BATS OF BORNEO SEMI-AUTOMATED CLASSIFIER FOR ECHOLOCATION CALLS (2021)

By Natalie Yoh (https://github.com/TallyYoh; tallyyoh@gmail.com)

Cite as - Yoh, N., Kingston, T., McArthur, E., Aylen, O.E., Huang, J.C.C., Jinggong, E.R., Khan, F.A.A., Lee, B.P.Y.H., Mitchell, S.L.M., Bicknell, J.E., and Struebig, M.J. (2022). A machine learning framework to classify Southeast Asian echolocating bats, Ecological Indicators, 136. doi:10.1016/j.ecolind.2022.108696

This script applies the Borneo Bat Classifier (BBC) machine learning classifier to collated bat call parameter measurements from Borneo and assigns relevant labels

The output includes:

- Pulse measurements
- Predicted classification labels (call type/sonotype/species)
- Confidence of classification labels
- Script to locate files based on labels & confidence on your desktop & reorder them for manual verification

Software used to create classifier

- R v3.6.3
- Kaleidoscope v5.1.9g (Wildlife Acoustics, 2019)
- Adobe Audition v12.1.5 (Adobe Creative Cloud)

Contributors

- Tigga Kingston (Department of Biological Sciences at Texas Tech University and the Southeast Asian Bat Conservation Research Unit, Lubbock, Texas, United Stated of America)
- Joe Chun-Chia Huang (Taiwan Forestry Research Institute, Taipei, Taiwan)
- Ellen McArthur (Faculty of Resource Science and Technology, Universiti Malaysia Sarawak)
- Benjamin P.Y.-H. Lee (Wildlife Management Division, National Parks Board, Singapore)
- Faisal Ali Anwarali Khan (Faculty of Resource Science and Technology, Universiti Malaysia Sarawak)
- Emy Ritta Jinggong (Faculty of Resource Science and Technology, Universiti Malaysia Sarawak)
- Oliver E. Aylen (, Department of Zoology, University of Otago, Otago, New Zealand)
- Simon L. Mitchell (DICE, School of Anthropology and Conservation, University of Kent, Canterbury, United Kingdom)
- Jake Bicknell (DICE, School of Anthropology and Conservation, University of Kent, Canterbury, United Kingdom)
- Matthew Struebig (DICE, School of Anthropology and Conservation, University of Kent, Canterbury, United Kingdom)

Our aim is continuously test and update this tool as new reference data becomes available. Therefore, we would greatly appreciate users sharing any issues they find, particularly if this relates to species' IDs. Thank you!

Updates from v1.0:

- The call parameter data used in the v2.0 models is no longer scaled to ensure predictions are made on "true" numerical values
- The naming error for the FMqCF sonotypes has been corrected

1. PREPARE ENVIRONMENT

Set memory size for Jupyter/R kernels

```
memory.size()
memory.limit(size=56000)
```

Load packages

Specify where your package directory

```
Dir_packages    <-"C:/Users/Documents/R/win-library"
setwd(Dir_packages)
```

Load packages

```
library(bioacoustics)                          # For extracting call parameters
library(caret)                                 # For supervised machine learning
library(dplyr)                                 # For data manipulation/selection
library(gdata)                                 # For data manipulation/selection
library(pbapply)                               # For progress bar
```

Specify user directories for importing & exporting

!!! Before running, please ensure there is a back up copy of your raw files !!!

Script includes moving files directly in file location

- Dir_clean_files_WAV = File location for 5 second calls (including all subfolders)
- Dir_user_inputs = File location for csv inputs (e.g. threshold info)
- Dir_user_outputs = File location for data outputs
- Dir_classifier_models = File location for importing classifier models
- Dir_files_AutoID_WAV = File location for WAV files to be manually verified

```
Dir_clean_files_WAV        <-"F:/Data_wav5sec_clean/All_WAV"
Dir_user_inputs            <-"F:/R_inputs"
Dir_user_outputs           <-"F:/Data_wav5sec_IDs_auto/CSVexports"
Dir_classifier_models      <-"F:/R_Models"
Dir_files_AutoID_WAV       <-"F:/Data_wav5sec_IDs_auto"
```

2. LOAD MODELS

Set working directory to folder where models are stored

```
setwd(Dir_classifier_models)
```

