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!

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_clean_files_WAVP1 = File location for 5 second calls (part 1 if WAV files across two subfolders*)
- Dir_clean_files_WAVP2 = File location for 5 second calls (part 2 if WAV files across two 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

*This provides the infrastructure to perform the pulse measurements in 2+ batches

```
Dir_clean_files_WAV        <-"F:/Data_wav5sec_clean/All_WAV"
Dir_clean_files_WAVP1      <-"F:/Data_wav5sec_clean/All_WAV/WAVP1"
Dir_clean_files_WAVP2      <-"F:/Data_wav5sec_clean/All_WAV/WAVP2"
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.rds")
```

Load stage 2 model - to CF species

```
model_S2_CF   <-  readRDS("model_CF_1000.rds")
```

Load stage 3 model - to FMqCF sonotype

```
model_S3_FMqCF <-  readRDS("model_FMqCF_2000.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_WAVP1, recursive = TRUE, full.names = TRUE, pattern = "[.]wav$")
files_P2        <- dir(Dir_clean_files_WAVP2, recursive = TRUE, full.names = TRUE, pattern = "[.]wav$")
```

Filter files for those identified as noise in Kaliedoscope

```
# convert to dataframe
files_P1        <-as.data.frame(files_P1)
files_P2        <-as.data.frame(files_P2)

# remove files listed as "noise"
files_P1_crop   <-as.character(files_P1[!grepl("NOISE", files_P1$files_P1),])
files_P2_crop   <-as.character(files_P2[!grepl("NOISE", files_P2$files_P2),])
```

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))
```

```
TDP2   <- setNames(
            pblapply(
                  files_P2_crop,
                  threshold_detection,
                  time_exp = 1,
                  threshold = 4,
                  SNR_thr = 4,
                  FFT_size = 512,), basename(files_P2_crop))
```

Collate measurements

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

```
TDP1    <- TDP1[lapply(TDP1, function(x) length(x$data)) >  ]
TDP2    <- TDP2[lapply(TDP2, function(x) length(x$data)) >  ]
```

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

```
Data_WAV_rawP1  <- do.call("rbind", c(lapply(TDP1, function(x) x$data$event_data),
                                     list(stringsAsFactors = FALSE)))
Data_WAV_rawP2  <- do.call("rbind", c(lapply(TDP2, function(x) x$data$event_data),
                                     list(stringsAsFactors = FALSE)))
```

Row bind data

```
Data_WAV_raw      <-rbind(Data_WAV_rawP1,Data_WAV_rawP2)
```

Remove file extention from filenames

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

Calculate number of audio events extracted

```
(Rawcalls         <-nrow(Data_WAV_raw))
```

Include filename location information

Combine file locations

```
totalfileloc <-c(files_P1_crop,files_P2_crop)
```

Extract filename only

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

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")
```

Scale call parameters

Create row ID for tracking pulses

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

Scale call parameter data

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

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")
```

Run confidence thresholds

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

Create vectors for loop

```
PredictionsFinalStage1$ID            <-as.factor(PredictionsFinalStage1$ID)
codes                                <-levels(droplevels(PredictionsFinalStage1$ID))
```

Create empty dataframe

```
RES_Class_firststage <-data.frame(matrix(ncol = 11, nrow =  ))
```

Loop to determine which thresholds the files achieved for call type

```
for (i in 1:length(codes)){

  # Select row no.
  code<-codes[i]

  # Select results/file
  step1<-PredictionsFinalStage1[PredictionsFinalStage1$ID==code,]

  # list species / file
  step1$predictionsResultsType  <-as.factor(step1$predictionsResultsType)
  sps                           <-c(levels(droplevels(step1$predictionsResultsType)))
  step1$predictionsResultsType  <-as.character(step1$predictionsResultsType)

  # Loop for different species within one recording
  for (y in 1:length(sps)){

    #select a single species
    sp<-sps[y]

    stepsps   <-step1[step1$predictionsResultsType==sp,]
    stepsp    <-select(stepsps,sp)
    maxprob   <-max(stepsp)

    # Specify confidence thresholds
    if (maxprob>=0.6)  R60<-as.character("done")
    if (maxprob<0.6)   R60<-as.character("check")
    if (maxprob>=0.65) R65<-as.character("done")
    if (maxprob<0.65)  R65<-as.character("check")
    if (maxprob>=0.7)  R70<-as.character("done")
    if (maxprob<0.7)   R70<-as.character("check")
    if (maxprob>=0.75) R75<-as.character("done")
    if (maxprob<0.75)  R75<-as.character("check")
    if (maxprob>=0.8)  R80<-as.character("done")
    if (maxprob<0.8)   R80<-as.character("check")
    if (maxprob>=0.85) R85<-as.character("done")
    if (maxprob<0.85)  R85<-as.character("check")
    if (maxprob>=0.9)  R90<-as.character("done")
    if (maxprob<0.9)   R90<-as.character("check")
    if (maxprob>=0.95) R95<-as.character("done")
    if (maxprob<0.95)  R95<-as.character("check")

    temp <-as.character(c(code, sp, maxprob, R60, R65, R70, R75, R80, R85, R90, R95))

