Package 'voice'

January 5, 2021

Type Package

Title General Tools for Voice Analysis, Speaker Recognition and Mood Inference

Version 0.0.0.9000

Date 2020-12-28

Author c(
person(``Zabala", ``Filipe J.", email = ``filipe.zabala@gmail.com", role = c(``cre",``aut")),
person(``Salum Jr.", ``Giovanni A.", email = ``gsalumjr@gmail.com", role = ``aut"))

Maintainer Filipe J. Zabala <filipe.zabala@gmail.com>

URL https://github.com/filipezabala/voice

BugReports https://github.com/filipezabala/voice/issues

Description General purpose tools for voice analysis, speaker recognition and mood inference. Gathers R and Python tools to solve problems concerning voice.

Depends R (>= 4.0.0), utf8

Imports R.utils, parallelSVM, reticulate, seewave, tidyr, wrassp, ellipse, ggplot2, ggfortify, RColorBrewer, dplyr, e1071, tibble, tuneR

License GPL-3 + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

R topics documented:

`class_svm` *Fits, forecasts and gets performance from SVM models, serial and parallelized.*

Description

Fits, forecasts and gets performance from SVM models, serial and parallelized.

Usage

```
class_svm(
  x,
  model,
  filtro = c("id"),
  percTreino = 0.5,
  setSeed = 1,
  custo = 1,
  gama = 1,
  paralelo = TRUE,
  print.cm = TRUE
)
```

Arguments

`x`	A data frame or tibble.
`model`	Character containing the model structure. See `expand_model`.
`filtro`	Column(s) to filter the samples.
`percTreino`	Percentage of the database used to train the model, filtered by `filtro`.
`setSeed`	Specified seed for the pseudo-random parts.
`custo`	Cost of constraints violation (default: 1)—it is the 'C'-constant of the regularization term in the Lagrange formulation.
`gama`	Needed for all kernels except linear (default: 1/(data dimension))
`paralelo`	Logical. Should the process be parallelized?
`print.cm`	Logical. Should the confusion matrix be shown?

Value

`fcast` predicted time series using the model that minimizes the forecasting mean square error.

`runtime` running time.

`mse.pred` mean squared error of prediction. Used to decide the best model.

References

Vapnik, Vladimir (2000). The Nature of Statistical Learning Theory, Springer.

Examples

```
library(voice)
```

conv *Convolute vectors.*

Description

Convolute vectors.

Usage

```
conv(
  y,
  compact.to,
  drop.zeros = FALSE,
  to.data.frame = FALSE,
  round.off = NULL
)
```

Arguments

y	A vector or time series.
compact.to	Proportion of remaining points after compactation, between (including) 0 and 1. If equals to 1 and keep.zeros = T, the original vector is presented.
drop.zeros	Logical. Drop repeated zeros? Default: 'FALSE'.
to.data.frame	Logical. Convert to data frame? Default: 'FALSE'.
round.off	Number of decimal places of the convoluted vector. Default: 'NULL'.

Value

A list of convoluted x and y values with length near to compact.to*length(y).

See Also

rm0, conv_mc, conv_df

Examples

```
(c1 <- conv(1:100, compact.to = 0.2, drop.zeros = TRUE))
length(c1$y)
plot(1:100, type = 'l')
points(c1$x, c1$y, col='red')

(v2 <- c(1:5, rep(0,10), 1:10, rep(0,5), 10:20, rep(0,10)))
length(v2)
conv(v2, 0.1, drop.zeros = TRUE, to.data.frame = FALSE)
conv(v2, 0.1, drop.zeros = TRUE, to.data.frame = TRUE)
conv(v2, 0.2, drop.zeros = TRUE)
conv(v2, 0.2, drop.zeros = FALSE)

(v3 <- c(rep(0,10), 1:20, rep(0,3)))
(c3 <- conv(v3, 1/3, drop.zeros = FALSE, to.data.frame = FALSE))
lapply(c3, length)
plot(v3, type = 'l')
```

```
points(c3$x, c3$y, col = 'red')

(v4 <- c(rnorm(1:100)))
(c4 <- conv(v4, 1/4, round.off = 3))
```

---

conv_df *Convolute data frames.*

Description

Convolute data frames.

