

Standard Operating Procedure #6:

Data Management by the Protocol Lead

Version 1.0

In Bat monitoring protocol for the Great Lakes Inventory and Monitoring Network: Version 1.0

Katy Goodwin, Rebecca Key, and Al Kirschbaum

National Park Service
Great Lakes Inventory and Monitoring Network
2800 Lakeshore Drive East
Ashland, Wisconsin 54806

Month 2020

Please cite this publication as:

Goodwin, K., R. Key, and A. Kirschbaum. 2020. Standard operating procedure #6: Data management by the protocol lead (version 1.0). *In* K. R. Goodwin. 2020. Bat monitoring protocol for the Great Lakes Inventory and Monitoring Network: Version 1.0. Natural Resource Report NPS/GLKN/NRR—2020/XXXX. National Park Service, Fort Collins, Colorado.

Revision History Log

The following table lists all edits and amendments to this document since the original publication date. Information entered in the log must be complete and concise. Users of this standard operating procedure will promptly notify the GLKN protocol lead and/or project coordinator about recommended and required changes. The protocol lead and/or project coordinator must review and incorporate all changes, complete the revision history log, and change the date and version number on the title page and in the header of the document file.

Previous Version #	Revision Date	Author (with title and affiliation)	Location in Document and Concise Description of Revision	Reason for Change	New Version #
Add rows as needed for each change or set of changes tied to an updated version number					

Contents

	Page
Figures.....	ii
1.0 Introduction.....	1
1.1 Data Storage.....	1
1.2 Folder Structure	3
1.3 Naming Conventions	5
1.4 Data Review During the Field Season	5
Datasheet	5
Photos	6
Program File	6
Summary File	6
WAV Files.....	6
Tracking.....	7
1.5 Data Review After the Field Season.....	9
1.6 Data Transfers	10
Appendix A. R Script to Rename Files.....	11
Appendix B. R Script to Compare the Number of Files in Database and On Hard Drive.....	15
Appendix C. R Script to Compare the Number of Files on Network Drive and Hard Drive	17
Appendix D. R Script to Compare the Modification Times and Sizes of Files on the Network Drive and Hard Drive.....	23

Figures

	Page
Figure 1. Example of required folder structure to organize data by year, park, and site.	3

1.0 Introduction

The Great Lakes Inventory and Monitoring Network (GLKN) is implementing a monitoring program to document changes in bat populations in national park units in the upper Midwest. We are using acoustic monitoring as an index for assessing long-term bat population trends in response to ecosystem changes including climate, habitat, and disease such as white-nose syndrome (WNS). The goals of this monitoring program are to assess probable presence and absence of bat species, and to determine relative activity levels of those species for each national park unit over time. This Standard Operating Procedure (SOP) describes the steps to be taken by the GLKN protocol lead to manage the data collected by field staff at each participating park. This includes guidelines for data storage, organization, and quality assurance. The goal of this SOP is to increase the quality and consistency of data through proper data review and management.

1.1 Data Storage

This project will generate enormous quantities of data due to the large number of sampling sites and the inherent large size of the WAV audio files. Properly storing and backing up these data is a critical component of the protocol lead's job, with guidance and assistance from the GLKN IT Specialist and Data Managers.

Park-based field staff are responsible for downloading raw data from the SD cards used in the acoustic detectors and saving the WAV files, program files, summary files, and dump files. In addition, they will save site photos and a scanned copy of the completed field datasheet for each deployment. Park staff will organize their raw data into folders by park and site and save it in two separate locations. One copy will be on the GLKN-provided external hard drive. The second copy will be a location selected by the park. For example, this may be another external hard drive or the park's network drive. Data management guidance for park staff is found in SOP #5.

At the conclusion of the field season, each park will send their data to the GLKN office on the GLKN-provided external drive. The protocol lead will immediately create two copies of the raw data received from the parks on two separate internal hard drives. It is suggested to use high capacity drives (e.g., 4- or 8-TB Seagate Desktop HDD) so that all parks' raw data can be stored together on the same drive. After copies are made, the individual drives sent in by the parks will be set aside as backups of the raw data. These drives will be stored at the GLKN office until the following spring when they are cleared and returned to the parks. One of the high capacity drives will also be set aside as a second backup copy of the raw data. The remaining high capacity drive will be used as the working copy for file organization, review, and corrections.