    # Bind results to main table
    RES_Class_firststage  <-rbind(RES_Class_firststage,temp)
    RES_Class_firststage %>% mutate_if(is.factor, as.character) -> RES_Class_firststage

  }
  # Show progress
  print(c("Loop ", i, "from", length(codes)))
}
```

Rename columns

```
names(RES_Class_firststage)<-c("ID","Prediction","S1_Accuracy","S1_R60",
                               "S1_R65","S1_R70","S1_R75","S1_R80","S1_R85",
                               "S1_R90","S1_R95")
```

Re-add call parameter/file info

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

(i) Isolate files for manual verification & create library

The following series (i) if for users who are only using the stage 1 classifications.

Skip the following series (i) 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.

!!! 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 !!!

Summarise pulses to bat passes

Determines the highest accuracy recorded for each ID for each file to quantify bat activity

Create empty dataframe

```
RES_firststage_sum           <-data.frame(matrix(ncol=ncol(RES_Class_firststage),nrow=0))
```

Rename columns

```
colnames(RES_firststage_sum) <-colnames(RES_Class_firststage)
```

Create vector for filenames

```
AllFiles           <-levels(as.factor(RES_Class_firststage$Filename))
```

Summarise bat passes by the highest accuracy pulse per file per species

```
for (a in 1:length(AllFiles)){

  # Select file
    FileLabel      <-AllFiles[a]

  # Subset main data for file
    TempDFByfile   <-RES_Class_firststage[RES_Class_firststage$Filename==FileLabel,]

  # Create vector for the no. of species within said file
    TempSpVector   <-levels(as.factor(TempDFByfile$Prediction))

  # For each species within said file find the highest accuracy
    for (b in 1:length(TempSpVector)){

        # Select species
          TempSp        <-TempSpVector[b]

        # Subset filename for pulses of that species only
          TempDFSp      <-TempDFByfile[TempDFByfile$Prediction == TempSp,]

        # Find highest accuracy value obtained
          MaxAcc        <-which.max(TempDFSp$S1_Accuracy)

        # Select row for pulse of highest accuracy
          TempRow       <-TempDFSp[MaxAcc, ]

        # Rowbind into main datatable
          RES_firststage_sum <-rbind(RES_firststage_sum, TempRow)

    }
    print(c("Loop", a, "from", length(AllFiles)))
}
```

Select files for manual verification

(i) Add data threshold information

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

RES_firststage_sum2        <- merge(RES_firststage_sum, Data_thres, by = "Prediction", keep.all=TRUE)

RES_firststage_sum2$ThresLevel <-""

RES_firststage_sum2$S1_Accuracy <-as.numeric(RES_firststage_sum2$S1_Accuracy) # make sure column is numeric

for (y in 1:nrow(RES_firststage_sum2)){

  if((RES_firststage_sum2$S1_Accuracy[y]*100) == RES_firststage_sum2$Threshold[y]) {
    RES_firststage_sum2$ThresLevel[y] <- "Met"
  }

  else if ((RES_firststage_sum2$S1_Accuracy[y]*100) > RES_firststage_sum2$Threshold[y]) {
    RES_firststage_sum2$ThresLevel[y] <- "Met" }

  else if ((RES_firststage_sum2$S1_Accuracy[y]*100) < RES_firststage_sum2$Threshold[y]) {
    RES_firststage_sum2$ThresLevel[y] <- "Not Met" }
}
```

(ii) Specify ID levels

```
Lvls_stage1  <-levels(as.factor(RES_firststage_sum2$Prediction))
```

(iii) Remove duplicates

```
# Filter data for confidence threshold
DF_NotMet        <-filter(RES_firststage_sum2, (ThresLevel=="Not Met"))

# Remove repeated files so a file is only manually checked once
DF_NotMet_unique <- DF_NotMet[!duplicated(DF_NotMet['Filename']),]
```

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

```
for (S in 1:length(Lvls_stage1)){

  # Specify prediction level
  TYPE           <-Lvls_stage1[S]

  # Filter data for target call type & confidence threshold
  RES_firststage_target  <-filter(DF_NotMet_unique, (ThresLevel=="Not Met") & (Prediction ==TYPE))

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

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

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

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

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

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

    # Select file
    FILE     <-Ty_file_list[F]

    # copy files
    file.copy(FILE, Dir_temp)

  }
  # Progress bar
  print(c("Loop", S, "from", length(Lvls_stage1)))

}
```