Load models

Load stage 1 model to call type

```
model_S1_type <-  readRDS("model_Type_1000_v2.0.rds")
```

Load stage 2 model - to CF species

```
model_S2_CF   <-  readRDS("model_CF_1000_v2.0.rds")
```

Load stage 3 model - to FMqCF sonotype

```
model_S3_FMqCF <-  readRDS("model_FMqCF_2000_v2.0.rds")
```

Load threshold reference information

Set working directory & import csv

```
setwd(Dir_user_inputs)
Data_thres   <-read.csv("ThresholdValues.csv")
```

3. IMPORT & EXTRACT CALL PARAMETERS

Extracts call parameters from WAV files for classification using the Bioacoustic.R package

WAV files for import should first have been subset to 5 second fragments to quantify a bat pass & be filtered for noise in Kaliedoscope or other sound analysis software. See Yoh et al. (2021) for more information

Select file directories for where files are stored. This will perform extractions in two batches

```
files_P1        <- dir(Dir_clean_files_WAV, recursive = TRUE, full.names = TRUE, pattern = "[.]wav$")
```

Filter files for those identified as noise in Kaliedoscope

```
# convert to dataframe
files_P1        <-as.data.frame(files_P1)

# remove files listed as "noise"
files_P1_crop   <-as.character(files_P1[!grepl("NOISE", files_P1$files_P1),])
```

Detect & extract pulse measurements

Extractions conducted using the Bioacoustics.R package threshold function (https://rdrr.io/cran/bioacoustics/) Extractions can be performed for time expansion 1 or 10 as necessary (use "time_exp = 10" if necessary)

```
TDP1   <- setNames(
             pblapply(
                  files_P1_crop,
                  threshold_detection,
                  time_exp = 1,
                  threshold = 4,
                  SNR_thr = 4,
                  FFT_size = 512,), basename(files_P1_crop))
```

Collate measurements

Remove filenames where no values were extracted (e.g. only noise)

```
TDP1    <- TDP1[lapply(TDP1, function(x) length(x$data)) > 0]
```

Keep the extracted features and merge in a single data frame for further analysis

```
Data_WAV_raw  <- do.call("rbind", c(lapply(TDP1, function(x) x$data$event_data),
                                    list(stringsAsFactors = FALSE)))
```

Remove file extention from filenames

```
Data_WAV_raw$filename <-sub(pattern = "(.*)\\..*$", replacement = "\\1", basename(Data_WAV_raw$filename))
head(Data_WAV_raw)
```

Include filename location information

Extract filename from file locations

```
filename     <-sub(pattern = "(.*)\\..*$", replacement = "\\1", basename(files_P1_crop))
```

Create dataframe with full file location & filename

```
FileLoc      <-data.frame(FileLoc=totalfileloc, filename=filename)
```

Add to main dataframe

```
Data_WAV_raw <-merge(Data_WAV_raw, FileLoc, by="filename")
```

Clean & export pulse measurements

Rename columns - Include/remove additionals where applicable

```
colnames(Data_WAV_raw) <-c("Filename", "starting_time", "duration",  "freq_max_amp" , "freq_max",
                          "freq_min", "bandwidth",  "freq_start", "freq_center","freq_end",
                          "freq_knee", "fc","freq_bw_knee_fc", "bin_max_amp","pc_freq_max_amp",
                           "pc_freq_max", "pc_freq_min", "pc_knee", "temp_bw_knee_fc", "slope",
                           "kalman_slope", "curve_neg", "curve_pos_start", "curve_pos_end",
                           "mid_offset" ,"snr", "hd","smoothness", "FileLoc")
```

Export raw call parameters

```
setwd(Dir_user_outputs)
write.csv(Data_WAV_raw, file="Data_Callparameters_unclassified.csv", na = "NA")
```

Create row ID for tracking pulses

```
Data_WAV_raw$ID         <-as.vector(1:nrow(Data_WAV_raw))
```

Isolate call parameter data

Note - In previous versions, the call parameter data was scaled at this point. This step is no longer necessary and has been removed

```
Data_CallValues_Scaled  <-as.data.frame(subset(Data_WAV_raw,
                                         select = -c(Filename, FileLoc, starting_time, ID)))
```

Select row information

```
Data_RowInfo            <-subset(Data_WAV_raw, select = c(ID, Filename, FileLoc, starting_time))
```

