DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROJECT REPORT

Topic: voice gender recognition using r programming language

Team members:

1.k.ajmaldeen -121012012717

2.m.manoj -121012012747

3.m.vaseekaran -121012012773

4.m.harshini -121011012776

Project description:

* Gender voice is considered one of the pivotal parts to be detected from a given voice,a task that involves certain complications.inorder to distinguish gender from a voice signal,a set of techniques have been employed to determine relevant features to be utilized for building a model from a training set

Goal:

* This project trains a computer program to identify a voice as male or female, based upon acoustic properties of the voice and speech. The model is trained on a dataset consisting of 3,168 recorded voice samples, collected from male and female speakers.
* The voice samples are pre-processed by acoustic analysis in R and then processed with artificial intelligence/machine learning algorithms to learn gender-specific traits for classifying the voice as male or female.
* The best model achieves an accuracy of 100% on the training set and 89% on the test set

Dataset information

Download the pre-processed [dataset](https://raw.githubusercontent.com/primaryobjects/voice-gender/master/voice.csv) as a CSV file.

The CSV file contains the following fields:

"meanfreq","sd","median","Q25","Q75","IQR","skew","kurt","sp.ent","sfm","mode","centroid","meanfun","minfun","maxfun","meandom","mindom","maxdom","dfrange","modindx","label"

"label" corresponds to the gender classification of the sample. The remaining fields are acoustic properties, detailed [below](https://github.com/primaryobjects/voice-gender#acoustic-properties-measured).

In R, you can load the dataset file [data.bin](https://raw.githubusercontent.com/primaryobjects/voice-gender/master/data.bin) directly as a data.frame with the command load('data.bin').

In addition to the pre-processed dataset, the raw voice samples used for training are included as .WAV files in a separate repository. The .WAV files are pre-processed in R to produce the above dataset.

