Package 'voiceR'

May 8, 2026

Type Package

Title Voice Analytics for Social Scientists

Version 0.1.0

Author Francesc Busquet [aut, cre],
Christian Hildebrand [aut]

Maintainer Francesc Busquet <francesc.busquet@unisg.ch>

Description Simplifies and largely automates practical voice analytics for social science research. This package offers an accessible and easy-to-use interface, including an interactive Shiny app, that simplifies the processing, extraction, analysis, and reporting of voice recording data in the behavioral and social sciences. The package includes batch processing capabilities to read and analyze multiple voice files in parallel, automates the extraction of key vocal features for further analysis, and automatically generates APA formatted reports for typical between-group comparisons in experimental social science research. A more extensive methodological introduction that inspired the development of the 'voiceR' package is provided in Hildebrand et al. 2020 <doi:10.1016/j.jbusres.2020.09.020>.

License GPL-2

Encoding UTF-8

LazyData true

LazyDataCompression xz

Depends R (>= 4.1.0)

Imports stats, utils, doParallel, DT, foreach, FSA, ggplot2, ggpubr, ggthemes, grid, gridExtra, gtable, kableExtra, knitr, MASS, phia, plotly, seewave, tuneR, soundgen, rcompanion, rlang, rmarkdown, shiny, shinyFiles, shinyjs, stringr, xfun

RoxygenNote 7.2.3

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2023-09-12 20:30:02 UTC

Contents

`autoExtract` *Automatically analyze audio files*

Description

Automatically analyzes audio files and outputs a data.frame with their main extracted audio features.

Usage

```
autoExtract(
  path = ".",
  audioList = list(),
  filter = NA,
  fileType = "wav",
  fileNamePattern = "ID_Condition_Dimension",
  sep = "_",
  parallel = FALSE,
  recursive = FALSE,
```

```
  preprocess = FALSE,
  extended = FALSE,
  ...
)
```

Arguments

path	An optional character string indicating the path to the folder containing the audio files. Default corresponds to the current working directory. (You should only define it if the audios you wish to analyze are not already read in R. Otherwise define the audioList parameter).
audioList	An optional list of Wave objects to analyze.
filter	An optional character vector indicating IDs, Conditions, Dimensions, or other patterns used to filter for specific audio files. Default corresponds to NA.
fileType	A character string indicating the audio file format (wav or mp3). Default corresponds to wav.
fileNamePattern	A character string indicating the naming format of the audio files, such as "ID-Condition-Dimension", "Condition_ID_Dimension" or just "ID". Default corresponds to "ID".
sep	A non alpha-numeric character that acts as separator between the different naming components. Default corresponds to an underscore. This field can be ignored if the audio file name only contains an ID component.
parallel	Logical value indicating whether to use parallelism to extract the different audio characteristics to enhance computational performance. Default corresponds to FALSE.
recursive	Logical value indicating whether subdirectories in the specified directory should be included when searching for voice files. Default corresponds to FALSE.
preprocess	Logical value indicating whether to preprocess (normalize amplitude and remove background noise) the audio files before extraction and analysis. Default corresponds to FALSE.
extended	Logical value indicating whether all features extracted by the soundgen package should be inputted. Default corresponds to FALSE.
...	Other options used to control preprocessing behavior.

Details

The voiceR package requires the audio file names to follow a specific pattern, in which the different components are separated by a non alphanumeric character (e.g., "_"). File name components refer to:

ID Unique identifier of the speaker or recording.

Condition Experimental condition or other grouping variable.

Dimension Additional survey or experiment information (e.g., additional conditions).

Order and presence of the different components is not important, as long as at least one of the aforementioned components is present. Furthermore, non-relevant components can be skipped by specifying "Null" in its position such as: ID_Null_Condition. Valid name patterns are, for example, 876h Interior (ID Condition), Exterior-3543h (Condition-ID), 983b-Exterior-q1 (ID-Condition-Dimension) or 455k (ID). All voice files within one session need to follow the same file naming pattern. Note: the non-alpha numeric separator should also be specified as sep.

Value

A data.frame is created with the following audio features:

duration Total duration in seconds.

voice_breaks_percent Proportion of unvoiced frames.

