.

focusTierLineType

> String or vector of strings giving the line type(s) for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth line type. Default is 'dotted'.

ind   Integer indexing waveform relative to other plot components. Default is NULL.

min_max_only   Logical; should only minimum and maximum values be given on the y-axis? Default is TRUE. Can also be a logical vector if some but not all plot components should have minimum and maximum values on the y-axis. Ignored for TextGrid component.

highlight   Named list giving parameters for differential highlighting of the spectrogram based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the start and end arguments which must be numbers or numeric vectors, or using the tier and label arguments to highlight based on information in a plotted TextGrid. Further contains the argument colors (vector of strings, see colors).

axisLabel   String giving the name of the label to print along the y-axis when plotting a spectrogram. Default is Frequency (Hz).

drawSize   Number indicating the line width of plot components where the _plotType is 'draw' (i.e., pitch, formants, or intensity rendered as line plots). Default is 1. Controls the lwd argument of graphics::lines.

speckleSize   Number indicating the point size of plot components where the _plotType is
                 'speckle' (i.e. pitch or formants rendered as point plots). Default is 1. Controls
                 the cex arguments of [graphics::points](#).

Value

No return values, called internally by [praatpicture](#) and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, frames='spectrogram')
```

---

talking_praatpicture   *Make Praat Picture style plots of acoustic data with embedded audio*

Description

Generate simple MP4 video files with Praat Picture style plots of acoustic data with time-aligned
transcriptions and embedded audio to use in presentations etc.

Usage

```
talking_praatpicture(
  sound,
  start = 0,
  end = 0,
  audio_start = start,
  audio_end = end,
  width = 1080,
  height = 720,
  pointsize = 25,
  outputFile = "praatvid.mp4",
  useViewer = TRUE,
  ...
)
```

Arguments

sound   String giving the file name of a sound file with the .wav extension.

start   Start time (in seconds) of desired plotted area. Default is 0.

end   End time (in seconds) of desired plotted area. Default is 0 (= the entire file).

audio_start   Start time (in seconds) of embedded audio. By default it is the same as start,
                 i.e. the embedded audio is the portion of the sound file that is being plotted.

audio_end	End time (in seconds) of embedded audio. By default it is the same as end, i.e. the embedded audio is the portion of the sound that is being plotted.
width	Number giving the desired width of the resulting animation in pixels; default is 1080.
height	Number giving the desired height of the resulting animation in pixels; default is 720.
pointsize	Number; which point size should be used for text in the animation? Default is 25. See grDevices::png() for more details.
outputFile	String giving the desired file name. Default is praatvid.mp4.
useViewer	Logical; should the video be shown in the Viewer pane in RStudio? Default is TRUE; if true, the video is oSnly saved in a temporary directory, but can be downloaded from a browser.
...	Further arguments passed to praatpicture.

Value

No return value, produces a video file.

See Also

This function is a wrapper for av::av_capture_graphics() used to produce plots similar to those made with praatpicture() with embedded audio. For more detail on your options, see the praatpicture() help file.

Examples

```
## Not run:
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
talking_praatpicture(soundFile)

## End(Not run)
```

---

tgplot   *Plot TextGrid*

Description

Function for plotting TextGrids called by praatpicture. Instead of using this function directly, just use praatpicture('my_sound_file', frames='TextGrid').

Usage

```
tgplot(
  tg,
  t,
  sr,
  start,
  end,
  tiers = 1,
  tfrom0 = TRUE,
  tierNames = TRUE,
  alignment = "central",
  edgeLabels = "keep",
  specialChar = FALSE,
  color = "black",
  highlight = NULL
)
```

Arguments

tg	TextGrid object loaded using [rPraat::tg.read](#)
t	Numeric vector giving times corresponding to the signal.
sr	Integer giving the sampling rate of the signal.
start	Start time (in seconds) of desired plotted area.
end	End time (in seconds) of desired plotted area.
tiers	Vector of number or strings giving either numeric identifiers of TextGrid tiers to plot or the names of TextGrid tiers to plot. Default is 1, which plots just the first tier.
tfrom0	Logical; should time on the x-axis run from 0 or from the original time? Default is TRUE.
tierNames	Logical; should TextGrid tier names be printed along the y-axis? Default is TRUE.
alignment	String giving the desired alignment of text in the TextGrids. Default is central; other options are left and right. Alternatively, a vector of strings if different alignments are needed.
edgeLabels	String specifying how to handle TextGrid labels in interval tiers that fall partially before start or partially after end. Default is 'keep', where labels are kept at the center of the interval. Other options are 'center', where labels are recentered to the visible part of the interval, or 'discard', where such labels are ignored.
specialChar	Logical; should Praat typesetting for special font types such as italic, bold, and small caps be converted into corresponding R-readable special font types. Default is FALSE, since special characters can create unfortunate text alignment artefacts. See https://www.fon.hum.uva.nl/praat/manual/Text_styles.html.
color	String or vector of strings giving the name of the color(s) to be used for the text in TextGrids. Default is 'black'. If a vector is provided, different colors are used for different tiers.

highlight   Named list giving parameters for differential highlighting of TextGrid intervals. This list should contain information about which intervals to highlight, using the `tier` and `label`. Further contains the argument `color`, and `background` (a string specifying a background color).

Value

No return values, called internally by [praatpicture](#) and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, frames='TextGrid')
```

---

   tg_createTier   *Interactively create a TextGrid tier*

Description

Function for creating TextGrid tiers called by [make_TextGrid](#). Instead of using this function directly, use [make_TextGrid](#).