Source code for voice gender recognition using r

|  |
| --- |
|  |
|  |  | library(seewave) |
|  |  | library(caTools) |
|  |  | library(caret) |
|  |  | library(rpart) |
|  |  | library(rpart.plot) |
|  |  | library(randomForest) |
|  |  | library(warbleR) |
|  |  | library(mice) |
|  |  | library(xgboost) |
|  |  |  |
|  |  | specan3 <- function(X, bp = c(0,22), wl = 512, threshold = 15, parallel = 1){ |
|  |  | # To use parallel processing: library(devtools), install\_github('nathanvan/parallelsugar') |
|  |  | if(class(X) == "data.frame") {if(all(c("sound.files", "selec", |
|  |  | "start", "end") %in% colnames(X))) |
|  |  | { |
|  |  | start <- as.numeric(unlist(X$start)) |
|  |  | end <- as.numeric(unlist(X$end)) |
|  |  | sound.files <- as.character(unlist(X$sound.files)) |
|  |  | selec <- as.character(unlist(X$selec)) |
|  |  | } else stop(paste(paste(c("sound.files", "selec", "start", "end")[!(c("sound.files", "selec", |
|  |  | "start", "end") %in% colnames(X))], collapse=", "), "column(s) not found in data frame")) |
|  |  | } else stop("X is not a data frame") |
|  |  |  |
|  |  | #if there are NAs in start or end stop |
|  |  | if(any(is.na(c(end, start)))) stop("NAs found in start and/or end") |
|  |  |  |
|  |  | #if end or start are not numeric stop |
|  |  | if(all(class(end) != "numeric" & class(start) != "numeric")) stop("'end' and 'selec' must be numeric") |
|  |  |  |
|  |  | #if any start higher than end stop |
|  |  | if(any(end - start<0)) stop(paste("The start is higher than the end in", length(which(end - start<0)), "case(s)")) |
|  |  |  |
|  |  | #if any selections longer than 20 secs stop |
|  |  | if(any(end - start>20)) stop(paste(length(which(end - start>20)), "selection(s) longer than 20 sec")) |
|  |  | options( show.error.messages = TRUE) |
|  |  |  |
|  |  | #if bp is not vector or length!=2 stop |
|  |  | if(!is.vector(bp)) stop("'bp' must be a numeric vector of length 2") else{ |
|  |  | if(!length(bp) == 2) stop("'bp' must be a numeric vector of length 2")} |
|  |  |  |
|  |  | #return warning if not all sound files were found |
|  |  | fs <- list.files(path = getwd(), pattern = ".wav$", ignore.case = TRUE) |
|  |  | if(length(unique(sound.files[(sound.files %in% fs)])) != length(unique(sound.files))) |
|  |  | cat(paste(length(unique(sound.files))-length(unique(sound.files[(sound.files %in% fs)])), |
|  |  | ".wav file(s) not found")) |
|  |  |  |
|  |  | #count number of sound files in working directory and if 0 stop |
|  |  | d <- which(sound.files %in% fs) |
|  |  | if(length(d) == 0){ |
|  |  | stop("The .wav files are not in the working directory") |
|  |  | } else { |
|  |  | start <- start[d] |
|  |  | end <- end[d] |
|  |  | selec <- selec[d] |
|  |  | sound.files <- sound.files[d] |
|  |  | } |
|  |  |  |
|  |  | # If parallel is not numeric |
|  |  | if(!is.numeric(parallel)) stop("'parallel' must be a numeric vector of length 1") |
|  |  | if(any(!(parallel %% 1 == 0),parallel < 1)) stop("'parallel' should be a positive integer") |
|  |  |  |
|  |  | # If parallel was called |
|  |  | if(parallel > 1) |
|  |  | { options(warn = -1) |
|  |  | if(all(Sys.info()[1] == "Windows",requireNamespace("parallelsugar", quietly = TRUE) == TRUE)) |
|  |  | lapp <- function(X, FUN) parallelsugar::mclapply(X, FUN, mc.cores = parallel) else |
|  |  | if(Sys.info()[1] == "Windows"){ |
|  |  | cat("Windows users need to install the 'parallelsugar' package for parallel computing (you are not doing it now!)") |
|  |  | lapp <- pbapply::pblapply} else lapp <- function(X, FUN) parallel::mclapply(X, FUN, mc.cores = parallel)} else lapp <- pbapply::pblapply |
|  |  |  |
|  |  | options(warn = 0) |
|  |  |  |
|  |  | if(parallel == 1) cat("Measuring acoustic parameters:") |
|  |  | x <- as.data.frame(lapp(1:length(start), function(i) { |
|  |  | r <- tuneR::readWave(file.path(getwd(), sound.files[i]), from = start[i], to = end[i], units = "seconds") |
|  |  |  |
|  |  | b<- bp #in case bp its higher than can be due to sampling rate |
|  |  | if(b[2] > ceiling(r@samp.rate/2000) - 1) b[2] <- ceiling(r@samp.rate/2000) - 1 |
|  |  |  |
|  |  |  |
|  |  | #frequency spectrum analysis |
|  |  | songspec <- seewave::spec(r, f = r@samp.rate, plot = FALSE) |
|  |  | analysis <- seewave::specprop(songspec, f = r@samp.rate, flim = b, plot = FALSE) |
|  |  |  |
|  |  | #save parameters |
|  |  | meanfreq <- analysis$mean/1000 |
|  |  | sd <- analysis$sd/1000 |
|  |  | median <- analysis$median/1000 |
|  |  | Q25 <- analysis$Q25/1000 |
|  |  | Q75 <- analysis$Q75/1000 |
|  |  | IQR <- analysis$IQR/1000 |
|  |  | skew <- analysis$skewness |
|  |  | kurt <- analysis$kurtosis |
|  |  | sp.ent <- analysis$sh |
|  |  | sfm <- analysis$sfm |
|  |  | mode <- analysis$mode/1000 |
|  |  | centroid <- analysis$cent/1000 |
|  |  |  |
|  |  | #Frequency with amplitude peaks |