Once data have been reviewed and all corrections have been made on the working drive, the protocol lead will create a second copy on another high capacity internal hard drive. One copy of the finalized, corrected data will be stored offsite at Apostle Islands National Lakeshore and one copy will be retained at the GLKN office. After this is achieved, the original backup of the raw data may be deleted.

Data processing and corrections will take place during the fall and winter months. Once spring arrives, the protocol lead must prepare each park's external drive for the next field season by deleting the old raw data, reformatting the drive, and finally adding a complete copy of the reviewed and corrected dataset. The drives will then be mailed or delivered back to each park. This system provides park staff with the most up-to-date and correct information on which to base their continuing field work and allows us to maintain three separate copies of the data: one at the park, one at the GLKN office, and one in offsite storage at Apostle Islands National Lakeshore.

1.2 Folder Structure

To main consistency and better data organization, the folder structure used by the GLKN protocol lead should be equivalent to the folder structure used by individual parks (see SOP #5). Data will be organized by year, park, and site. Each site folder will contain the program file, summary files, dump files (if any), datasheet, and photos for that site, as well as a separate subfolder containing the WAV audio files (Figure 1). Use the same folder structure in all versions of the data (raw, working, and corrected).

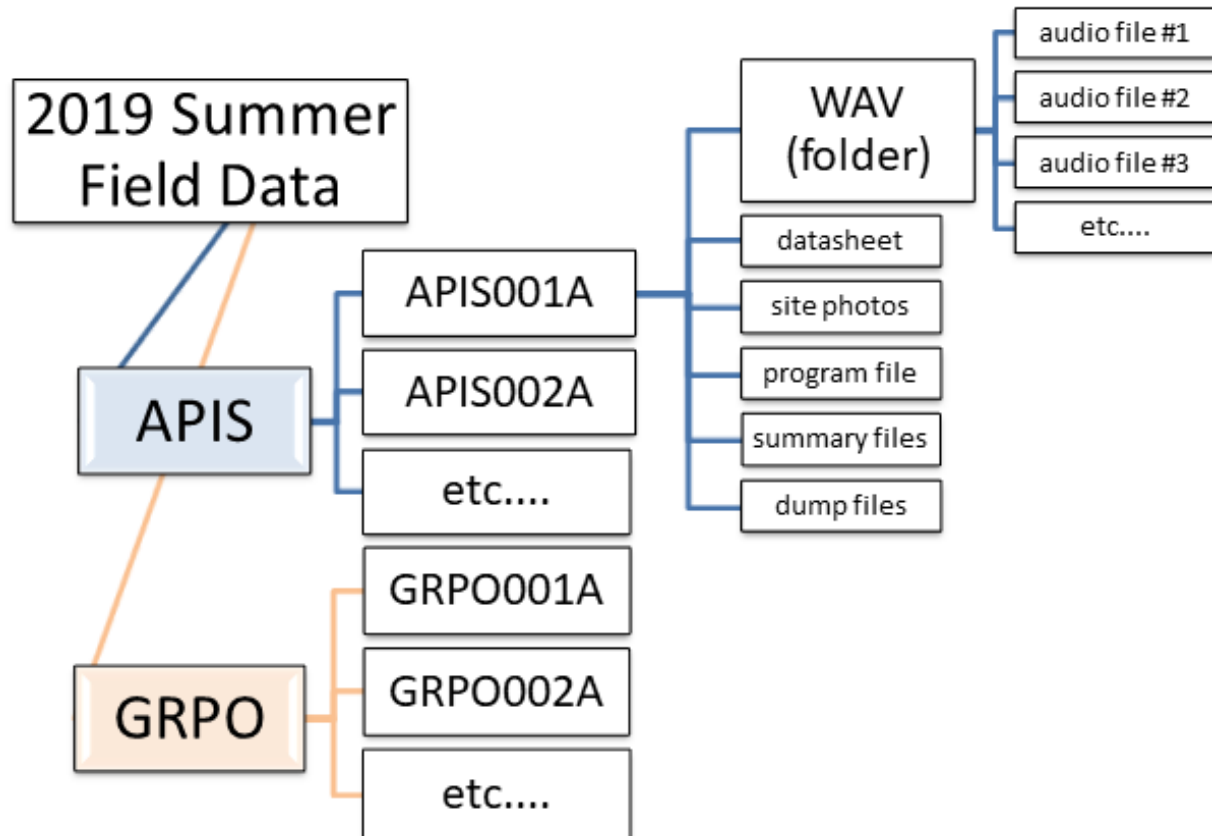


Figure 1. Example of required folder structure to organize data by year, park, and site.