Recombine

```
Data_WAV_scaled         <-droplevels(cbind(Data_RowInfo,Data_CallValues_Scaled))
```

4. PERFORM CLASSIFICATIONS - STAGE 1

Predict the call type of each file using the first machine learning model

Run predictions

Run prediction without confidence values

```
predictionsResultsType           <-predict(model_S1_type, Data_CallValues_Scaled)
```

Run prediction with confidence values

```
PredictionResultsTypeProb        <-predict(model_S1_type, Data_CallValues_Scaled, type = "prob")

PredictionResultsTypeProb$ID     <-as.vector(Data_WAV_raw$ID)

PredictionResultsTypeProb$ID     <-as.factor(as.character(PredictionResultsTypeProb$ID))
```

Combine predictions with confidence values

```
PredictionResultsTypeCombined    <-cbind(PredictionResultsTypeProb, predictionsResultsType)
```

Combine with file information

```
PredictionsFinalStage1           <-merge(PredictionResultsTypeCombined, Data_WAV_scaled, by="ID")
```

Export stage 1 predictions

```
setwd(Dir_user_outputs)
write.csv(PredictionsFinalStage1, file="Data_PredictionsStage1.csv", na = "NA")
```

Summarise results

Summarises the pulse predictions to call type identification to the file/bat pass level

Convert to factor for grouping

```
PredictionsFinalStage1$ID            <-as.factor(PredictionsFinalStage1$ID)
```

Create vectors for grouping columns

```
cols_sp    <- c("FM","CF","FMqCF", "QCF")
cols_ID    <- c("ID", "predictionsResultsType")
cols_Files <- c("Filename", "predictionsResultsType")
```

Isolate the confidence of the predicted species into new column

```
Temp_S1_Max_ID <- PredictionsFinalStage1 %>%
                         group_by(across(all_of(cols_ID))) %>%
                         mutate(MaxByID = max(c(FM, CF, FMqCF, QCF), na.rm = T))
```

Find the pulse of highest confidence within each file for each species

```
RES_S1_summary <- Temp_S1_Max_ID %>%
                         group_by(across(all_of(cols_Files))) %>%
                         summarise(MaxbyFile = max(MaxByID, na.rm = T))
```

Rename columns

```
names(RES_S1_summary)<-c("Filename","S1_Prediction","S1_Accuracy")
```

Isolate files for manual verification & create library

The following steps if for users who are only using the stage 1 classifications.

Skip to stage 2 (section 5) if you are using stage 2/3 classifications to sonotype/species

Selects WAV files which do not reach the necessary confidence threshold using their original filepathways and copies them into a new filepathway based on ID prediction & confidence threshold.

Determine which files need manual verification

Rename column levels to match

```
colnames(Data_thres) <- c("Prediction", "Threshold")
```

Merge confidence threshold information with the predictions data

```
RES_S1_summary        <- merge(RES_S1_summary, Data_thres, by = "Prediction", keep.all=TRUE)
```

Create threshold level column

```
RES_S1_summary$ThresLevel <-""
```

Ensure accuracy column is numeric

```
RES_S1_summary$Accuracy   <-as.numeric(RES_S1_summary$Accuracy)
```

Remove predictions below 60% confidence

```
RES_S1_summary <-RES_S1_summary[RES_S1_summary$Accuracy > 0.59,]
```

Loop to determine which files met the necessarily confidence threshold

```
for (y in 1:nrow(RES_S1_summary)){

  if((RES_S1_summary$Accuracy[y]*100) == RES_S1_summary$Threshold[y]) {
    RES_S1_summary$ThresLevel[y] <- "Met"
  }

  else if ((RES_S1_summary$Accuracy[y]*100) > RES_S1_summary$Threshold[y]) {
    RES_S1_summary$ThresLevel[y] <- "Met" }

  else if ((RES_S1_summary$Accuracy[y]*100) < RES_S1_summary$Threshold[y]) {
    RES_total_sum$ThresLevel[y] <- "Not Met" }
}
```

Filter data for files which did not meet the confidence threshold

```
DF_NotMet        <-filter(RES_S1_summary, (ThresLevel=="Not Met"))
```

Remove repeated files so a file is only manually checked once

```
DF_NotMet_unique <- DF_NotMet[!duplicated(DF_NotMet['filename']),]
```

Save data outputs

```
setwd(Dir_user_outputs)
write.csv(DF_NotMet_unique, file="DF_NotMet_unique.csv", na = "NA")
write.csv(RES_S1_summary,   file="Data_PredictionsSummary_max.csv", na = "NA")
```

!!! THE FOLLOWING CODE WILL MOVE FILES DIRECTORY ON YOUR COMPUTER !!!