|  |  | peakf <- 0#seewave::fpeaks(songspec, f = r@samp.rate, wl = wl, nmax = 3, plot = FALSE)[1, 1] |
|  |  |  |
|  |  | #Fundamental frequency parameters |
|  |  | ff <- seewave::fund(r, f = r@samp.rate, ovlp = 50, threshold = threshold, |
|  |  | fmax = b[2] \* 1000, plot = FALSE, wl = wl)[, 2] |
|  |  | meanfun<-mean(ff, na.rm = T) |
|  |  | minfun<-min(ff, na.rm = T) |
|  |  | maxfun<-max(ff, na.rm = T) |
|  |  |  |
|  |  | #Dominant frecuency parameters |
|  |  | y <- seewave::dfreq(r, f = r@samp.rate, wl = wl, ovlp = 0, plot = F, threshold = threshold, bandpass = b \* 1000, fftw = TRUE)[, 2] |
|  |  | meandom <- mean(y, na.rm = TRUE) |
|  |  | mindom <- min(y, na.rm = TRUE) |
|  |  | maxdom <- max(y, na.rm = TRUE) |
|  |  | dfrange <- (maxdom - mindom) |
|  |  | duration <- (end[i] - start[i]) |
|  |  |  |
|  |  | #modulation index calculation |
|  |  | changes <- vector() |
|  |  | for(j in which(!is.na(y))){ |
|  |  | change <- abs(y[j] - y[j + 1]) |
|  |  | changes <- append(changes, change) |
|  |  | } |
|  |  | if(mindom==maxdom) modindx<-0 else modindx <- mean(changes, na.rm = T)/dfrange |
|  |  |  |
|  |  | #save results |
|  |  | return(c(duration, meanfreq, sd, median, Q25, Q75, IQR, skew, kurt, sp.ent, sfm, mode, |
|  |  | centroid, peakf, meanfun, minfun, maxfun, meandom, mindom, maxdom, dfrange, modindx)) |
|  |  | })) |
|  |  |  |
|  |  | #change result names |
|  |  |  |
|  |  | rownames(x) <- c("duration", "meanfreq", "sd", "median", "Q25", "Q75", "IQR", "skew", "kurt", "sp.ent", |
|  |  | "sfm","mode", "centroid", "peakf", "meanfun", "minfun", "maxfun", "meandom", "mindom", "maxdom", "dfrange", "modindx") |
|  |  | x <- data.frame(sound.files, selec, as.data.frame(t(x))) |
|  |  | colnames(x)[1:2] <- c("sound.files", "selec") |
|  |  | rownames(x) <- c(1:nrow(x)) |
|  |  |  |
|  |  | return(x) |
|  |  | } |
|  |  |  |
|  |  | processFolder <- function(folderName) { |
|  |  | # Start with empty data.frame. |
|  |  | data <- data.frame() |
|  |  |  |
|  |  | # Get list of files in the folder. |
|  |  | list <- list.files(folderName, '\\.wav') |
|  |  |  |
|  |  | # Add file list to data.frame for processing. |
|  |  | for (fileName in list) { |
|  |  | row <- data.frame(fileName, 0, 0, 20) |
|  |  | data <- rbind(data, row) |
|  |  | } |
|  |  |  |
|  |  | # Set column names. |
|  |  | names(data) <- c('sound.files', 'selec', 'start', 'end') |
|  |  |  |
|  |  | # Move into folder for processing. |
|  |  | setwd(folderName) |
|  |  |  |
|  |  | # Process files. |
|  |  | acoustics <- specan3(data, parallel=1) |
|  |  |  |
|  |  | # Move back into parent folder. |
|  |  | setwd('..') |
|  |  |  |
|  |  | acoustics |
|  |  | } |
|  |  |  |
|  |  | gender <- function(filePath) { |
|  |  | if (!exists('genderBoosted')) { |
|  |  | load('model.bin') |
|  |  | } |
|  |  |  |
|  |  | # Setup paths. |
|  |  | currentPath <- getwd() |
|  |  | fileName <- basename(filePath) |
|  |  | path <- dirname(filePath) |
|  |  |  |
|  |  | # Set directory to read file. |
|  |  | setwd(path) |
|  |  |  |
|  |  | # Start with empty data.frame. |
|  |  | data <- data.frame(fileName, 0, 0, 20) |
|  |  |  |
|  |  | # Set column names. |
|  |  | names(data) <- c('sound.files', 'selec', 'start', 'end') |
|  |  |  |
|  |  | # Process files. |
|  |  | acoustics <- specan3(data, parallel=1) |
|  |  |  |
|  |  | # Restore path. |
|  |  | setwd(currentPath) |
|  |  |  |
|  |  | predict(genderCombo, newdata=acoustics) |
|  |  | } |
|  |  |  |
|  |  | # Load data |
|  |  | males <- processFolder('male') |
|  |  | females <- processFolder('female') |
|  |  |  |
|  |  | # Set labels. |
|  |  | males$label <- 1 |
|  |  | females$label <- 2 |
|  |  | data <- rbind(males, females) |
|  |  | data$label <- factor(data$label, labels=c('male', 'female')) |
|  |  |  |
|  |  | # Remove unused columns. |
|  |  | data$duration <- NULL |
|  |  | data$sound.files <- NULL |
|  |  | data$selec <- NULL |
|  |  | data$peakf <- NULL |
|  |  |  |
|  |  | # Remove rows containing NA's. |
|  |  | data <- data[complete.cases(data),] |
|  |  |  |
|  |  | # Write out csv dataset. |
|  |  | write.csv(data, file='voice.csv', sep=',', row.names=F) |
|  |  |  |
|  |  | # Create a train and test set. |
|  |  | set.seed(777) |
|  |  | spl <- sample.split(data$label, 0.7) |
|  |  | train <- subset(data, spl == TRUE) |
|  |  | test <- subset(data, spl == FALSE) |
|  |  |  |
|  |  | # Build models. |
|  |  | genderLog <- glm(label ~ ., data=train, family='binomial') |
|  |  | genderCART <- rpart(label ~ ., data=train, method='class') |
|  |  | prp(genderCART) |
|  |  | genderForest <- randomForest(label ~ ., data=train) |
|  |  |  |
|  |  | # Assume a basline model of always predicting male. |
|  |  | # Accuracy: 0.50 |
|  |  | table(train$label) |
|  |  | 1107 / nrow(train) |
|  |  |  |
|  |  | # Accuracy: 0.50 |
|  |  | table(test$label) |
|  |  | 475 / nrow(test) |
|  |  |  |
|  |  | # Accuracy: 0.72 |