1.3 Naming Conventions

Naming conventions for folders are described below. Note that the folders containing audio files will always be named simply “WAV” with no site-specific indicator in the folder name. Naming conventions for individual files must follow the guidelines given in SOP #5. Refer to SOP #1 for more details on Site IDs (i.e., use of A/B at end).

- **Folder for Year:** Year plus “Summer Field Data” (Example: 2019 Summer Field Data)
- **Folder for Park:** Standard 4-letter park code (Example: APIS, GRPO)
- **Folder for Site:** Site ID composed of 4-letter park code, cell’s GRTS order number, and letter indicating order of site establishment (Example: APIS001A)
- **Folder for Audio Files:** WAV; or if multiple deployments WAV_deploy1, WAV_deploy2

1.4 Data Review During the Field Season

In order to check the functionality of acoustic detectors and adherence to protocols, park field staff are required to upload their data to the project’s shared online drive throughout the field season. This includes datasheets, photos, program files, summary files, dump files and a small sample of recorded WAV files (five per site). The GLKN protocol lead is then responsible for downloading these data, reviewing, performing quality assurance checks, and providing feedback to park staff if changes are needed.

On the shared online drive there is a folder labeled “Upload Data Here”, which contains subfolders for each park and each sampling site. Park staff will upload their data to the appropriate folders as each deployment is completed. To begin reviewing data for a particular site, the GLKN protocol lead should first locate the site folder on the shared drive and download it as a zipped folder. Save the zipped folder within the appropriate Year and Park folders on the GLKN network drive at M:\Public > Monitoring > Bats > Data > Field Data > YEAR Summer Field Data > Park. Then extract the files and review using the following step-by-step guidelines.

Datasheet

1. Completely filled out?
2. Correct file naming convention?
3. Compare latitude/longitude to previous years.
4. Compare latitude/longitude written on datasheet to latitude/longitude programmed into Song Meter (by opening a WAV file in the SM3 or SM4 Configurator Utility).
5. Compare microphone orientation/height to previous years.

Photos

1. All four directional photos, plus setup, datasheet, and mic aspect photos?
2. Correct file naming convention?
3. Compare to previous years' photos to see if it looks like the same location.
4. Compare to sketch on datasheet to verify that orientation of sketch and labeling of photos matches. For example, if sketch shows mic pointing south towards a trail, is the trail visible in the mic aspect and south photos?
5. Compare to habitat indicated on datasheet.

Program File

1. Correct file naming convention?
2. Verify correct recording schedule was used (refer to SOP #3).
3. Verify correct audio settings were used (refer to SOP #3).
4. Verify site prefix (Site ID) was programmed correctly.

Summary File

1. Correct file naming convention?
2. Compare start and end date/time to dates indicated on datasheet.
3. Verify correct recording schedule from 18:00:00 to 08:00:00 each night.
4. Verify mic was connected throughout entire deployment.
5. Verify starting battery voltage was ~ 6.4 V, indicating fresh batteries.

WAV Files

1. Open one WAV file in SM3 or SM4 Configurator Utility and check the following:
 - a. Were proper audio settings and recording schedule used? Refer to SOP #3.
 - b. Verify site prefix (Site ID) was programmed correctly.
 - c. Compare latitude/longitude to those recorded on datasheet.
 - d. Was "delay start" option used and does this match notes on datasheet?
2. Open each WAV file in Kaleidoscope Pro and look for the following:
 - a. Are bat call sequences visible, or is it bird/insect/noise? If there are bat call sequences visible in at least one of the sample WAV files, you can assume that the microphone was functioning properly. If all five sample WAV files show noise, there may be a

problem. Contact park to obtain another 5-10 WAV files for review, from a variety of dates and times.

- b. Was the latest version of firmware used?
- c. Compare Song Meter serial number to number recorded on datasheet.