Usage

```
tg_createTier(
  sound,
  tierName,
  start = 0,
  end = 0,
  show = "wave",
  channel = 1,
  sampa2ipa = FALSE
)
```

Arguments

sound	String giving the file name of a sound file with the .wav extension.
tierName	String giving the name of the tier.
start	Start time (in seconds) of desired plotted area. Default is `0`.
end	End time (in seconds) of desired plotted area. Default is `0` (= the entire file).
show	String giving the type of plot to show. Default is `wave`, another option is `spectrogram`. Note that spectrogram plotting is relatively slow within this function.
channel	Number indicating which audio channel to show. Default is 1.
sampa2ipa	Logical; should SAMPA transcriptions be converted to IPA? Default is `FALSE`.

Value

A list object identical to a single tier created by rPraat::tg.read() when loading TextGrid objects into R.

Examples

```
## Not run:
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/2.wav')
tg <- make_TextGrid(soundFile, tierNames='Mary')
# Follow the steps shown in the console

praatpicture(soundFile, tg_obj=tg)

## End(Not run)
```

---

tg_stylize   *Convert Praat font styles to R font styles*

Description

Helper function for converting Praat font styles such as italics, bold, and small caps into expressions that can be read by base R plots. Instead of using this function directly, just use `praatpicture('my_sound_file',`
`frames='TextGrid', tg_specialChar=TRUE)`.

Usage

```
tg_stylize(lab)
```

Arguments

lab   A string or vector of strings with labels from a TextGrid.

Value

A list with elements of class `expression`.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')

# With stylized text
praatpicture(soundFile, frames='TextGrid')

# Without stylized text
praatpicture(soundFile, frames='TextGrid', tg_specialChar=FALSE)
```

---

waveplot   *Plot waveform*

Description

Function for plotting waveforms called by praatpicture. Instead of using this function directly, just
use praatpicture('my_sound_file', frames='sound').

Usage

```
waveplot(
  sig,
  bit,
  t,
  nchan = 1,
  color = "black",
  tgbool = FALSE,
  lines = NULL,
  focusTierColor = "black",
  focusTierLineType = "dotted",
  ind = NULL,
  line_comp = NULL,
  rect_comp = NULL,
  arr_comp = NULL,
  annot_comp = NULL,
  draw_lines = NULL,
  draw_rectangle = NULL,
  draw_arrow = NULL,
  annotate = NULL,
  channelNames = FALSE,
  lineWidth = 1,
  cn = NULL,
  min_max_only = TRUE,
  highlight = NULL
)
```

Arguments

sig	Numeric vector corresponding to a sound signal.
bit	Numeric; will generally be grabbed from a loaded WaveMC object.
t	Numeric vector giving times corresponding to the signal.
nchan	Numeric; how many channels will be plotted? Default is 1.
color	String giving the name of the color to be used for plotting the waveform. Default is 'black'. Alternatively, a vector of colors, if different channels should be plotted with different colors.

`tgbool`	Logical; should dotted lines be plotted corresponding to locations in a TextGrid? Default is `FALSE`.
`lines`	Numeric vector giving locations in seconds of locations from a TextGrid to be plotted with dotted lines. Default is `NULL`.
`focusTierColor`	String or vector of strings giving the color(s) to use for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth color. Default is `'black'`.
`focusTierLineType`	
	String or vector of strings giving the line type(s) for plotting focus tier lines. If multiple tiers are focused, a vector of the same length can be passed, and the nth tier will be plotted in the nth line type. Default is `'dotted'`.
`ind`	Integer indexing waveform relative to other plot components. Default is `NULL`.
`line_comp`	Vector of strings or numbers giving plot components to draw straight lines on. Default is `NULL`.
`rect_comp`	Vector of strings or numbers giving plot components to draw rectangles on. Default is `NULL`.
`arr_comp`	Vector of strings of numbers giving plot components to draw arrows on. Default is `NULL`.
`annot_comp`	Vector of strings of numbers giving plot components to annotate. Default is `NULL`.
`draw_lines`	List of arguments for drawing straight lines passed from `praatpicture()`. Default is `NULL`.
`draw_rectangle`	List of arguments for drawing rectangles passed from `praatpicture()`. Default is `NULL`.
`draw_arrow`	List of arguments for drawing arrows passed from `praatpicture()`. Default is `NULL`.
`annotate`	List of arguments for annotating passed from `praatpicture()`. Default is `NULL`.
`channelNames`	Logical; should names of audio channels be printed on the y-axis? Default is `FALSE`.
`lineWidth`	Number giving the line width to use for plotting the waveform. Default is 1.
`cn`	Vector of strings with channel names to be printed on the y-axis if `channelNames` is `TRUE`.
`min_max_only`	Logical; should only minimum and maximum values be given on the y-axis? Default is `TRUE`. Can also be a logical vector if some but not all plot components should have minimum and maximum values on the y-axis. Ignored for TextGrid component.
`highlight`	Named list giving parameters for differential highlighting of the waveform based on the time domain. This list should contain information about which parts of the plot to highlight, either done with the `start` and `end` arguments which must be numbers or numeric vectors, or using the `tier` and `label` arguments to highlight based on information in a plotted TextGrid. Further contains the argument `color` (string, see `color`), and `background` (a string specifying a background color).

Value

No return values, called internally by praatpicture and sibling functions.

Examples

```
# Don't use directly
datapath <- system.file('extdata', package='praatpicture')
soundFile <- paste0(datapath, '/1.wav')
praatpicture(soundFile, frames='sound')
```

Index