|  |  | predictLog <- predict(genderLog, type='response') |
|  |  | table(train$label, predictLog >= 0.5) |
|  |  | (814 + 777) / nrow(train) |
|  |  |  |
|  |  | # Accuracy: 0.71 |
|  |  | predictLog2 <- predict(genderLog, newdata=test, type='response') |
|  |  | table(test$label, predictLog2 >= 0.5) |
|  |  | (339 + 335) / nrow(test) |
|  |  |  |
|  |  | # Accuracy: 0.81 |
|  |  | predictCART <- predict(genderCART) |
|  |  | predictCART.prob <- predictCART[,2] |
|  |  | table(train$label, predictCART.prob >= 0.5) |
|  |  | (858 + 941) / nrow(train) |
|  |  |  |
|  |  | # Accuracy: 0.78 |
|  |  | predictCART2 <- predict(genderCART, newdata=test) |
|  |  | predictCART2.prob <- predictCART2[,2] |
|  |  | table(test$label, predictCART2.prob >= 0.5) |
|  |  | (364 + 378) / nrow(test) |
|  |  |  |
|  |  | # Accuracy: 1 |
|  |  | predictForest <- predict(genderForest, newdata=train) |
|  |  | table(train$label, predictForest) |
|  |  |  |
|  |  | # Accuracy: 0.86 |
|  |  | predictForest <- predict(genderForest, newdata=test) |
|  |  | table(test$label, predictForest) |
|  |  | (410 + 409) / nrow(test) |
|  |  |  |
|  |  | # Tune random-forest and return best model. |
|  |  | # Accuracy: 0.87 |
|  |  | set.seed(777) |
|  |  | genderTunedForest <- tuneRF(train[, -21], train[, 21], stepFactor=.5, doBest=TRUE) |
|  |  | predictForest <- predict(genderTunedForest, newdata=test) |
|  |  | table(test$label, predictForest) |
|  |  | (412 + 416) / nrow(test) |
|  |  |  |
|  |  | # Try a boosted tree model. |
|  |  | # Accuracy: 0.91 |
|  |  | set.seed(777) |
|  |  | genderBoosted <- train(label ~ ., data=train, method='gbm') |
|  |  | predictBoosted <- predict(genderBoosted, newdata=train) |
|  |  | confusionMatrix(predictBoosted, train$label) |
|  |  |  |
|  |  | # Accuracy: 0.84 |
|  |  | predictBoosted <- predict(genderBoosted, newdata=test) |
|  |  | confusionMatrix(predictBoosted, test$label) |
|  |  |  |
|  |  | # Try XGBoost. |
|  |  | # Accuracy: 1 |
|  |  | trainx <- sapply(train, as.numeric) |
|  |  | trainx[,21] <- trainx[,21] - 1 |
|  |  | set.seed(777) |
|  |  | genderXG <- xgboost(data = trainx[,-21], label = trainx[,21], eta=0.2, nround = 500, subsample = 0.5, colsample\_bytree = 0.5, objective = "binary:logistic") |
|  |  | results <- predict(genderXG, trainx) |
|  |  | table(trainx[,21], results >= 0.5) |
|  |  |  |
|  |  | # Accuracy: 0.87 |
|  |  | testx <- sapply(test, as.numeric) |
|  |  | testx[,21] <- testx[,21] - 1 |
|  |  | results <- predict(genderXG, testx) |
|  |  | table(testx[,21], results >= 0.5) |
|  |  | (414 + 413) / nrow(test) |
|  |  |  |
|  |  | # trans <- processFolder('trans') |
|  |  | # trans$label <- c(2, 2, 1, 1, 2, 2, 1, 1) |
|  |  | # trans$label <- factor(trans$label, labels=c('male', 'female')) |
|  |  | # tpred <- predict(genderLog, newdata=trans, type='response') |
|  |  | # table(trans$label, tpred >= 0.5) |
|  |  |  |
|  |  | # tpred <- predict(genderForest, newdata=trans) |
|  |  | # table(trans$label, tpred) |
|  |  |  |
|  |  | # tpred <- predict(genderBoosted, newdata=trans) |
|  |  | # confusionMatrix(trans$label, tpred) |
|  |  |  |
|  |  | # tpred <- predict(genderCART, newdata=trans) |
|  |  | # tpred.prob <- tpred[,2] |
|  |  | # table(trans$label, tpred.prob >= 0.5) |
|  |  |  |
|  |  | # trans <- processFolder('trans2') |
|  |  | # trans$label <- c(2, 2, 2, 1, 2) |
|  |  | # trans$label <- factor(trans$label, labels=c('male', 'female')) |
|  |  | # tpred <- predict(genderLog, newdata=trans, type='response') |
|  |  | # table(trans$label, tpred >= 0.5) |
|  |  |  |
|  |  | # tpred <- predict(genderForest, newdata=trans) |
|  |  | # table(trans$label, tpred) |
|  |  |  |
|  |  | # tpred <- predict(genderBoosted, newdata=trans) |
|  |  | # confusionMatrix(trans$label, tpred) |
|  |  |  |
|  |  | # tpred <- predict(genderCART, newdata=trans) |
|  |  | # tpred.prob <- tpred[,2] |
|  |  | # table(trans$label, tpred.prob >= 0.5) |
|  |  |  |
|  |  | # trans <- processFolder('trans4') |
|  |  | # trans$label <- c(2, 2, 2, 2, 1, 1, 1, 1) |
|  |  | # trans$label <- factor(trans$label, labels=c('male', 'female')) |
|  |  | # |
|  |  | # tpred <- predict(genderLog, newdata=trans, type='response') |
|  |  | # table(trans$label, tpred >= 0.5) |
|  |  | # |
|  |  | # tpred <- predict(genderTunedForest, newdata=trans, type='response') |
|  |  | # table(trans$label, tpred) |
|  |  | # |
|  |  | # trans[,1:3] <- NULL |
|  |  | # trans$peakf <- NULL |
|  |  | # testx <- sapply(trans, as.numeric) |
|  |  | # testx[,21] <- testx[,21] - 1 |
|  |  | # results <- predict(genderXG, testx) |
|  |  | # table(testx[,21], results >= 0.5) |