Tracking

The GLKN protocol lead should keep track which deployments have been reviewed and what issues arose. Data review spreadsheets for each year can be found at M:\Public > Monitoring > Bats > Admin > Protocol Lead > Data Review Process > ParkDataReview_YEAR.xlsx. The spreadsheet contains one tab for each park, with all sites listed. During the review process, note any missing items or discrepancies. Minor issues such as incorrectly named photos may be fixed by the protocol lead. Other problems or questions should be resolved by contacting the park staff. Let park staff know immediately if recording schedule or audio settings are wrong, so they can correct them prior to redeploying the unit.

1.5 Data Review After the Field Season

As mentioned previously, park staff will submit their data to the GLKN office at the conclusion of the field season. The GLKN protocol lead is responsible for ensuring all data are reviewed for completeness and accuracy, organized appropriately, and backed up securely.

Upon receipt of data from parks, the first step will be to back up this raw data. Refer to Section 1.1 for data backup and storage procedures.

Once raw data has been properly backed up, the protocol lead can begin reviewing and correcting the data. Use ONLY the high capacity drive that has been designated the “Working Copy”. Many potential problems should have already been resolved by reviewing incoming data throughout the field season but there may be some lingering issues. Contact park staff as soon as possible to obtain missing items and resolve questions. Timeliness is critical since many field staff are seasonal employees. Check for the following:

1. Have data been received from every park?
2. Are data for every sample site included?
3. If a sample site is not included, has the reason for omitting it been documented? Parks should submit a datasheet for omitted sites, filling in simply the Site ID, site name, observer, and a justification for why omitted.
4. Does every site have at minimum a scanned datasheet, site photos, program file, summary file, and WAV files? (Some sites may also have extra photos, multiple summary files, dump files, or other text files containing deployment notes.)
5. Is the datasheet completely filled out? If needed, datasheet additions/corrections can be made either manually by printing, marking up, and rescanning the form; or digitally using a PDF editing program such as Adobe Pro.
6. Are files organized correctly? Refer to the example folder structure in Figure 1 above. Data should be organized by year, then by park, then by site, with each site’s WAV files in a folder separate from other items.
7. Are files named correctly? Refer to file naming conventions in SOP #5.
8. If the site prefix was not programmed correctly, yielding an entire batch of misnamed files, these can be renamed all at once using an R script. Because of the risk of a botched or failed output, it is critical to make sure all data is backed up securely prior to running the R renaming script. It is a good idea to make a copy of a small batch of test data, and run that to ensure everything is working correctly before running the full data set. See Appendix A for instructions on running the R script to rename files.

Once all data have been reviewed for completeness and accuracy, organized into the proper folder structure, and all corrections made, the next step is entering data into the NPS Bats Acoustic Survey Database. Refer to SOP #7 for details of the data entry and QAQC procedures. Note that this process may uncover additional corrections that need to be made.

Finally, before sending the data to backup storage or to the contractor to be analyzed, there are three more checks to perform.

1. Compare the number of WAV files that are on the corrected drive to the number of WAV files entered in the database for the deployment. See Appendix B for instructions on running the R script to perform this check.
2. Compare the number of files that are located on the M drive to the number of files on the corrected drive for each deployment. See Appendix C for instructions on running the R script to perform this check.
3. Compare the sizes and modification dates of the files on the M drive to the files on the corrected drive. See Appendix D for instructions on running the R script to perform this check.

Once all checks and corrections have been made, it is critical to back up the final version of the clean, corrected data in order to have two copies in two separate locations. Refer to Section 1.1 for data backup and storage procedures.

1.6 Data Transfers

Data collected under this protocol are the property of the National Park Service and are categorized as “Sensitive but Unclassified” due to the inclusion of information pertaining to threatened and endangered species. Federally listed bat species that are potentially present at GLKN parks include the Northern Myotis (*Myotis septentrionalis*) and the Indiana Myotis (*Myotis sodalis*). Prior to sharing data with cooperators, contractors, or the public, the data must be entered into the database, quality assurance checks must be conducted, and the data to be shared must be certified. In other words, ALL data in the database do not need to be certified for a portion of it to be shared, as long as the data set to be shared has been fully QAQC checked and certified. Consult with the GLKN Data Managers to verify that QAQC and certification is properly completed.

When sharing data (including sending it to a contractor) it is important to keep track of what was sent, to whom, and when. Record this information for each data transfer, both incoming and outgoing, on the tracking spreadsheet located at M:\Public > Monitoring > Bats > Data > Data_Transfers_Tracking_List.xlsx.