RMS_env Root mean square of the amplitude envelope.

mean_loudness Average subjective loudness in sone.

mean_F0 Average fundamental frequency in Hertz.

sd_F0 Standard deviation of the fundamental frequency in Hertz.

mean_entropy Average Wiener entropy. A value of 0 indicates a pure tone, while a value of 1 indicates white noise.

mean_HNR Average Harmonics-to-Noise Ratio.

ID ID component of the audio file.

Condition If fileNamePattern and audio names include a Condition, an additional column with the Condition component of the audio file is included.

Dimension If fileNamePattern and audio names include a Dimension, an additional column with the Dimension component of the audio file is included.

See Also

[soundgen::analyze()], [seewave::duration()], [seewave::rms()], [seewave::env()]

Examples

```
audioData <- autoExtract(audioList = testAudioList, filter = c("5b438f516066ad470d3be72c52005251"))
```

`autoReport` *Generate a Summary Report*

Description

Generates a summary report containing the output of autoExtract(), normalityPlots() and comparisonPlots().

Usage

```
autoReport(
  audioData,
  savePath = getwd(),
  includeDimensions = FALSE,
  avoidNormalCheck = c(),
  filename = "voiceR_report.html"
)
```

Arguments

audioData	A data.frame generated by autoExtract() function.
savePath	Character string indicating the full path to the folder to which we want to save the generated report. By default it is set to the current working directory.
includeDimensions	Logical value indicating whether Dimensions should be also included as a factor or not. Default corresponds to FALSE.
avoidNormalCheck	Logical vector, indicating if and what variables' distribution were transformed to normal. By default it is set to FALSE for each of the measures. Alternatively, you can set it to TRUE to avoid checking normality for all the measures.
filename	Optional character string indicating the file name of the generated report. Default corresponds to "voiceR_report.html".

Value

html report file, which is saved in the selected path, but returns nothing.

Examples

```
autoReport(testAudioData = testAudioData)
```

---

comparisonPlots *Create boxplots for extracted audio features*

Description

Generates boxplots for each exracted feature. Plots can be split by experimental condition.

Usage

```
comparisonPlots(
  audioData,
  by = c(),
 measures = c("duration", "voice_breaks_percent", "RMS_env", "mean_loudness", "mean_F0",
    "sd_F0", "mean_entropy", "mean_HNR"),
  normalSig = 0.05,
  avoidNormalCheck = FALSE
)
```

Arguments

audioData	A data.frame generated by the autoExtract() function.
by	An optional character vector indicating column(s) from which the comparison groups are to be retrieved.
measures	An optional character vector indicating the name of the variables to be plotted.
normalSig	Set significance level to test for normality assumptions. Default corresponds to "0.05".
avoidNormalCheck	Logical value forcing to avoid checking for normality. When defined as TRUE, it is assumed that the data is normally distributed. Default corresponds to FALSE.

Value

A list containing the generated boxplots.

Examples

```
comparisonPlots(testAudioData, by = "Condition")
```

---

createEmptyDF *Create an Empty data.frame*

Description

Internal function which creates an empty data.frame in which the different audio files represent rows and the extracted measures represent columns. Several options can be configured such as joining dimensions or separating conditions.

Usage

```
createEmptyDF(
  path = ".",
  audioList = list(),
  fileType = "wav",
  fileNamePattern = "ID_Condition_Dimension",
  sep = "_",
  measures = c(),
  jointDimensions = FALSE,
  separateConditions = TRUE,
  filter = NA,
  recursive = FALSE
)
```

Arguments

path	An optional character string indicating the path to the folder containing the audio files. Default corresponds to current working directory.
audioList	Optional list with already loaded Wave objects to analyze.
fileType	Character string indicating the file format (wav or mp3) of the audio files. Default corresponds to wav.
fileNamePattern	A character string indicating the naming format, such as "ID-Condition-Dimension", "Condition_ID_Dimension" or "ID". Default corresponds to "ID_Condition_Dimension".
sep	A non alpha-numeric character that acts as separator between the different naming components. Default corresponds to underscore. This field can be ignored if the audio file names only contain an ID component.
measures	A character vector of measures that should appear in the data frame columns.
jointDimensions	Logical value indicating whether dimensions should be joint into a single or not. Default corresponds to FALSE.
separateConditions	Logical value indicating whether conditions should be separated or not. Default corresponds to TRUE.
filter	Optional character vector indicating IDs, Conditions, Dimensions or other name patterns. Default corresponds to NA.
recursive	Logical value indicating whether subdirectories should be included when searching for audio files. Default corresponds to FALSE.