!!! ENSURE IT IS WORKING CORRECTLY USING A TEST FILE/BACK UP YOUR DATA BEFORE PROCEEDING !!!

Specify ID levels

```
Lvls_stageType     <-levels(as.factor(RES_S1_summary$Prediction))
```

Not reversible: Loop to create new folder pathway and copy WAV files - user needs to update pathway below

```
for (S in 1:length(Lvls_stageType)){

  # Specify prediction level
  TYPE           <-Lvls_stageType[S]

  # Filter data for target species & confidence threshold
  RES_total_target  <-filter(DF_NotMet_unique_cleaned, Prediction ==TYPE)

  # Create filename vector including file locations
  Sp_file_list      <-as.character(RES_total_target$FileLoc)

  # Create output folder for ID level
  setwd(Dir_files_AutoID_WAV)
  newdir             <-paste0(TYPE,"_", "ThresholdNotMet")
  dir.create(newdir)

  # Create directory in R to Species specific folder
  Dir_temp           <-paste0("F:/Data_wav5sec_IDs_auto/",newdir)     # **** NEEDS UPDATING BY USER ****

  # Go back to input WAV files directory
  setwd(Dir_clean_files_WAV2012)

  # Move each individual file to new directory
  for (F in 1:length(Sp_file_list)){

    # Select file
    FILE     <-Sp_file_list[F]

    # copy files
    file.copy(FILE, Dir_temp)

    # Progress bar
    print(c("Loop", S, "from", length(Lvls_stageType), "File", F, "from", length(Sp_file_list)))

  }

}
```

------- End for users only classifying to call type -------

5. PERFORM CLASSIFICATIONS - STAGE 2

Split data based on the predictions from stage 1

Remove predictions below 60% confidence

```
Temp_S1_Max_ID <-Temp_S1_Max_ID[Temp_S1_Max_ID$MaxByID > 0.59,]
```

Divide into Type specific datasets based on predictions

```
Stage1_FM     <-Temp_S1_Max_ID[Temp_S1_Max_ID$predictionsResultsType=="FM", ]
Stage1_QCF    <-Temp_S1_Max_ID[Temp_S1_Max_ID$predictionsResultsType=="QCF", ]
Stage1_FMqCF  <-Temp_S1_Max_ID[Temp_S1_Max_ID$predictionsResultsType=="FMqCF", ]
Stage1_CF     <-Temp_S1_Max_ID[Temp_S1_Max_ID$predictionsResultsType=="CF", ]
```

For species which were identified as "CF" (constant-frequency) conduct a second classification stage using the second machine learning model which prioritises maximum frequency

Prepare CF data

Remove identifying information from CF data

```
Data_CallValues_CF_noID        <-Stage1_CF[,c("duration","freq_max_amp" , "freq_max",
                                 "freq_min", "bandwidth", "freq_start","freq_center",
                                 "freq_end", "freq_knee", "fc", "freq_bw_knee_fc",
                                 "bin_max_amp","pc_freq_max_amp", "pc_freq_max",
                                 "pc_freq_min","pc_knee", "temp_bw_knee_fc",
                                 "slope", "kalman_slope", "curve_neg", "curve_pos_start",
                                 "curve_pos_end","mid_offset" ,"snr", "hd",
                                 "smoothness")]
```

Filter for complete cases

```
Data_CallValues_CF_noID        <-Data_CallValues_CF_noID[complete.cases(Data_CallValues_CF_noID), ]
Data_CallValues_CF_noID        <-drop.levels(Data_CallValues_CF_noID)
```

Run predictions

Run prediction without confidence values

```
predictionsResultsCF            <-predict(model_S2_CF, Data_CallValues_CF_noID)
```

Run prediction without confidence

```
PredictionResultsProb_CF        <-predict(model_S2_CF, Data_CallValues_CF_noID, type = "prob")
PredictionResultsProb_CF$ID     <-Stage1_CF$ID
```