Appendix A. R Script to Rename Files

We use the program R to efficiently rename large numbers of files.

Make sure the programs R and RStudio are installed on your computer. We will work in RStudio but R is also required because it will run in the background. Open RStudio and open the script to rename wav files for one park at a time (rename_wav_one_park_v09.R). Or, if you're renaming wav files from PIRO, open the script named 'rename_wav_piro_v04.R'. A separate script needed to be created for the files from PIRO because this park uses different acoustic loggers and the filenames are formatted differently. These scripts contain code that will search your computer for the R packages needed for these scripts and will automatically download and install them if they are needed.

Next, double check your folder and file structure (see Section 1.2). This is very important because the R script pulls information from the folder structure, specifically from folder names and folder levels. If not set up correctly from the beginning, either the script will encounter an error and stop running, or the resulting file names will be wrong. There are two rules to follow to avoid encountering naming problems. First, you must run the script on data that is stored on an external drive. Second, you must use this exact folder structure (drive letter and year may be different).

DRIVE:/YEAR Bat Acoustic Data/PARK/SITE folder/WAV folder/WAV files

For example, F:/2018 Bat Acoustic Data/GRPO/GRPO001A/WAV/WAV files

The R scripts we will use are located at M:\Public > Monitoring > Bats > R. There are multiple scripts depending on whether you want to rename files for PIRO or for a park other than PIRO.

Double click the file to open it in RStudio. It will appear in the upper left quarter of the screen, where you can see a list of commands. Text preceded by a # are simply notes to help explain what the program is doing or provide examples. Here are the two most current versions of rename scripts; one for PIRO and another for all other parks. While each of the scripts run, you will see any error messages in the lower left quarter. Once the program has completed, return to your data and check that it looks correct.

R script for renaming wav files for all parks *except* PIRO.

Change the line that contains the code for 'folder = ...', then select all code and press 'Run'.

```
m:\Monitoring\Bats\R\rename_wav_one_park_v09.R
# PURPOSE: This program renames .wav files for one park at a time, using the name of the site
folder.
list.of.packages <- c("stringr")
new.pkgs <- list.of.pkgs[!(list.of.pkgs %in% installed.packages()[,"Package"])]
if(length(new.pkgs)) install.packages(new.pkgs, repos='http://cran.rstudio.com/')
library(stringr)
# Put the letter of the drive and park code here. This defines the term "folder" as the drive and
park-specific folder.
folder = "F:/2017 Bat Acoustic Data/GRPO"
# This will create a list of all files within "folder" that end in .wav
string <- list.files(folder,pattern = "*.wav", recursive = TRUE, full.names = TRUE)
# This will run a loop renaming each item (i.e. each .wav file) in the list
for (i in string) {
```

```

# For example, one item in the list is "G:GRPO/GRPO004A/WAV/GRPO003__0__20170621_011041.wav"
# "a" Splits the file name and takes just the latter portion of the name
"__0__20170621_011041.wav"
a <- gsub("^.*? ", "", i)
# "b" Takes the label of the folder containing the file "GRPO004A"
b <- strsplit(i, "/")[[1]][4]
# "d" Pastes the two pieces together "GRPO004A__0__20170621_011041.wav"
d <- paste0(b,a)
# "e" Takes the file directory portion of the name "G:GRPO/GRPO004A/WAV/"
e <- str_sub(i,start=1,end = 48)
# "f" Pastes it back onto the front of the file name
"G:GRPO/GRPO004A/WAV/GRPO004A__0__20170621_011041.wav"
f <- paste0(e,d)
# And finally each file is renamed using the result of "f"
file.rename(i,f)
}

```

R script for renaming wav files for PIRO.