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

Split data based on call type predictions for stage 2

Divide into Type specific datasets based on predictions

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

5. PERFORM CLASSIFICATIONS - STAGE 2

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

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")]

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 with confidence values

```
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

Create vectors for loop

```
PredictionsFinalStage2$ID            <-as.factor(PredictionsFinalStage2$ID)
codes                                <-levels(droplevels(PredictionsFinalStage2$ID))
```

Create empty dataframe

```
RES_Class_secondstage <-data.frame(matrix(ncol = 11, nrow =  ))
```

Loop to determine which thresholds the files achieved for CF species

```
for (i in 1:length(codes)){

  # Select row no.
  code<-codes[i]

  # Select results/file
  step1<-PredictionsFinalStage2[PredictionsFinalStage2$ID==code,]

  # list species / file
  step1$predictionsResultsCF  <-as.factor(step1$predictionsResultsCF)
  sps                         <-c(levels(droplevels(step1$predictionsResultsCF)))
  step1$predictionsResultsCF  <-as.character(step1$predictionsResultsCF)

  # Loop for different species within one recording
  for (y in 1:length(sps)){

    #select a single species
    sp<-sps[y]

    stepsps   <-step1[step1$predictionsResultsCF==sp,]
    stepsp    <-select(stepsps,sp)
    maxprob   <-max(stepsp)

    # Specify confidence thresholds
    if (maxprob>=0.6)  R60<-as.character("done")
    if (maxprob<0.6)   R60<-as.character("check")
    if (maxprob>=0.65) R65<-as.character("done")
    if (maxprob<0.65)  R65<-as.character("check")
    if (maxprob>=0.7)  R70<-as.character("done")
    if (maxprob<0.7)   R70<-as.character("check")
    if (maxprob>=0.75) R75<-as.character("done")
    if (maxprob<0.75)  R75<-as.character("check")
    if (maxprob>=0.8)  R80<-as.character("done")
    if (maxprob<0.8)   R80<-as.character("check")
    if (maxprob>=0.85) R85<-as.character("done")
    if (maxprob<0.85)  R85<-as.character("check")
    if (maxprob>=0.9)  R90<-as.character("done")
    if (maxprob<0.9)   R90<-as.character("check")
    if (maxprob>=0.95) R95<-as.character("done")
    if (maxprob<0.95)  R95<-as.character("check")

    temp <-as.character(c(code, sp, maxprob, R60, R65, R70, R75, R80, R85, R90, R95))

    # Bind results to main table
    RES_Class_secondstage  <-rbind(RES_Class_secondstage,temp)
    RES_Class_secondstage %>% mutate_if(is.factor, as.character) -> RES_Class_secondstage

  }
  # Show progress
  print(c("Loop ", i, "from", length(codes)))
}
```

Rename columns

```
names(RES_Class_secondstage)<-c("ID","S2_Prediction","S2_Accuracy","S2_R60",
                                "S2_R65","S2_R70","S2_R75","S2_R80","S2_R85",
                                "S2_R90","S2_R95")
```

Re-add call parameter/file info

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

5. 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

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")]

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 prediction without confidence values

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

Run predictions with confidences

```
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

Create vectors for loop

```
PredictionsFinalStage3$ID            <-as.factor(PredictionsFinalStage3$ID)
codes                                <-levels(droplevels(PredictionsFinalStage3$ID))
```

Create empty dataframe

```
RES_Class_thirdstage <-data.frame(matrix(ncol = 11, nrow =  ))
```