Value

An empty data.frame in which ID's represent rows and dimensions/measures represent columns.

getComponents *Get Components Name and Positions*

Description

Indicates the presence and order in which components are retrieved from the file name of each recording.

Usage

```
getComponents(fileNames, fileNamePattern = "ID_Condition_Dimension", sep = "_")
```

Arguments

fileNames	A character vector of audio file names.
fileNamePattern	A character string indicating the naming format of the audio files, such as "ID-Condition-Dimension", "Condition_ID_Dimension" or "ID". Default corresponds to "ID_Condition_Dimension".
sep	A non alpha-numeric that acts as separator between the different naming components. Default corresponds to underscore.

Value

A list, containing a vector of positions for each component and a data.frame containing the values for each component of the audio files.

Examples

```
getComponents(names(testAudioList), fileNamePattern = "ID_Condition_", sep = "_")
```

getConditions *Get Conditions*

Description

Retrieves the experimental conditions from the file name following a naming pattern in which the various components (IDs, conditions, and dimensions) are separated by a non-alphanumeric character.

Usage

```
getConditions(
  path = ".",
  audioList = list(),
  fileType = "wav",
  fileNamePattern = "ID_Condition_Dimension",
  sep = "_",
  filter = NULL,
  recursive = FALSE
)
```

Arguments

path	A character string indicating the path to the folder containing the audio files. Default corresponds to the current working directory.
audioList	Optional list with Wave objects to analyze.
fileType	Character string indicating the file format (wav or mp3) of the audio files. Default corresponds to wav.
fileNamePattern	Character string indicating the naming format of the audio files, such as "ID-Condition-Dimension", "Condition_ID_Dimension" or "ID". Default corresponds to "ID_Condition_Dimension".
sep	A non alpha-numeric that acts as separator between the different naming components. Default corresponds to underscore.
filter	Optional character vector used to filter for specific audio files. Default corresponds to NULL.
recursive	A logical value indicating whether subdirectories should be included when searching for voice files. Default corresponds to FALSE.

Value

Character vector, which contains all the unique conditions of the voice files extracted from the name pattern of the audio files.

Examples

```
getConditions(audioList = testAudioList,
fileNamePattern = "ID_Condition_Dimension", sep = "_")
```

`getDimensions` *Get Dimensions*

Description

Retrieves the unique dimensions from the file name of multiple audio files following a naming pattern in which the various components (IDs, dimensions and, conditions) are separated by a non-alphanumeric character.

Usage

```
getDimensions(
  path = ".",
  audioList = list(),
  fileType = "wav",
  fileNamePattern = "ID_Condition_Dimension",
  sep = "_",
  filter = NULL,
  recursive = FALSE
)
```

Arguments

`path`	A character string indicating the path to the folder containing the audio files. Default corresponds to the current working directory.
`audioList`	Optional list with Wave objects to analyze.
`fileType`	Character string indicating the file format (wav or mp3) of the audio files. Default corresponds to wav.
`fileNamePattern`	cCharacter string indicating the naming format of the audio files, such as "ID-Condition-Dimension", "Condition_ID_Dimension" or "ID". Default corresponds to "ID_Condition_Dimension".
`sep`	A non alpha-numeric that acts as separator between the different naming components. Default corresponds to underscore.
`filter`	Optional character vector to filter for specific audio files. Default corresponds to NULL.
`recursive`	A logical value indicating whether subdirectories should be included when searching for voice files. Default corresponds to FALSE.

Value

Character vector, which contains all the unique dimensions of the voice files found in the specified directory.

Examples

```
getDimensions(audioList = testAudioList,
fileNamePattern = "ID_Condition_Dimension", sep = "_")
```

---

getIds *Get IDs*

Description

Retrieves the unique IDs from the file name of multiple audio files following a naming pattern in which the various components (IDs, dimensions, conditions) are separated by a non-alphanumeric character.