| |  |  |  | | --- | --- | --- | |  |  |  | |  |  |  |   **Accuracy**  **The trained models have achieved the following accuracies (train/test):**  **Baseline Algorithm (always male)**  **50%/50%**  **Baseline Algorithm (simple frequency threshold)**  **61%/59%**  **Logistic Regression**  **72%/71%**  **Classification and Regression Tree (CART)**  **81%/78%**  **Random Forest**  **100%/87%**  **Generalized Boosted Tree Regression**  **91%/84%**  **XGBoost**  **100%/87%**  **XGBoost (Updated with frequency range 0hz-280hz)**  **100%/99%**  **Acoustic Properties Measured**  **The following acoustic properties of each voice are measured:**   * **duration: length of signal** * **meanfreq: mean frequency (in kHz)** * **sd: standard deviation of frequency** * **median: median frequency (in kHz)** * **Q25: first quantile (in kHz)** * **Q75: third quantile (in kHz)** * **IQR: interquantile range (in kHz)** * **skew: skewness (see note in specprop description)** * **kurt: kurtosis (see note in specprop description)** * **sp.ent: spectral entropy** * **sfm: spectral flatness** * **mode: mode frequency** * **centroid: frequency centroid (see specprop)** * **peakf: peak frequency (frequency with highest energy)** * **meanfun: average of fundamental frequency measured across acoustic signal** * **minfun: minimum fundamental frequency measured across acoustic signal** * **maxfun: maximum fundamental frequency measured across acoustic signal** * **meandom: average of dominant frequency measured across acoustic signal** * **mindom: minimum of dominant frequency measured across acoustic signal** * **maxdom: maximum of dominant frequency measured across acoustic signal** * **dfrange: range of dominant frequency measured across acoustic signal** * **modindx: modulation index. Calculated as the accumulated absolute difference between adjacent measurements of fundamental frequencies divided by the frequency range**   **Classification and Regression Decision Tree**  **The following decision tree, produced by the CART model, provides a high-level overview of important properties of the voice samples that may determine a specific gender classification of male versus female.**  **[Screenshot 1](https://raw.githubusercontent.com/primaryobjects/voice-gender/master/images/voice-plot-1.png)**  **After narrowing the frequency range to 0hz-280hz with a sound threshold of 15%, the accuracy is boosted to near perfect, and the following CART model is described. Mean fundamental frequency serves as a powerful indicator of voice gender, with a threshold of 140hz separating male from female classifications.**  **[Screenshot 2](https://raw.githubusercontent.com/primaryobjects/voice-gender/master/images/voice-plot-2.png)** | | | |  |  |
| Conclusion:  The aim of this project is to build a gender recognition system based on voice. As stated above std, dev, kurtosis, and skewness seem to have the most discriminant power. Some unexpected behavior is in the peaks at very low frequencies (< 50 Hz) that can be viewed in mode and median. This can be due to the presence of noise in the audio recordings. Even if it was filtered, not all sources of noise were accounted for and further analysis in this way would be interesting but falls beyond the scope of this project. | | | |  | install.packages(setdiff(packages, rownames(installed.packages()))) |
|  | | | |  | } |
|  | | | |  | library(tuneR) |
|  | | | |  | library(seewave) |
|  | | | |  | library(caTools) |
|  | | | |  | library(caret) |
|  | | | |  | library(rpart) |
|  | | | |  | library(rpart.plot) |
|  | | | |  | library(randomForest) |
|  | | | |  | library(warbleR) |
|  | | | |  | library(mice) |
|  | | | |  | library(xgboost) |
|  | | | |  |  |
|  | | | |  | specan3 <- function(X, bp = c(0,22), wl = 512, threshold = 15, parallel = 1){ |
|  | | | |  | # To use parallel processing: library(devtools), install\_github('nathanvan/parallelsugar') |
|  | | | |  | if(class(X) == "data.frame") {if(all(c("sound.files", "selec", |
|  | | | |  | "start", "end") %in% colnames(X))) |
|  | | | |  | { |
|  | | | |  | start <- as.numeric(unlist(X$start)) |
|  | | | |  | end <- as.numeric(unlist(X$end)) |
|  | | | |  | sound.files <- as.character(unlist(X$sound.files)) |
|  | | | |  | selec <- as.character(unlist(X$selec)) |
|  | | | |  | } else stop(paste(paste(c("sound.files", "selec", "start", "end")[!(c("sound.files", "selec", |
|  | | | |  | "start", "end") %in% colnames(X))], collapse=", "), "column(s) not found in data frame")) |
|  | | | |  | } else stop("X is not a data frame") |
|  | | | |  |  |
|  | | | |  | #if there are NAs in start or end stop |
|  | | | |  | if(any(is.na(c(end, start)))) stop("NAs found in start and/or end") |
|  | | | |  |  |
|  | | | |  | #if end or start are not numeric stop |
|  | | | |  | if(all(class(end) != "numeric" & class(start) != "numeric")) stop("'end' and 'selec' must be numeric") |
|  | | | |  |  |
|  | | | |  | #if any start higher than end stop |
|  | | | |  | if(any(end - start<0)) stop(paste("The start is higher than the end in", length(which(end - start<0)), "case(s)")) |
|  | | | |  |  |
|  | | | |  | #if any selections longer than 20 secs stop |