Loop to determine which thresholds the files achieved for FMCF sonotypes/species

```
for (i in 1:length(codes)){

  # Select row no.
  code<-codes[i]

  # Select results/file
  step1<-PredictionsFinalStage3[PredictionsFinalStage3$ID==code,]

  # list species / file
  step1$predictionsResultsFMqCF  <-as.factor(step1$predictionsResultsFMqCF)
  sps                            <-c(levels(droplevels(step1$predictionsResultsFMqCF)))
  step1$predictionsResultsFMqCF  <-as.character(step1$predictionsResultsFMqCF)

  # Loop for different species within one recording
  for (y in 1:length(sps)){

    #select a single species
    sp<-sps[y]

    stepsps   <-step1[step1$predictionsResultsFMqCF==sp,]
    stepsp    <-select(stepsps,sp)
    maxprob   <-max(stepsp)

    # Specify confidence thresholds
    if (maxprob>=0.6)  R60<-as.character("done")
    if (maxprob<0.6)   R60<-as.character("check")
    if (maxprob>=0.65) R65<-as.character("done")
    if (maxprob<0.65)  R65<-as.character("check")
    if (maxprob>=0.7)  R70<-as.character("done")
    if (maxprob<0.7)   R70<-as.character("check")
    if (maxprob>=0.75) R75<-as.character("done")
    if (maxprob<0.75)  R75<-as.character("check")
    if (maxprob>=0.8)  R80<-as.character("done")
    if (maxprob<0.8)   R80<-as.character("check")
    if (maxprob>=0.85) R85<-as.character("done")
    if (maxprob<0.85)  R85<-as.character("check")
    if (maxprob>=0.9)  R90<-as.character("done")
    if (maxprob<0.9)   R90<-as.character("check")
    if (maxprob>=0.95) R95<-as.character("done")
    if (maxprob<0.95)  R95<-as.character("check")

    temp <-as.character(c(code, sp, maxprob, R60, R65, R70, R75, R80, R85, R90, R95))

    # Bind results to main table
    RES_Class_thirdstage  <-rbind(RES_Class_thirdstage,temp)
    RES_Class_thirdstage %>% mutate_if(is.factor, as.character) -> RES_Class_thirdstage

  }
  # Show progress
  print(c("Loop ", i, "from", length(codes)))
}
```

Rename columns

```
names(RES_Class_thirdstage)<-c("ID","S3_Prediction", "S3_Accuracy","S3_R60","S3_R65",
                               "S3_R70","S3_R75","S3_R80","S3_R85","S3_R90","S3_R95")
```

Re-add call parameter/file info

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

6. COMBINE FINAL PREDICTIONS

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

Create summaries

Select relevant columns

```
SummaryQCF              <-Stage1_QCF[, c("ID","Prediction","S1_Accuracy","S1_R60",
                                         "S1_R65","S1_R70","S1_R75","S1_R80","S1_R85",
                                         "S1_R90","S1_R95", "Filename", "FileLoc")]

SummaryFM               <-Stage1_FM[, c("ID","Prediction","S1_Accuracy","S1_R60",
                                        "S1_R65","S1_R70","S1_R75","S1_R80","S1_R85",
                                        "S1_R90","S1_R95", "Filename", "FileLoc")]

SummaryCF               <-RES_Class_secondstage[, c("ID","S2_Prediction","S2_Accuracy",
                                                    "S2_R60","S2_R65","S2_R70","S2_R75","S2_R80",
                                                    "S2_R85", "S2_R90","S2_R95", "Filename", "FileLoc")]

SummaryFMqCF            <-RES_Class_thirdstage[, c("ID","S3_Prediction","S3_Accuracy",
                                                   "S3_R60","S3_R65","S3_R70","S3_R75","S3_R80",
                                                   "S3_R85", "S3_R90","S3_R95", "Filename", "FileLoc")]
```

Rename columns to match

```
colnames(SummaryQCF)      <-c("ID","Prediction","Accuracy","R60","R65","R70","R75","R80",
                              "R85","R90","R95", "Filename", "FileLoc")
colnames(SummaryFM)       <-c("ID","Prediction","Accuracy","R60","R65","R70","R75","R80",
                              "R85","R90","R95", "Filename", "FileLoc")
colnames(SummaryCF)       <-c("ID","Prediction","Accuracy","R60","R65","R70","R75","R80",
                              "R85","R90","R95", "Filename", "FileLoc")
colnames(SummaryFMqCF)    <-c("ID","Prediction","Accuracy","R60","R65","R70","R75","R80",
                              "R85","R90","R95", "Filename", "FileLoc")
```

Combine tables

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

Subset for files above 60% confidence

```
RES_total_raw      <-RES_total_raw[RES_total_raw$R60=="done",]
```

Export complete predictions

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

Summarise pulses to bat passes

Determines the highest accuracy recorded for each ID for each file to quantify bat activity

Convert ID 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 information

```
head(FileLoc)
```

Add to main dataframe

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

Export summary predictions

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

(ii) Isolate files for manual verification & create library

The following series (ii) if for users who use all three classification stages. If you used series (i) the following series will overwrite those files

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.

!!! 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 !!!

(i) Merge with threshold information

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

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

RES_total_sum$ThresLevel <-""

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

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" }
}
```

(ii) Specify ID levels

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

(iii) Remove duplicate levels

```
# Filter data for 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']),]
```

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

```
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, 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:/Data_wav5sec_IDs_auto/",newdir)      # **** NEEDS UPDATING BY USER ****

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

  # 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)))

}
```

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