For this script, simply select all code and press ‘Run’. The code will produce a dialog box that will have the user choose the correct PIRO folder to run the code on.

```

m:\Monitoring\Bats\R\rename_wav_piro_v04.R
# PURPOSE: This program was written to rename .wav files for PIRO.
# ONLY USE FOR PIRO!!!
# For use with files produced from Petterson audio loggers.

# Revised: 4/27/18 by Al Kirschbaum

# Package management
list.of.pkgs <- c("stringr","rChoiceDialogs","xlsx","dplyr")
new.pkgs <- list.of.pkgs[!(list.of.pkgs %in% installed.packages()[,"Package"])]
if(length(new.pkgs)) install.packages(new.pkgs, repos='http://cran.rstudio.com/')

library(stringr)
library(rChoiceDialogs)
library(xlsx) #exporting and reading xlsx
library(dplyr)

# Capture errors
#options(error = utils::recover)

# Folder designation
folder <- rchoose.dir(default = "d:/",caption="PIRO folder, example d:/Bat_2016_corrected/PIRO")
folder <- gsub("\\\\", "/", folder, fixed=T)
folder <- gsub("//", "/", folder, fixed=T)

# Create a list of all files within "folder" that end in .wav
filelist <- list.files(folder,pattern = "*.wav", recursive = TRUE, full.names = TRUE, ignore.case = TRUE)
# Use this to keep track of progress
wav.length <- length(filelist)

# Checking for PIRO RAW files and making sure user wants to rename these. A raw file for PIRO starts w/ M
# create empty object
possible = NULL
for (i in filelist){
  split <- strsplit(i,"[/]")
  filename <- tail(split[[1]],1)
  #"P 05 Miners Falls-20170621_223741.wav"
  begin <- substr(filename,1,1)
  if (begin=="M"){
    possible = rbind(possible,data.frame(i))
  }
}
}

```

```

# if (length(possible!=0)){stop("There are sound files that begin w/ 'M', you should check these
before proceeding. If you determine the files in the list are indeed RAW files that start w/ M,
then comment out the stop line and rerun the script.")}
test <- nrow(possible)
stopifnot(test==0)

# Use this for counting up in the loop
n <- 1

# Create empty lists to store data
# I did a test comparing storing information in a dataframe vs a list and it was remarkably
different. 11 mins (list) vs 2 hrs (dataframe)
# d = NULL (if you want to store into a dataframe)
list_i <- list()
list_new.pn <- list()

start_time <- Sys.time()

# Now for the rename loop
for (i in filelist) {
  # "D:\\Bat_2016_corrected\\PIRO\\PIRO005\\WAV\\P 05 Miners Falls-20170621_223741.wav"
  dash1 <- strsplit(i,"-")
  dash1_len <- length(dash1[[1]])
  # "D:\\Bat_2016_corrected\\PIRO\\PIRO005\\WAV\\P 05 Miners Falls" "20170621_223741.wav"
  # split2 <- strsplit(i,"/")
  slash1 <- strsplit(i,"/")
  slash1_len <- length(slash1[[1]])
  # [1] "D:\\Bat_2016_corrected\\PIRO" "" "PIRO005"
  # [4] "WAV" "P 05 Miners Falls-20170621_223741.wav"
  if (slash1_len!=6){stop("You need to make sure there are 6 elements")}
  stopifnot(slash1_len==6)
  part1 <- slash1[[1]][1]
  # "D:\\Bat_2016_corrected\\PIRO"
  site <- slash1[[1]][4]
  # "PIRO005"
  part3 <- slash1[[1]][5]
  # "WAV"
  part4 <- tail(slash1[[1]],1)
  # "P 05 Miners Falls-20170621_223741.wav"
  begin <- substr(part4,1,1)
  # "P"
  if (length(dash1[[1]])==1){
    print (paste0(i, " has already been renamed and will be skipped. "))
    new.pathname1 <- i
  }
  if (begin=="M"){
    # "D:\\Bat_2016_corrected\\PIRO\\PIRO005\\WAV\\M01472.WAV"
    print (paste0("This file ", i, " begins w/ an M and will be treated as if it is a raw
file. "))
    part4.split1 <- strsplit(part4,"[.]")
    # "M01472" "WAV"
    name <- part4.split1[[1]][1]
    # "M01472"
    ext.wav <- part4.split1[[1]][2]
    # "WAV"
    lastpart <- substring(name,2)
    # "01472"
    new.pathname1 <- paste0(part1,"/",site,"/",part3,"/",site,"_",lastpart,".",ext.wav)
    # "D:\\Bat_2016_corrected\\PIRO\\PIRO005\\WAV\\PIRO005_01472.WAV"
    file.rename(i,new.pathname1)
  }
  # split1 length should normally equal 2, this is the most likely scenario
  if (dash1_len==2){
    print (paste0(i, " could be a normal file. "))
    part4.split2 <- strsplit(part4,"-")
    # "P 05 Miners Falls" "20170621_223741.wav"
    test <- part4.split2[[1]][2]
    # "20170621_223741.wav"
    nchar_test <- nchar(test)
    if (nchar_test==8){
      new.pathname1 <- i
    }
  }
}