Usage

```
conv_df(
  x,
  compact.to,
  colnum = NULL,
  id = "id",
  by.filter = id,
  drop.x = FALSE,
  drop.zeros = FALSE,
  to.data.frame = TRUE,
  round.off = NULL,
  mc.cores = parallel::detectCores()
)
```

Arguments

x	A data frame.
compact.to	Percentage of remaining points after compactation. If equals to 1 and keep.zeros = T, the original vector is presented.
colnum	A char vector indicating the numeric colnames. If NULL, uses the columns of the numeric class.
id	The identification column. Default: 'id'.
by.filter	A char indicating the column to filter by. If NULL uses the id content.
drop.x	Logical. Drop columns containing .x? Default: 'FALSE'.
drop.zeros	Logical. Drop repeated zeros or compress to 1 zero per null set? Default: 'FALSE'.
to.data.frame	Logical. Should the return be a data frame? If F returns a list. Default: 'TRUE'.
round.off	Number of decimal places of the convoluted vector. Default: 'NULL'.
mc.cores	The number of cores to mclapply. By default uses parallel::detectCores().

Value

A vector of convoluted values with length near to compact.to*length(x).

See Also

conv, conv_mc

Examples

```
library(voice)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)

# getting all the 1092 features
ef <- extract_features(dirname(path2wav), features = c('f0','formants',
'zcr','mhs','rms','gain','rfc','ac','cep','dft','css','lps','mfcc'),
mc.cores = 1)

## Not run:
(cef.df <- conv_df(ef, 0.1, id = 'file_name', mc.cores = 1))
(cef.df2 <- conv_df(ef, 0.1, id = 'file_name', drop.x = TRUE, mc.cores = 1))

dim(ef)
dim(cef.df)
dim(cef.df2)
(cef.list <- conv_df(ef, 0.1, id = 'file_name', to.data.frame = FALSE, mc.cores = 1))

## End(Not run)
```

---

conv_mc *Convolute vectors using multicore.*

Description

Convolute vectors using multicore.

Usage

```
conv_mc(
  y,
  compact.to,
  drop.zeros = FALSE,
  to.data.frame = FALSE,
  round.off = NULL,
  mc.cores = parallel::detectCores()
)
```

Arguments

y	A numeric vector, matrix or data frame.
compact.to	Percentage of remaining points after compression. If equals to 1 and keep.zeros = T, the original vector is presented.
drop.zeros	Logical. Drop repeated zeros? Default: 'FALSE'.
to.data.frame	Logical. Convert to data frame? Default: 'FALSE'.
round.off	Number of decimal places of the convoluted vector. Default: 'NULL'.
mc.cores	The number of cores to mclapply. Default: 'parallel::detectCores()'.

Value

A list of x and y convoluted values with length near to `compact.to*length(y)`.

See Also

`rm0`, `conv`, `conv_df`

Examples

```
library(voice)
# Same result of conv() function if x is a vector
conv(1:100, compact.to = 0.1, drop.zeros = TRUE, to.data.frame = FALSE)
conv_mc(1:100, compact.to = 0.1, drop.zeros = TRUE, to.data.frame = FALSE)

conv(1:100, compact.to = 0.1, drop.zeros = TRUE, to.data.frame = TRUE)
conv_mc(1:100, compact.to = 0.1, drop.zeros = TRUE, to.data.frame = TRUE)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)

# getting all the 1092 features
ef <- extract_features(dirname(path2wav), features = c('f0','formants',
'zcr','mhs','rms','gain','rfc','ac','cep','dft','css','lps','mfcc'),
mc.cores = 1)

ef.num <- ef[-1]
nrow(ef.num)
cm1 <- conv_mc(ef.num, compact.to = 0.1, drop.zeros = TRUE,
to.data.frame = FALSE, mc.cores = 1)
names(cm1)
lapply(cm1$f0, length)
```

`expand_model` *Expand model given* `y` *and* `x` *variables.*

Description

Expand model given `y` and `x` variables.

Usage

```
expand_model(y, x, k)
```

Arguments

`y`	The Y variable.
`x`	The X variables.
`k`	Number of additive components.

Value

A `char` vector containing the expanded models.

Examples

```
expand_model('y', LETTERS[1:4], 1)
expand_model('y', LETTERS[1:4], 2)
expand_model('y', LETTERS[1:4], 3)
expand_model('y', LETTERS[1:4], 4)
```

`extract_features` *Extracts features from WAV audio files.*

Description

Extracts features from WAV audio files.