|  | | | |  | if(any(end - start>20)) stop(paste(length(which(end - start>20)), "selection(s) longer than 20 sec")) |
|  | | | |  | options( show.error.messages = TRUE) |
|  | | | |  |  |
|  | | | |  | #if bp is not vector or length!=2 stop |
|  | | | |  | if(!is.vector(bp)) stop("'bp' must be a numeric vector of length 2") else{ |
|  | | | |  | if(!length(bp) == 2) stop("'bp' must be a numeric vector of length 2")} |
|  | | | |  |  |
|  | | | |  | #return warning if not all sound files were found |
|  | | | |  | fs <- list.files(path = getwd(), pattern = ".wav$", ignore.case = TRUE) |
|  | | | |  | if(length(unique(sound.files[(sound.files %in% fs)])) != length(unique(sound.files))) |
|  | | | |  | cat(paste(length(unique(sound.files))-length(unique(sound.files[(sound.files %in% fs)])), |
|  | | | |  | ".wav file(s) not found")) |
|  | | | |  |  |
|  | | | |  | #count number of sound files in working directory and if 0 stop |
|  | | | |  | d <- which(sound.files %in% fs) |
|  | | | |  | if(length(d) == 0){ |
|  | | | |  | stop("The .wav files are not in the working directory") |
|  | | | |  | } else { |
|  | | | |  | start <- start[d] |
|  | | | |  | end <- end[d] |
|  | | | |  | selec <- selec[d] |
|  | | | |  | sound.files <- sound.files[d] |
|  | | | |  | } |
|  | | | |  |  |
|  | | | |  | # If parallel is not numeric |
|  | | | |  | if(!is.numeric(parallel)) stop("'parallel' must be a numeric vector of length 1") |
|  | | | |  | if(any(!(parallel %% 1 == 0),parallel < 1)) stop("'parallel' should be a positive integer") |
|  | | | |  |  |
|  | | | |  | # If parallel was called |
|  | | | |  | if(parallel > 1) |
|  | | | |  | { options(warn = -1) |
|  | | | |  | if(all(Sys.info()[1] == "Windows",requireNamespace("parallelsugar", quietly = TRUE) == TRUE)) |
|  | | | |  | lapp <- function(X, FUN) parallelsugar::mclapply(X, FUN, mc.cores = parallel) else |
|  | | | |  | if(Sys.info()[1] == "Windows"){ |
|  | | | |  | cat("Windows users need to install the 'parallelsugar' package for parallel computing (you are not doing it now!)") |
|  | | | |  | lapp <- pbapply::pblapply} else lapp <- function(X, FUN) parallel::mclapply(X, FUN, mc.cores = parallel)} else lapp <- pbapply::pblapply |
|  | | | |  |  |
|  | | | |  | options(warn = 0) |
|  | | | |  |  |
|  | | | |  | if(parallel == 1) cat("Measuring acoustic parameters:") |
|  | | | |  | x <- as.data.frame(lapp(1:length(start), function(i) { |
|  | | | |  | r <- tuneR::readWave(file.path(getwd(), sound.files[i]), from = start[i], to = end[i], units = "seconds") |
|  | | | |  |  |
|  | | | |  | b<- bp #in case bp its higher than can be due to sampling rate |
|  | | | |  | if(b[2] > ceiling(r@samp.rate/2000) - 1) b[2] <- ceiling(r@samp.rate/2000) - 1 |
|  | | | |  |  |
|  | | | |  |  |
|  | | | |  | #frequency spectrum analysis |
|  | | | |  | songspec <- seewave::spec(r, f = r@samp.rate, plot = FALSE) |
|  | | | |  | analysis <- seewave::specprop(songspec, f = r@samp.rate, flim = b, plot = FALSE) |
|  | | | |  |  |
|  | | | |  | #save parameters |
|  | | | |  | meanfreq <- analysis$mean/1000 |
|  | | | |  | sd <- analysis$sd/1000 |
|  | | | |  | median <- analysis$median/1000 |
|  | | | |  | Q25 <- analysis$Q25/1000 |
|  | | | |  | Q75 <- analysis$Q75/1000 |
|  | | | |  | IQR <- analysis$IQR/1000 |
|  | | | |  | skew <- analysis$skewness |
|  | | | |  | kurt <- analysis$kurtosis |
|  | | | |  | sp.ent <- analysis$sh |
|  | | | |  | sfm <- analysis$sfm |
|  | | | |  | mode <- analysis$mode/1000 |
|  | | | |  | centroid <- analysis$cent/1000 |
|  | | | |  |  |
|  | | | |  | #Frequency with amplitude peaks |
|  | | | |  | peakf <- 0#seewave::fpeaks(songspec, f = r@samp.rate, wl = wl, nmax = 3, plot = FALSE)[1, 1] |
|  | | | |  |  |
|  | | | |  | #Fundamental frequency parameters |
|  | | | |  | ff <- seewave::fund(r, f = r@samp.rate, ovlp = 50, threshold = threshold, |
|  | | | |  | fmax = b[2] \* 1000, plot = FALSE, wl = wl)[, 2] |
|  | | | |  | meanfun<-mean(ff, na.rm = T) |
|  | | | |  | minfun<-min(ff, na.rm = T) |
|  | | | |  | maxfun<-max(ff, na.rm = T) |
|  | | | |  |  |
|  | | | |  | #Dominant frecuency parameters |
|  | | | |  | y <- seewave::dfreq(r, f = r@samp.rate, wl = wl, ovlp = 0, plot = F, threshold = threshold, bandpass = b \* 1000, fftw = TRUE)[, 2] |
|  | | | |  | meandom <- mean(y, na.rm = TRUE) |
|  | | | |  | mindom <- min(y, na.rm = TRUE) |
|  | | | |  | maxdom <- max(y, na.rm = TRUE) |
|  | | | |  | dfrange <- (maxdom - mindom) |
|  | | | |  | duration <- (end[i] - start[i]) |
|  | | | |  |  |
|  | | | |  | #modulation index calculation |
|  | | | |  | changes <- vector() |
|  | | | |  | for(j in which(!is.na(y))){ |
|  | | | |  | change <- abs(y[j] - y[j + 1]) |
|  | | | |  | changes <- append(changes, change) |
|  | | | |  | } |
|  | | | |  | if(mindom==maxdom) modindx<-0 else modindx <- mean(changes, na.rm = T)/dfrange |
|  | | | |  |  |