Usage

```
getIds(
  path = ".",
  audioList = NULL,
  fileType = "wav",
  fileNamePattern = "ID_Condition_Dimension",
  sep = "_",
  filter = NULL,
  recursive = FALSE
)
```

Arguments

path	A character string indicating the path to the folder containing the audio files. Default corresponds to the current working directory.
audioList	Optional list with Wave objects to analyze.
fileType	Character string indicating the file format (wav or mp3) of the audio files. Default corresponds to wav.
fileNamePattern	Character string indicating the naming format of the audio files, such as "ID-Condition-Dimension", "Condition_ID_Dimension" or "ID". Default corresponds to "ID_Condition_Dimension".
sep	A non alpha-numeric that acts as separator between the different naming components. Default corresponds to underscore.
filter	Optional character vector used to filter for specific audio files. Default corresponds to NULL.
recursive	A logical value indicating whether subdirectories should be included when searching for voice files. Default corresponds to FALSE.

Value

Character vector, which contains all the unique IDs extracted from the name pattern of the audio files.

Examples

```
getIds(audioList = testAudioList,
fileNamePattern = "ID_Condition_Dimension", sep = "_")
```

`MissDimPerId` *What IDs have missing dimensions?*

Description

Indicates whether and which dimensions are missing for each ID.

Usage

```
MissDimPerId(
  path = ".",
  audioList = NULL,
  ids = c(),
  fileType = "wav",
  fileNamePattern = "ID_Condition_Dimension",
  sep = "_",
  recursive = FALSE
)
```

Arguments

`path`	Character string indicating the path to the folder containing the audio files. Default corresponds to the current working directory.
`audioList`	Optional list with Wave objects to analyze.
`ids`	Character vector indicating the IDs of the files.
`fileType`	Character string indicating the file format (wav or mp3) of the audio files. Default corresponds to wav.
`fileNamePattern`	Character string indicating the naming format of the audio files, such as "ID-Condition-Dimension", "Condition_ID_Dimension" or "ID". Default corresponds to "ID_Condition_Dimension".
`sep`	A non alpha-numeric that acts as separator between the different naming components. Default corresponds to underscore.
`recursive`	A logical value indicating whether subdirectories should be included when searching for voice files. Default corresponds to FALSE.

Value

A data.frame, in which rows represent IDs and columns represent missing vs. present Dimensions.

Examples

```
MissDimPerId(audioList = testAudioList)
```

`normalityPlots` *Normality Plots*

Description

Generates plots showing the normality of the different measures from the data.frame obtained from autoExtract.

Usage

```
normalityPlots(
  audioData,
 measures = c("duration", "voice_breaks_percent", "RMS_env", "mean_loudness", "mean_F0",
    "sd_F0", "mean_entropy", "mean_HNR")
)
```

Arguments

`audioData`	A data.frame generated by autoExtract.
`measures`	An optional character vector indicating the name of the variables to be plotted.

Value

A list containing the different plots that are generated.

Examples

```
normalityPlots(testAudioData)
```

`normalizeData` *Normalize audio data using Box-Cox transformation*

Description

This function normalizes audio data using the Box-Cox transformation. It takes in a data frame of audio data and a vector of measures to be normalized. Users can choose to normalize by dimensions and/or conditions.

Usage

```
normalizeData(
  audioData,
 measures = c("duration", "voice_breaks_percent", "RMS_env", "mean_loudness", "mean_F0",
    "sd_F0", "mean_entropy", "mean_HNR"),
  includeDimensions = FALSE,
  includeConditions = FALSE
)
```

Arguments

`audioData`	A data.frame generated by the autoExtract() function.
`measures`	A vector of strings specifying the measures to be normalized. Default corresponds to all the measures extracted by autoExtract().
`includeDimensions`	A logical value indicating whether or not to include dimensions in the normalization process. Default corresponds to FALSE.
`includeConditions`	A logical value indicating whether or not to include conditions in the normalization process. Default corresponds to FALSE.

Value

A list containing three elements: (1) a data frame of the normalized audio data and (2) a logical vector indicating whether or not each measure was transformed using Box-Cox transformation and (3) the Box-cox constant added to each measure.

Examples

```
normalizeData(testAudioData)
```

preprocess *Preprocess list of Audio objects*

Description

Automatically preprocesses a list of Wave objects by normalizing their amplitude and removing background noise.

Usage

```
preprocess(audioList, normalizeAmplitude = TRUE, removeNoise = TRUE, ...)
```

Arguments

audioList	A list of Wave objects.
normalizeAmplitude	A logical value indicating whether to normalize amplitude.
removeNoise	A logical value indicating whether to remove background noise.
...	Other options used to control preprocessing behavior.