Usage

```
extract_features(
  directory,
  filesRange = NULL,
 features = c("f0", "formants", "zcr", "mhs", "rms", "gain", "rfc", "ac", "mfcc"),
  gender = "u",
  windowShift = 5,
  numFormants = 8,
  numcep = 12,
  dcttype = c("t2", "t1", "t3", "t4"),
  fbtype = c("mel", "htkmel", "fcmel", "bark"),
  resolution = 40,
  usecmp = FALSE,
  mc.cores = parallel::detectCores(),
  full.names = TRUE,
  recursive = FALSE,
  check.mono = TRUE,
  stereo2mono = FALSE,
  overwrite = FALSE,
  freq = 44100
)
```

Arguments

`directory`	A directory containing audio file(s) in WAV format. If more than one directory is provided, only the first one is used.
`filesRange`	The desired range of directory files (default: `NULL`, i.e., all files).
`features`	Vector of features to be extracted. (default: 'f0','formants','zcr','mhs','rms','gain','rfc','ac','mfcc'). The following four features contain 4*257 = 1028 columns (257 each): `'cep'`, `'dft'`, `'css'` and `'lps'`.
`gender`	= <code> set gender specific parameters where <code> = `'f'`[emale], `'m'`[ale] or `'u'`[nknown] (default: `'u'`). Used by `wrassp::ksvF0`, `wrassp::forest` and `wrassp::mhsF0`.
`windowShift`	= <dur> set analysis window shift to <dur>ation in ms (default: 5.0). Used by `wrassp::ksvF0`, `wrassp::forest`, `wrassp::mhsF0`, `wrassp::zcrana`, `wrassp::rfcana`, `wrassp::acfana`, `wrassp::cepstrum`, `wrassp::dftSpectrum`, `wrassp::cssSpectrum` and `wrassp::lpsSpectrum`.

numFormants	= <num> <num>ber of formants (default: 8). Used by wrassp::forest.
numcep	Number of Mel-frequency cepstral coefficients (cepstra) to return (default: 12). Used by tuneR::melfcc.
dcttype	Type of DCT used. 't1' or 't2', 't3' for HTK 't4' for feacalc (default = 't2'). Used by tuneR::melfcc.
fbtype	Auditory frequency scale to use: 'mel', 'bark', 'htkmel', 'fcmel' (default: 'mel'). Used by tuneR::melfcc.
resolution	= <freq> set FFT length to the smallest value which results in a frequency resolution of <freq> Hz or better (default: 40.0). Used by wrassp::cssSpectrum, wrassp::dftSpectrum and wrassp::lpsSpectrum.
usecmp	Logical. Apply equal-loudness weighting and cube-root compression (PLP instead of LPC) (default: FALSE). Used by tuneR::melfcc.
mc.cores	Number of cores to be used in parallel processing. (default: parallel::detectCores())
full.names	Logical. If TRUE, the directory path is prepended to the file names to give a relative file path. If FALSE, the file names (rather than paths) are returned. (default: TRUE) Used by base::list.files.
recursive	Logical. Should the listing recursively into directories? (default: FALSE) Used by base::list.files.
check.mono	Logical. Check if the WAV file is mono. (default: TRUE)
stereo2mono	Logical. Should files be converted from stereo to mono? (default: FALSE)
overwrite	Logical. Should converted files be overwritten? If not, the file gets the suffix _mono. (default: FALSE)
freq	Frequency in Hz to write the converted files when stereo2mono=TRUE. (default: 44100)

Examples

```
library(voice)
library(RColorBrewer)
library(ellipse)
library(ggplot2)
library(grDevices)
library(ggfortify)

# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)

# getting all the 1092 features
ef <- extract_features(dirname(path2wav), features = c('f0','formants',
'zcr','mhs','rms','gain','rfc','ac','cep','dft','css','lps','mfcc'),
mc.cores = 1)
dim(ef)
ef

# using the default, i.e., not using 'cep','dft','css' and 'lps' (4*257 = 1028 columns)
ef2 <- extract_features(dirname(path2wav), mc.cores = 1)
dim(ef2)
ef2
table(ef2$file_name)
```

```
# limiting filesRange
ef3 <- extract_features(dirname(path2wav), filesRange = 3:6, mc.cores = 1)
dim(ef3)
ef3
table(ef3$file_name)

# calculating correlation of ef2
data <- cor(ef2[-1])

# pane with 100 colors using RcolorBrewer
my_colors <- brewer.pal(5, 'Spectral')
my_colors <- colorRampPalette(my_colors)(100)

# ordering the correlation matrix
ord <- order(data[1, ])
data_ord <- data[ord, ord]
plotcorr(data_ord , col=my_colors[data_ord*50+50] , mar=c(1,1,1,1))

# Principal Component Analysis (PCA)
(pc <- prcomp(na.omit(ef2[-1]), scale = TRUE))
stats::screeplot(pc, type = 'lines')

autoplot(pc, data = na.omit(ef2), colour = 'file_name',
loadings = TRUE, loadings.label = TRUE)
```