Combine predictions with confidence values

```
PredictionResultsCombined_CF    <-cbind(PredictionResultsProb_CF, predictionsResultsCF)
```

Combine with file information

```
PredictionsFinalStage2          <-merge(PredictionResultsCombined_CF, Stage1_CF, by="ID")
```

Export stage 2 predictions

```
setwd(Dir_user_outputs)
write.csv(PredictionsFinalStage2, file="Data_PredictionsStage2.csv", na = "NA")
```

Run confidence thresholds

Creates table to see which files meet the confidence thresholds necessary for file structure later

Convert ID to factor for grouping

```
PredictionsFinalStage2$ID            <-as.factor(PredictionsFinalStage2$ID)
```

Identify which CF are present in the data

```
Levels_CF                            <-levels(as.factor(PredictionsFinalStage2$predictionsResultsCF))
```

Create vector for grouping species (***user needs to edit depending on the species listed in Levels_CF***)

```
cols_sp    <- c( "CF_H140", "CF_Hate","CF_Hbic","CF_Hcer","CF_Hcox" ,
              "CF_Hdia","CF_Hgal",  "CF_Hlar",  "CF_Hrid",  "CF_Racu" , "CF_Raff",
              "CF_Rbor"  , "CF_Rcre","CF_Rluc" ,  "CF_Rphi", "CF_Rsed","CF_Rtri" )
```

Create vectors for grouping

```
cols_ID    <- c("ID", "predictionsResultsCF")
cols_Files <- c("Filename", "predictionsResultsCF")
```

Isolate the confidence of the predicted species into new column (***user needs to edit depending on the species listed in Levels_CF***)

```
Temp_S2_Max_ID <- PredictionsFinalStage2 %>%
                         group_by(across(all_of(cols_ID))) %>%
                         mutate(MaxByID = max(c(CF_H140, CF_Hate,CF_Hbic,CF_Hcer,CF_Hcox ,
                                         CF_Hdia,CF_Hgal,  CF_Hlar,  CF_Hrid, CF_Racu ,
                                         CF_Raff,CF_Rbor, CF_Rcre,CF_Rluc ,
                                         CF_Rphi, CF_Rsed,CF_Rtri), na.rm = T))
```

Find the pulse of highest confidence within each file for each species

```
RES_S2_summary <- Temp_S2_Max_ID %>%
                         group_by(across(all_of(cols_Files))) %>%
                         summarise(MaxbyFile = max(MaxByID, na.rm = T))
```

Rename columns

```
names(RES_S2_summary)<-c("Filename","S2_Prediction","S2_Accuracy")
```

6. PERFORM CLASSIFICATIONS - STAGE 3

For species which were identified as "FMqCF" (frequency modulated quasi-constant frequency) conduct a second classification stage using the third machine learning model which prioritises call shape

Prepare FMqCF data

Remove identifying information from FMqCF data

```
Data_CallValues_FMqCF_noID        <-Stage1_FMqCF[,c("duration","freq_max_amp" , "freq_max",
                                     "freq_min", "bandwidth","freq_start",
                                     "freq_center","freq_end", "freq_knee", "fc",
                                     "freq_bw_knee_fc", "bin_max_amp",  "pc_freq_max_amp",
                                     "pc_freq_max",  "pc_freq_min", "pc_knee",
                                     "temp_bw_knee_fc", "slope", "kalman_slope",
                                     "curve_neg", "curve_pos_start", "curve_pos_end",
                                     "mid_offset" ,"snr", "hd", "smoothness")]
```

Filter for complete cases

```
Data_CallValues_FMqCF_noID        <-Data_CallValues_FMqCF_noID[complete.cases(Data_CallValues_FMqCF_noID), ]
Data_CallValues_FMqCF_noID        <-drop.levels(Data_CallValues_FMqCF_noID)
```

Run predictions

Run predication without confidence

```
predictionsResultsFMqCF            <-predict(model_S3_FMqCF, Data_CallValues_FMqCF_noID)
```

Run predictions with confidence

```
PredictionResultsProb_FMqCF        <-predict(model_S3_FMqCF, Data_CallValues_FMqCF_noID, type = "prob")
PredictionResultsProb_FMqCF$ID     <-Stage1_FMqCF$ID
```

Combine predictions with confidence values

```
PredictionResultsCombined_FMqCF    <-cbind(PredictionResultsProb_FMqCF, predictionsResultsFMqCF)
```