```

```

        print (paste0(i, " this file has already been renamed and has spp tacked onto the end.
Nothing will be done."))
    } else {
        print (paste0(i, " will be renamed."))
        yde <- part4.split2[[1]][2]
        #"20170621_223741.wav"
        new.pathname1 <- paste0(part1,"/",site,"/",part3,"/",site,"_",yde)
        #"D:\\Bat_2016_corrected\\PIRO/PIRO005/WAV/PIRO005_20170621_223741.wav"
        file.rename(i,new.pathname1)
    }
}
if (length(dash1[[1]])>5){stop("Something else is going on, check on folder depth and names")}
# If the 'raw' file from PIRO has spp tacked onto the end of the filename, then the length of
split1 will be greater than two
# but should not be longer than 5 elements long. Sometimes the filename has multiple spp tacked
onto the end which will also make it longer.
# I drop the spp off the end of the filename b/c it is not needed and is always stored in the
RAW backup.
if (dash1_len>2){
    #"D:\\Bat_2016_corrected\\PIRO//PIRO005/WAV/P 05 Miners Falls-20170621_215542-Mylyu.wav"
    value <- "Non-GRTS"
    g <- grepl(value,i)
    part4.split3 <- strsplit(part4,"-")
    #"P 05 Miners Falls" "20170621_215542" "Mylyu.wav"
    part4.split4 <- strsplit(part4,".")
    #"P 05 Miners Falls-20170621_215542-Mylyu" "wav"
    ext <- part4.split4[[1]][2]
    #"wav"
    if (g=="TRUE"){
        print (paste0(i," should have non-grts in the name"))
        yde.ext <- part4.split3[[1]][3]
        part4.split5 <- strsplit(yde.ext,".")
        yde <- part4.split5[[1]][1]
        #"20170621_215542"
        new.pathname1 <- paste0(part1,"/",site,"/",part3,"/",site,"_",yde,".",ext)
        file.rename(i,new.pathname1)
    } else {
        print (paste0(i, " has multiple spp attached to the end, up to 4.))
        yde <- part4.split3[[1]][2]
        #"20170621_215542"
        new.pathname1 <- paste0(part1,"/",site,"/",part3,"/",site,"_",yde,".",ext)
        #"D:\\Bat_2016_corrected\\PIRO/PIRO005/WAV/PIRO005_20170621_215542.wav"
        file.rename(i,new.pathname1)
    }
}
print (paste0("File ",n," of ",wav.length," was processed"))
n <- n+1
#d = rbind (d,data.frame(i,new.pathname1)) for storing in a dataframe
list_i <- c(list_i,i)
list_new.pn <- c(list_new.pn,new.pathname1)
}

end_time <- Sys.time()

TIME <- end_time-start_time

df_i <- data.frame(matrix(unlist(list_i),nrow=wav.length,byrow = T),stringsAsFactors = F)
df_new.pn <- data.frame(matrix(unlist(list_new.pn),nrow=wav.length,byrow = T),stringsAsFactors =
F)
df <- cbind (df_i,df_new.pn)
df1 <- df %>%
    rename (filename=matrix.unlist.list_i...nrow...wav.length..byrow...T.,
            new.pathname=matrix.unlist.list_new.pn...nrow...wav.length..byrow...T.)

write.csv(df1,file=paste0(folder,"/", "rename_wav_piro.csv"),row.names = F)

```

Appendix B. R Script to Compare the Number of Files in Database and On Hard Drive

We use the program R to efficiently compare numbers of WAV files entered into the database for each site and the number of WAV files located on the hard drive.

Make sure the programs R and RStudio are installed on your computer. We will work in RStudio but R is also required because it will run in the background. Open RStudio and open the script to count the number of WAV files on the hard drive (count_wav_files_v03.R). These scripts contain code that will search your computer for the R packages needed for these scripts and will automatically download and install them if they are needed.