`extract_features_py` *Extract features from WAV audios using Python's Parselmouth library.*

Description

Extract features from WAV audios using Python's Parselmouth library.

Usage

```
extract_features_py(
  directory,
  filesRange = 0,
  features = c("f0", "formants"),
  windowShift = 5/1000,
  full.names = TRUE,
  recursive = FALSE
)
```

Arguments

`directory`	A directory/folder containing WAV files.
`filesRange`	The desired range of directory files (default: 0, i.e., all files).
`features`	Vector of features to be extracted. (default: 'f0' (pitch),'formants' (F1:F8)).
`windowShift`	= <dur> set analysis window shift to <dur>ation in ms (default: 5/1000).
`full.names`	Logical. If `TRUE`, the directory path is prepended to the file names to give a relative file path. If `FALSE`, the file names (rather than paths) are returned. (default: `TRUE`)
`recursive`	Logical. Should the listing recursively into directories? (default: `FALSE`)

Details

The function uses the `getwd()` folder to write the temp files.

Value

`char` vector containing the expanded models.

Examples

```
## Not run:
library(voice)

path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)
efp <- extract_features_py(dirname(path2wav))
efp
table(efp$file_name)

# limiting filesRange
efpl <- extract_features_py(dirname(path2wav), filesRange = 3:6)
efpl
table(efpl$file_name)

## End(Not run)
```

`is_mono` *Verify if an audio is mono.*

Description

Verify if an audio is mono.

Usage

```
is_mono(x)
```

Arguments

`x`	Path to WAV audio file.

Examples

```
library(voice)
# get path to audio file
path2wav <- list.files(system.file('extdata', package = 'wrassp'),
pattern <- glob2rx('*.wav'), full.names = TRUE)
is_mono(path2wav[1])
sapply(path2wav, is_mono)
```

memory *Gives the amount of memory RAM free, total and percentage = free/total.*

Description

Gives the amount of memory RAM free, total and percentage = free/total.

Usage

```
memory()
```

Examples

```
## Not run:
library(voice)
memory()

## End(Not run)
```

rm0 *Transforms* n *sets of* m>n *zeros (alternated with sets of non zeros) into* n *sets of* n *zeros.*

Description

Transforms n sets of m>n zeros (alternated with sets of non zeros) into n sets of n zeros.

Usage

```
rm0(y)
```

Arguments

y	A vector or time series.

Value

Vector with n zeros.

Examples

```
(v0 <- c(1:20,rep(0,10)))
(r0 <- rm0(v0))
length(v0)
length(r0)
sum(v0 == 0)

(v1 <- c(rep(0,10),1:20))
(r1 <- rm0(v1))
length(r1)
```

```
(v2 <- rep(0,10))
(r2 <- rm0(v2))
length(r2)

(v3 <- c(0:10))
(r3 <- rm0(v3))
length(r3)

(v4 <- c(rep(0,10), 1:10, rep(0,5), 10:20, rep(0,10)))
(r4 <- rm0(v4))
length(r4)
sum(v4 == 0)
```

`rp` *Prenda's rule. Returns* `2t+p`.

Description

Prenda's rule. Returns `2t+p`.

Usage

```
rp(true, pred)
```

Arguments

`true`	A number.
`pred`	A number.

Value

`t,p -> 2t+p`

`0,0 -> 0`

`0,1 -> 1`

`1,0 -> 2`

`1,1 -> 3`

`10,12 -> 32`

`12,10 -> 34`

Examples

```
rp(0,0) # 0
rp(0,1) # 1
rp(1,0) # 2
rp(1,1) # 3
rp(10,12) # 32
rp(12,10) # 34
```

`write_list` *Writes a list to a path.*

Description

Writes a list to a path.

Usage

```
write_list(x, directory)
```

Arguments

`x`	A list.
`directory`	A directory.

Index