|  | | | |  | #save results |
|  | | | |  | return(c(duration, meanfreq, sd, median, Q25, Q75, IQR, skew, kurt, sp.ent, sfm, mode, |
|  | | | |  | centroid, peakf, meanfun, minfun, maxfun, meandom, mindom, maxdom, dfrange, modindx)) |
|  | | | |  | })) |
|  | | | |  |  |
|  | | | |  | #change result names |
|  | | | |  |  |
|  | | | |  | rownames(x) <- c("duration", "meanfreq", "sd", "median", "Q25", "Q75", "IQR", "skew", "kurt", "sp.ent", |
|  | | | |  | "sfm","mode", "centroid", "peakf", "meanfun", "minfun", "maxfun", "meandom", "mindom", "maxdom", "dfrange", "modindx") |
|  | | | |  | x <- data.frame(sound.files, selec, as.data.frame(t(x))) |
|  | | | |  | colnames(x)[1:2] <- c("sound.files", "selec") |
|  | | | |  | rownames(x) <- c(1:nrow(x)) |
|  | | | |  |  |
|  | | | |  | return(x) |
|  | | | |  | } |
|  | | | |  |  |
|  | | | |  | processFolder <- function(folderName) { |
|  | | | |  | # Start with empty data.frame. |
|  | | | |  | data <- data.frame() |
|  | | | |  |  |
|  | | | |  | # Get list of files in the folder. |
|  | | | |  | list <- list.files(folderName, '\\.wav') |
|  | | | |  |  |
|  | | | |  | # Add file list to data.frame for processing. |
|  | | | |  | for (fileName in list) { |
|  | | | |  | row <- data.frame(fileName, 0, 0, 20) |
|  | | | |  | data <- rbind(data, row) |
|  | | | |  | } |
|  | | | |  |  |
|  | | | |  | # Set column names. |
|  | | | |  | names(data) <- c('sound.files', 'selec', 'start', 'end') |
|  | | | |  |  |
|  | | | |  | # Move into folder for processing. |
|  | | | |  | setwd(folderName) |
|  | | | |  |  |
|  | | | |  | # Process files. |
|  | | | |  | acoustics <- specan3(data, parallel=1) |
|  | | | |  |  |
|  | | | |  | # Move back into parent folder. |
|  | | | |  | setwd('..') |
|  | | | |  |  |
|  | | | |  | acoustics |
|  | | | |  | } |
|  | | | |  |  |
|  | | | |  | gender <- function(filePath) { |
|  | | | |  | if (!exists('genderBoosted')) { |
|  | | | |  | load('model.bin') |
|  | | | |  | } |
|  | | | |  |  |
|  | | | |  | # Setup paths. |
|  | | | |  | currentPath <- getwd() |
|  | | | |  | fileName <- basename(filePath) |
|  | | | |  | path <- dirname(filePath) |
|  | | | |  |  |
|  | | | |  | # Set directory to read file. |
|  | | | |  | setwd(path) |
|  | | | |  |  |
|  | | | |  | # Start with empty data.frame. |
|  | | | |  | data <- data.frame(fileName, 0, 0, 20) |
|  | | | |  |  |
|  | | | |  | # Set column names. |
|  | | | |  | names(data) <- c('sound.files', 'selec', 'start', 'end') |
|  | | | |  |  |
|  | | | |  | # Process files. |
|  | | | |  | acoustics <- specan3(data, parallel=1) |
|  | | | |  |  |
|  | | | |  | # Restore path. |
|  | | | |  | setwd(currentPath) |
|  | | | |  |  |
|  | | | |  | predict(genderCombo, newdata=acoustics) |
|  | | | |  | } |
|  | | | |  |  |
|  | | | |  | # Load data |
|  | | | |  | males <- processFolder('male') |
|  | | | |  | females <- processFolder('female') |
|  | | | |  |  |
|  | | | |  | # Set labels. |
|  | | | |  | males$label <- 1 |
|  | | | |  | females$label <- 2 |
|  | | | |  | data <- rbind(males, females) |
|  | | | |  | data$label <- factor(data$label, labels=c('male', 'female')) |
|  | | | |  |  |
|  | | | |  | # Remove unused columns. |
|  | | | |  | data$duration <- NULL |
|  | | | |  | data$sound.files <- NULL |
|  | | | |  | data$selec <- NULL |
|  | | | |  | data$peakf <- NULL |
|  | | | |  |  |
|  | | | |  | # Remove rows containing NA's. |
|  | | | |  | data <- data[complete.cases(data),] |
|  | | | |  |  |
|  | | | |  | # Write out csv dataset. |
|  | | | |  | write.csv(data, file='voice.csv', sep=',', row.names=F) |
|  | | | |  |  |
|  | | | |  | # Create a train and test set. |
|  | | | |  | set.seed(777) |
|  | | | |  | spl <- sample.split(data$label, 0.7) |
|  | | | |  | train <- subset(data, spl == TRUE) |
|  | | | |  | test <- subset(data, spl == FALSE) |
|  | | | |  |  |
|  | | | |  | # Build models. |
|  | | | |  | genderLog <- glm(label ~ ., data=train, family='binomial') |
|  | | | |  | genderCART <- rpart(label ~ ., data=train, method='class') |
|  | | | |  | prp(genderCART) |
|  | | | |  | genderForest <- randomForest(label ~ ., data=train) |
|  | | | |  |  |
|  | | | |  | # Assume a basline model of always predicting male. |
|  | | | |  | # Accuracy: 0.50 |
|  | | | |  | table(train$label) |
|  | | | |  | 1107 / nrow(train) |
|  | | | |  |  |
|  | | | |  | # Accuracy: 0.50 |
|  | | | |  | table(test$label) |
|  | | | |  | 475 / nrow(test) |
|  | | | |  |  |
|  | | | |  | # Accuracy: 0.72 |
|  | | | |  | predictLog <- predict(genderLog, type='response') |
|  | | | |  | table(train$label, predictLog >= 0.5) |
|  | | | |  | (814 + 777) / nrow(train) |
|  | | | |  |  |
|  | | | |  | # Accuracy: 0.71 |
|  | | | |  | predictLog2 <- predict(genderLog, newdata=test, type='response') |
|  | | | |  | table(test$label, predictLog2 >= 0.5) |
|  | | | |  | (339 + 335) / nrow(test) |
|  | | | |  |  |