Value

A list of (processed) Wave objects.

Examples

```
preprocess(testAudioList)
```

readAudio *Read Audio Files*

Description

Loads all audio files in the specified directory and provides the option to filter via ID, conditions, and/or dimensions.

Usage

```
readAudio(path = ".", filter = c(), fileType = "wav", recursive = FALSE)
```

Arguments

path	Character string indicating the full path to the folder containing the audio files. Default corresponds to the current working directory.
filter	Optional character vector, containing patterns, such as IDs or conditions of each audio file.
fileType	Character string indicating the file format (wav or mp3) of the audio files. Default corresponds to wav.
recursive	A logical value indicating whether subdirectories should be included when searching for voice files. Default corresponds to FALSE.

Value

Returns a list of Wave objects.

Examples

```
readAudio(system.file("Audios", package = "voiceR"),
fileType = "wav", recursive = TRUE)
```

---

saveAudio *Save Audio Files*

Description

Save all objects in a list of Wave objects as a .wav or .mp3 file in the specified path.

Usage

```
saveAudio(audioList, path = "./Preprocessed/", fileType = "wav")
```

Arguments

audioList	A list of Wave objects.
path	A character string indicating the full path to the folder containing the audio files. Default corresponds to the current working directory.
fileType	Character string indicating the file format (wav or mp3) of the audio files. Default corresponds to wav.

Value

Save objects of a Wave list as .mp3 or .wav files.

Examples

```
saveAudio(testAudioList, fileType = "wav")
```

tableANOVA *Create a Table of ANOVA results*

Description

Automatically generates HTML table with results for one-way or two-way ANOVAs.

Usage

```
tableANOVA(
  audioData,
  by = c(),
  measure = "duration",
  nameMeasure = c(),
  figureNumber = 1
)
```

Arguments

audioData	A data.frame generated by the autoExtract() function.
by	A character vector indicating the name of the factor(s).
measure	Name of the dependent variable.
nameMeasure	Optional relabelling of dependent variable for the output table. If no value is provided, the original variable name is used.
figureNumber	An integer indicating the figure number to create the title for the table. Default corresponds to 1.

Value

HTML table showing the ANOVA results in APA formatting style.

Examples

```
tableANOVA(testAudioData, by = c("Condition", "Dimension"), measure = "duration")
```

`tableDunn` *Create a Table for the results of a Dunn's test*

Description

Automatically generates HTML table with results for Dunn's test.

Usage

```
tableDunn(
  audioData,
  by = c(),
  measure = "duration",
  nameMeasure = c(),
  figureNumber = 1
)
```

Arguments

`audioData`	A data.frame generated by the autoExtract() function.
`by`	A character vector indicating the name of the factor(s).
`measure`	Name of the dependent variable.
`nameMeasure`	Optional string to rename dependent variable in the output table. If no value is provided, then original variable name is displayed.
`figureNumber`	Integer indicating the figure number, used to create the title for the table. Default corresponds to 1.

Value

HTML table showing Dunn's test results in APA formatting style.

Examples

```
tableDunn(testAudioData, by = "Condition", measure = "duration")
```

`tableKruskal` *Create a table for Kruskal-Wallis test results*

Description

Automatically generates HTML table with the results of a Kruskal-Wallis test.

Usage

```
tableKruskal(
  audioData,
  by = c(),
  measure = "duration",
  nameMeasure = c(),
  figureNumber = 1,
  InfoTable = FALSE
)
```

Arguments

`audioData`	A data.frame generated by the autoExtract() function.
`by`	A character vector indicating the name of the factor(s).
`measure`	Name of the dependent variable.
`nameMeasure`	Optional string to rename dependent variable in the output table. If no value is provided, original variable name is displayed.
`figureNumber`	Integer indicating the figure number, used to create the title for the table. Default corresponds to 1.
`InfoTable`	Logical value indicating the type of table to be reported. If FALSE, a table containing the mean ranks for each group is displayed; if TRUE, a table containing the main results for the Kruskal-Wallis test is displayed. Default corresponds to FALSE.

Value

HTML table showing Kruskal-Wallis test results in APA formatting style.