Combine with file information

```
PredictionsFinalStage3          <-merge(PredictionResultsCombined_FMqCF, Stage1_FMqCF, by="ID")
```

Export stage 3 predictions

```
setwd(Dir_user_outputs)
write.csv(PredictionsFinalStage3, file="Data_PredictionsStage3.csv", na = "NA")
```

Run confidence thresholds

Creates table to see which files meet the confidence thresholds necessary for file structure later

Convert ID to factor for grouping

```
PredictionsFinalStage3$ID            <-as.factor(PredictionsFinalStage3$ID)
```

Identify which FMqCF are present in the data

```
Levels_FMqCF                         <-levels(as.factor(PredictionsFinalStage3$predictionsResultsFMqCF))
```

Create vector for grouping species (***user may need to edit depending on the species listed in Levels_FMqCF***)

```
cols_sp    <- c("FMqCF1" , "FMqCF2","FMqCF3","FMqCF4" ,"FMqCF5","LF"  , "LF_Acup")
```

Create vectors for grouping

```
cols_ID    <- c("ID", "predictionsResultsFMqCF")
cols_Files <- c("Filename", "predictionsResultsFMqCF")
```

Isolate the confidence of the predicted species into new column

```
Temp_S3_Max_ID <- PredictionsFinalStage3 %>%
                         group_by(across(all_of(cols_ID))) %>%
                         mutate(MaxByID = max(c(FMqCF1 , FMqCF2,FMqCF3,FMqCF4 ,
                                         FMqCF5,LF  , LF_Acup), na.rm = T))
```

Find the pulse of highest confidence within each file for each species

```
RES_S3_summary <- Temp_S3_Max_ID %>%
                         group_by(across(all_of(cols_Files))) %>%
                         summarise(MaxbyFile = max(MaxByID, na.rm = T))
```

Rename columns

```
names(RES_S3_summary)<-c("Filename","S3_Prediction","S3_Accuracy")
```

7. COMBINE FINAL PREDICTIONS

Collate predictions from each classification stage & isolate files for manual verification

Create summaries

Select relevant columns

```
SummaryQCF            <-Stage1_QCF[, c("ID","predictionsResultsType","MaxByID", "Filename", "FileLoc")]

SummaryFM             <-Stage1_FM[, c("ID","predictionsResultsType","MaxByID", "Filename", "FileLoc")]

SummaryCF             <-Temp_S2_Max_ID[, c("ID","predictionsResultsCF","MaxByID", "Filename", "FileLoc")]

SummaryFMqCF          <-Temp_S3_Max_ID[, c("ID","predictionsResultsFMqCF","MaxByID", "Filename", "FileLoc")]
```

Rename column names to match

```
colnames(SummaryQCF)      <-c("ID","Prediction","Accuracy","Filename", "FileLoc")
colnames(SummaryFM)       <-c("ID","Prediction","Accuracy","Filename", "FileLoc")
colnames(SummaryCF)       <-c("ID","Prediction","Accuracy","Filename", "FileLoc")
colnames(SummaryFMqCF)    <-c("ID","Prediction","Accuracy","Filename", "FileLoc")
```

Join predictions

```
RES_total_raw      <-rbind(SummaryQCF, SummaryFM, SummaryCF, SummaryFMqCF)
```

Subset again for files above 60% confidence

This will subset for files identified in stages 2 and 3 to sonotype/species

```
RES_total_raw      <-RES_total_raw[RES_total_raw$Accuracy>0.59,]
```

Export final predictions

```
setwd(Dir_user_outputs)
write.csv(RES_total_raw, file="Data_PredictionsSummary.csv", na = "NA")
```

Summarise results

Summarises the pulse predictions to sonotype/species identification to the file/bat pass level

Convert to factor for grouping

```
RES_total_raw$ID            <-as.factor(RES_total_raw$ID)
```

Create vectors for grouping columns

```
cols_sp    <- levels(as.factor(RES_total_raw$Prediction))
cols_Files <- c("Filename", "Prediction")
```

Find the pulse of highest confidence within each file for each species

```
RES_total_sum <- RES_total_raw %>%
                      group_by(across(all_of(cols_Files))) %>%
                      summarise(MaxbyFile = max(Accuracy, na.rm = T))
```

Rename columns

```
names(RES_total_sum)<-c("filename","Prediction","Accuracy")
```

Readd file location

```
RES_total_sum <-merge(RES_total_sum, FileLoc, by="filename", all= FALSE)
```

Isolate files for manual verification & create library

The following steps if for users who are using stage 2/3 classifications.