|  | | | |  | # Accuracy: 0.81 |
|  | | | |  | predictCART <- predict(genderCART) |
|  | | | |  | predictCART.prob <- predictCART[,2] |
|  | | | |  | table(train$label, predictCART.prob >= 0.5) |
|  | | | |  | (858 + 941) / nrow(train) |
|  | | | |  |  |
|  | | | |  | # Accuracy: 0.78 |
|  | | | |  | predictCART2 <- predict(genderCART, newdata=test) |
|  | | | |  | predictCART2.prob <- predictCART2[,2] |
|  | | | |  | table(test$label, predictCART2.prob >= 0.5) |
|  | | | |  | (364 + 378) / nrow(test) |
|  | | | |  |  |
|  | | | |  | # Accuracy: 1 |
|  | | | |  | predictForest <- predict(genderForest, newdata=train) |
|  | | | |  | table(train$label, predictForest) |
|  | | | |  |  |
|  | | | |  | # Accuracy: 0.86 |
|  | | | |  | predictForest <- predict(genderForest, newdata=test) |
|  | | | |  | table(test$label, predictForest) |
|  | | | |  | (410 + 409) / nrow(test) |
|  | | | |  |  |
|  | | | |  | # Tune random-forest and return best model. |
|  | | | |  | # Accuracy: 0.87 |
|  | | | |  | set.seed(777) |
|  | | | |  | genderTunedForest <- tuneRF(train[, -21], train[, 21], stepFactor=.5, doBest=TRUE) |
|  | | | |  | predictForest <- predict(genderTunedForest, newdata=test) |
|  | | | |  | table(test$label, predictForest) |
|  | | | |  | (412 + 416) / nrow(test) |
|  | | | |  |  |
|  | | | |  | # Try a boosted tree model. |
|  | | | |  | # Accuracy: 0.91 |
|  | | | |  | set.seed(777) |
|  | | | |  | genderBoosted <- train(label ~ ., data=train, method='gbm') |
|  | | | |  | predictBoosted <- predict(genderBoosted, newdata=train) |
|  | | | |  | confusionMatrix(predictBoosted, train$label) |
|  | | | |  |  |
|  | | | |  | # Accuracy: 0.84 |
|  | | | |  | predictBoosted <- predict(genderBoosted, newdata=test) |
|  | | | |  | confusionMatrix(predictBoosted, test$label) |
|  | | | |  |  |
|  | | | |  | # Try XGBoost. |
|  | | | |  | # Accuracy: 1 |
|  | | | |  | trainx <- sapply(train, as.numeric) |
|  | | | |  | trainx[,21] <- trainx[,21] - 1 |
|  | | | |  | set.seed(777) |
|  | | | |  | genderXG <- xgboost(data = trainx[,-21], label = trainx[,21], eta=0.2, nround = 500, subsample = 0.5, colsample\_bytree = 0.5, objective = "binary:logistic") |
|  | | | |  | results <- predict(genderXG, trainx) |
|  | | | |  | table(trainx[,21], results >= 0.5) |
|  | | | |  |  |
|  | | | |  | # Accuracy: 0.87 |
|  | | | |  | testx <- sapply(test, as.numeric) |
|  | | | |  | testx[,21] <- testx[,21] - 1 |
|  | | | |  | results <- predict(genderXG, testx) |
|  | | | |  | table(testx[,21], results >= 0.5) |
|  | | | |  | (414 + 413) / nrow(test) |
|  | | | |  |  |
|  | | | |  | # trans <- processFolder('trans') |
|  | | | |  | # trans$label <- c(2, 2, 1, 1, 2, 2, 1, 1) |
|  | | | |  | # trans$label <- factor(trans$label, labels=c('male', 'female')) |
|  | | | |  | # tpred <- predict(genderLog, newdata=trans, type='response') |
|  | | | |  | # table(trans$label, tpred >= 0.5) |
|  | | | |  |  |
|  | | | |  | # tpred <- predict(genderForest, newdata=trans) |
|  | | | |  | # table(trans$label, tpred) |
|  | | | |  |  |
|  | | | |  | # tpred <- predict(genderBoosted, newdata=trans) |
|  | | | |  | # confusionMatrix(trans$label, tpred) |
|  | | | |  |  |
|  | | | |  | # tpred <- predict(genderCART, newdata=trans) |
|  | | | |  | # tpred.prob <- tpred[,2] |
|  | | | |  | # table(trans$label, tpred.prob >= 0.5) |
|  | | | |  |  |
|  | | | |  | # trans <- processFolder('trans2') |
|  | | | |  | # trans$label <- c(2, 2, 2, 1, 2) |
|  | | | |  | # trans$label <- factor(trans$label, labels=c('male', 'female')) |
|  | | | |  | # tpred <- predict(genderLog, newdata=trans, type='response') |
|  | | | |  | # table(trans$label, tpred >= 0.5) |
|  | | | |  |  |
|  | | | |  | # tpred <- predict(genderForest, newdata=trans) |
|  | | | |  | # table(trans$label, tpred) |
|  | | | |  |  |
|  | | | |  | # tpred <- predict(genderBoosted, newdata=trans) |
|  | | | |  | # confusionMatrix(trans$label, tpred) |
|  | | | |  |  |
|  | | | |  | # tpred <- predict(genderCART, newdata=trans) |
|  | | | |  | # tpred.prob <- tpred[,2] |
|  | | | |  | # table(trans$label, tpred.prob >= 0.5) |
|  | | | |  |  |
|  | | | |  | # trans <- processFolder('trans4') |
|  | | | |  | # trans$label <- c(2, 2, 2, 2, 1, 1, 1, 1) |
|  | | | |  | # trans$label <- factor(trans$label, labels=c('male', 'female')) |
|  | | | |  | # |
|  | | | |  | # tpred <- predict(genderLog, newdata=trans, type='response') |
|  | | | |  | # table(trans$label, tpred >= 0.5) |
|  | | | |  | # |
|  | | | |  | # tpred <- predict(genderTunedForest, newdata=trans, type='response') |
|  | | | |  | # table(trans$label, tpred) |
|  | | | |  | # |
|  | | | |  | # trans[,1:3] <- NULL |
|  | | | |  | # trans$peakf <- NULL |
|  | | | |  | # testx <- sapply(trans, as.numeric) |
|  | | | |  | # testx[,21] <- testx[,21] - 1 |
|  | | | |  | # results <- predict(genderXG, testx) |
|  | | | |  | # table(testx[,21], results >= 0.5) |