Examples

```
tableKruskal(testAudioData, by = "Condition", measure = "duration")
```

`tableNormality` *Create a table for Shapiro Wilk results*

Description

This function returns a data.frame containing Shapiro Wilk results by conditions (and dimensions if "includeDimensions" is set to TRUE). Likewise, if "HTMLTable" is set to TRUE, it outputs results as an HTML table in APA style.

Usage

```
tableNormality(
  audioData,
  measure = "duration",
  includeDimensions = FALSE,
  figureNumber = 1,
  nameMeasure = c(),
  HTMLTable = FALSE
)
```

Arguments

`audioData`	A data.frame generated by the autoExtract() function.
`measure`	Name of the dependent variable. Default corresponds to duration.
`includeDimensions`	Logical value indicating if different dimensions should be also included as a factor when testing for normality. Default corresponds to FALSE.
`figureNumber`	Integer indicating the figure number, used to create the title for the table. Default corresponds to 1.
`nameMeasure`	Optional string indicating the name to be displayed for the dependent variable in the output table. If no value is provided, the string used for the measure attribute is displayed.
`HTMLTable`	Logical value indicating if an HTML table should be generated. Default corresponds to FALSE.

Value

If "HTMLTable" is set to FALSE, this function returns a data.frame with Shapiro-Wilk test results for each condition (if condition column exists) and dimension (if dimension column exists and "includeDimensions" is set to TRUE). Otherwise an HTML table showing test results in APA formatting style is created.

Examples

```
tableNormality(testAudioData, measure = "duration")
```

`tableSchreirer` *Create a table for the Scheirer-Ray-Hare test results*

Description

Automatically generates an HTML table with the results for Scheirer-Ray-Hare test.

Usage

```
tableSchreirer(
  audioData,
  by = c(),
  measure = "duration",
  nameMeasure = c(),
  figureNumber = 1
)
```

Arguments

`audioData`	A data.frame generated by the autoExtract() function.
`by`	A character vector indicating the name of the factor(s).
`measure`	Name of the dependent variable.
`nameMeasure`	Optional string to rename the dependent variable in the output table. If no value is provided, the original variable name is displayed.
`figureNumber`	Integer indicating the figure number, used to create the title for the table. Default corresponds to 1.

Value

HTML table showing Scheirer-Ray-Hare test results in APA formatting style.

Examples

```
tableSchreirer(testAudioData, by = c("Condition", "Dimension"), measure = "duration")
```

`tableSimpleMainEffects` *Create a table for simple main effects analysis*

Description

Automatically generates an HTML table with the results of a simple main effects analysis.

Usage

```
tableSimpleMainEffects(
  audioData,
  by = c(),
  measure = "duration",
  nameMeasure = c(),
  figureNumber = 1
)
```

Arguments

`audioData`	A data.frame generated by the autoExtract() function.
`by`	A character vector indicating the name of the factors. Note: it requires two factors.
`measure`	Name of the dependent variable.
`nameMeasure`	Optional string to rename the dependent variable in the output table. If no value is provided, the original variable name is displayed.
`figureNumber`	Integer indicating the figure number, used to create the title for the table. Default corresponds to 1.

Value

HTML table showing simple main effects analysis results in APA formatting style.

Examples

```
tableSimpleMainEffects(testAudioData, by = c("Condition", "Dimension"), measure = "duration")
```

tableSummary *Create a summary table*

Description

Automatically generates HTML table with the main descriptive statistics (mean, standard deviation and number of data points) for each variable of the data set. Descriptive statistics can be conditioned by groups if grouping variables are defined.

Usage

```
tableSummary(
  audioData,
  by = c(),
 measures = c("duration", "voice_breaks_percent", "RMS_env", "mean_loudness", "mean_F0",
    "sd_F0", "mean_entropy", "mean_HNR"),
  nameMeasures = c(),
  figureNumber = 1
)
```

Arguments

audioData	A data.frame generated by the autoExtract() function.
by	A character vector indicating the name of the factor(s).
measures	A character vector indicating the name of the variables to be included in the table.
nameMeasures	Optional character vector indicating the names to be displayed for each of the different measures in the output table. If no value is provided, original variable names are displayed.
figureNumber	Integer indicating figure number, used to create the title for the table. Default corresponds to 1.

Value

HTML file summarizing the main descriptive statistics for each variable, conditioned by group(s) if one or more factors are provided.