Selects WAV files which do not reach the necessary confidence threshold using their original filepathways and copies them into a new filepathway based on ID prediction & confidence threshold.

Determine which files need manual verification

Rename column levels to match

```
colnames(Data_thres) <- c("Prediction", "Threshold")
```

Merge confidence threshold information with the predictions data

```
RES_total_sum        <- merge(RES_total_sum, Data_thres, by = "Prediction", keep.all=TRUE)
```

Create threshold level column

```
RES_total_sum$ThresLevel <-""
```

Ensure accuracy column is numeric

```
RES_total_sum$Accuracy   <-as.numeric(RES_total_sum$Accuracy)
```

Remove predictions below 60% confidence

```
RES_total_sum <-RES_total_sum[RES_total_sum$Accuracy > 0.59,]
```

Loop to determine which files met the necessarily confidence threshold

```
for (y in 1:nrow(RES_total_sum)){

  if((RES_total_sum$Accuracy[y]*100) == RES_total_sum$Threshold[y]) {
    RES_total_sum$ThresLevel[y] <- "Met"
  }

  else if ((RES_total_sum$Accuracy[y]*100) > RES_total_sum$Threshold[y]) {
    RES_total_sum$ThresLevel[y] <- "Met" }

  else if ((RES_total_sum$Accuracy[y]*100) < RES_total_sum$Threshold[y]) {
    RES_total_sum$ThresLevel[y] <- "Not Met" }
}
```

Filter data for files which did not meet the confidence threshold

```
DF_NotMet        <-filter(RES_total_sum, (ThresLevel=="Not Met"))
```

Remove repeated files so a file is only manually checked once

```
DF_NotMet_unique <- DF_NotMet[!duplicated(DF_NotMet['filename']),]
```

Save data outputs

```
setwd(Dir_user_outputs)
write.csv(DF_NotMet_unique, file="DF_NotMet_unique.csv", na = "NA")
write.csv(RES_total_sum,   file="Data_PredictionsSummary_max.csv", na = "NA")
```

!!! THE FOLLOWING CODE WILL MOVE FILES DIRECTORY ON YOUR COMPUTER !!!

!!! ENSURE IT IS WORKING CORRECTLY USING A TEST FILE/BACK UP YOUR DATA BEFORE PROCEEDING !!!

Specify ID levels

```
Lvls_stageAll     <-levels(as.factor(RES_total_sum$Prediction))
```

Not reversible: Loop to create new folder pathway and copy WAV files - user needs to update pathway below

In [ ]:

```
for (S in 1:length(Lvls_stageAll)){

  # Specify prediction level
  SPECIES           <-Lvls_stageAll[S]

  # Filter data for target species & confidence threshold
  RES_total_target  <-filter(DF_NotMet_unique_cleaned, Prediction ==SPECIES)

  # Create filename vector including file locations
  Sp_file_list      <-as.character(RES_total_target$FileLoc)

  # Remove duplicates (shouldn't remove any values)
  Sp_file_list      <-Sp_file_list[!duplicated(Sp_file_list)]

  # Create output folder for ID level
  setwd(Dir_files_AutoID_WAV)
  newdir            <-paste0(SPECIES,"_", "ThresholdNotMet")
  dir.create(newdir)

  # Create directory in R to Species specific folder
  Dir_temp          <-paste0("F:/SAFE_Data_wav5sec_IDs_auto/",newdir)     # **** NEEDS UPDATING BY USER ****

  # Go back to input WAV files directory
  setwd(Dir_clean_files_WAV2012)

  # Move each individual file to new directory
  for (F in 1:length(Sp_file_list)){

    # Select file
    FILE     <-Sp_file_list[F]

    # copy files
    file.copy(FILE, Dir_temp)

    # Progress bar
    print(c("Loop", S, "from", length(Lvls_stageAll), "File", F, "from", length(Sp_file_list)))

  }
}
```

---- End for all users -----

Note - Recordings will differ depending on the recording equipment and conditions. Please ensure you manually check a subset of the calls that meet the confidence threshold to ensure the classifier works adequately on your data before analysing the data