Examples

```
tableSummary(testAudioData, by = c("Condition", "Dimension"),
measures = c("duration", "voice_breaks_percent", "RMS_env", "mean_loudness",
"mean_F0"))
```

`tableTukey` *Create a table for Tukey HSD test results*

Description

Automatically generates an HTML table with the results of a Tukey HSD test.

Usage

```
tableTukey(
  audioData,
  by = c(),
  measure = "duration",
  nameMeasure = c(),
  figureNumber = 1
)
```

Arguments

`audioData`	A data.frame generated by the autoExtract() function.
`by`	A character vector indicating the name of the factor(s).
`measure`	Name of the dependent variable.
`nameMeasure`	Optional string to rename the dependent variable in the output table. If no value is provided, the original variable name is displayed.
`figureNumber`	Integer indicating the figure number, used to create the title for the table. Default corresponds to 1.

Value

HTML table showing Tukey HSD test results in APA formatting style.

Examples

```
tableTukey(testAudioData, by = "Condition", measure = "duration")
```

testAudioData *voiceR test Audio Data*

Description

A test audio features data.frame, obtained by using autoExtract() on the extended version of testAudioList, found here.

Format

'testAudioData' A data.frame containing 90 observations and 11 variables, which is the result of applying the autoExtract() function to the extended version of the data found on testAudioList. This data.frame contains several voice features for 90 audio files, which correspond to 15 English-speaking participants. Participants first completed a Baseline Voice Task in which they were instructed to read two predefined phrases ((1) Bar: "I go to the bar", (2) Beer: "I drink a beer") aloud in their normal (neutral) voice. Participants were then instructed to read the predefined phrases in either a happy or a sad voice. The experimenter requested each emotion one at a time and in a random sequence to counter-order effects. Participants were then asked to describe their experience of mimicking the stated phrases for each of the specified emotional states. Thus the data contains 6 observations per participant: two observations for the neutral state, two for the happy simulated state, and two for the sad simulated state. Below we also provide information about the columns this data.frame contains:

ID Participant identifier

Condition refers to the intention or emotional aspect that the speaker is conveying: Happy, or Sad. This component makes reference to the main point that we want to compare in our data; in the voiceR package the main comparison component is called Condition, because it usually refers to experimental conditions.

Dimensions Phrase participants read: (1) Bar: "I go to the bar"; (2): Beer: "I drink a beer"

duration Total duration in seconds.

voice_breaks_percent Proportion of unvoiced frames.

RMS_env Root mean square of the amplitude envelope.

mean_loudness Average subjective loudness in sone.

mean_F0 Average fundamental frequency in Hertz.

sd_F0 Standard deviation of the fundamental frequency in Hertz.

mean_entropy Average Wiener entropy. A value of 0 indicates a pure tone, while a value of 1 indicates white noise.

mean_HNR Average Harmonics-to-Noise Ratio.

Examples

```
data(testAudioData)
```

testAudioList *voiceR test Audio List*

Description

An audio list containing recordings for ten English-speaking participants. Interested users can download an extended version of this data set <a href="https://osf.io/zt5h2/?view_only=348d1d172435449391e8d645477164
Participants were seated with an experimenter in a sound-isolated room. All voice recordings were collected using a Blue Yeti Microphone. We used Audacity version 2.4.2 to record the audio files and saved all files using a 32-bit WAV format. Participants were instructed to read the phrase "I go to the bar" in either a happy or a sad voice. The experimenter requested each emotion one at a time and in random sequence to counter order effects.

Format

'testAudioList' A list containing 20 Wave objects. Each Wave object has as name its file name. This file name contains three components separated by an underscore:

First component combination of numbers and letters refers to the participant identifier, i.e., the ID component

Second component refers to the intention or emotional aspect that the speaker is conveying: Happy, or Sad. This component makes reference to the main point that we want to compare in our data; in the voiceR package the main comparison component is called Condition, because it usually refers to experimental conditions.

Third component provides additional information in the voiceR package.

Examples

```
data(testAudioList)
```

voiceRApp *Start voiceR Shiny App*

Description

Launches the voiceR Shiny app, providing the opportunity to analyze multiple audio files via a graphical user interface (no-code interface).

Usage

```
voiceRApp()
```

Value

This function launches the voiceR Shiny app.

Examples

```
if(interactive()){
   voiceRApp()
}
```

Index