This script requires a .csv file with the number of audio files (.wav) entered for each site. First we need to export this .csv file from the bat database. Open the bat database and open the design view of the query called 'qry_GLKN_NumFiles'. Change the field called 'DeployYear' to the year you're analyzing. Run the query and save the .csv to a location on your computer. At this point you can run the R script 'count_wav_files_v03'.

R script for comparing the number of audio files

For this script, simply select all code and press 'Run'. The code will produce a dialog box that will have the user choose the .csv file that was output from the database. A separate dialog box will appear asking the user to choose the folder containing all of the audio files on the hard drive.

```
m:\Monitoring\Bats\R\count_wav_files_v03.R
# Script written to compare the number of wav files noted in the bat database to the number of
# wav files on a drive
# Rebecca Key is able to output a csv file that has the count of wav files by site. I import this
# csv file and
# compare that number to the number of wav files on the hard drive before archiving and
# creating clones.
# Written by: Al Kirschbaum
# Date: 3/26/18

# Package check/management -----
list.of.packages <- c("dplyr","xlsx","rChoiceDialogs")
new.pkgs <- list.of.pkgs[!(list.of.pkgs %in% installed.packages()[,"Package"])]
if(length(new.pkgs)) install.packages(new.pkgs, repos='http://cran.rstudio.com/')
library(dplyr)
library(xlsx) #exporting and reading xlsx
library(rChoiceDialogs) #better user interface dialog box

# User inputs -----
# this file needs to contain the following columns: year, park, site_cell, location, deploy, and
# wav...in this order
# the columns don't need to be named, b/c we will name them as they're imported
file <- rchoose.files(default="C:/Users/akirschbaum/documents",caption="Select csv file from RK")
db_data <- read.csv(file, header = F, col.names =
c("year","park","site_cell","location","deploy","wav"))

# Folder containing wav files for current year
# choose the root folder containing all of the parks, such as 'Bat_corrected_2017'
fromFolder <- rchoose.dir(default = "f:/",caption="Folder with wav files, example:
F:Bat_2016_corrected")
fromFolder <- gsub("\\\\","/",fromFolder,fixed=T)

year.list <- c('2015','2016','2017','2018','2019')
```

```

i <- menu(year.list,graphics = T,title = "Choose year")
year <- year.list[i]

# create a list of folders w/in the folder from above, this list will be used in the loop below
listOfFolders <- list.dirs(path=fromFolder,full.names = TRUE, recursive = FALSE)

# create two empty lists to store information during the loop process
list_loc <- list()
list_wav <- list()

# the loop goes into the first folder (i=APIS), then w/in this folder creates a list (sub) of
those folders (sites)
for (i in listOfFolders){
  print (i)
  #"F:/Bat_corrected_2017/APIS"
  subs <- list.dirs(path=i, full.names=TRUE,recursive=FALSE)
  print (subs)
  for (s in subs){
    print (s)
    #"F:/Bat_corrected_2017/APIS/APIS026A"
    path_split <- strsplit(s,'[/]')
    loc_name <- path_split[[1]][4]
    print (loc_name)
    #"APIS026A"
    listOfWavs <- list.files(s,pattern="\\.wav$",recursive = T,full.names = T,ignore.case = T)
    len <- length(listOfWavs)
    print (len)
    #2606
    list_loc <- c(list_loc,loc_name)
    list_wav <- c(list_wav,len)
  }
}

# There are now two populated lists, one with site names, one with the number of wav files in
that site folder
# You have to do a fair amount of programming to convert the list into a normal dataframe. This
was the path of
# least resistance when I was writing this program. If I had the time, I would try to populate
data frames instead of lists.

# the length the future data frame is needed
len <- length(list_loc)
# convert the lists to a dataframe
df_loc <- data.frame(matrix(unlist(list_loc),nrow=len,byrow = T),stringsAsFactors = F)
df_wav <- data.frame(matrix(unlist(list_wav),nrow=len,byrow = T),stringsAsFactors = F)
df <- cbind (df_loc,df_wav)
df2 <- df %>%
  rename
(location=matrix.unlist.list_loc...nrow...len..byrow...T.,drive_wav=matrix.unlist.list_wav...nrow
...len..byrow...T.)

# now I can compare the db results to the drive results
#db_data <- rename (db_data, location=Loc)
db_drive_compare <- right_join(db_data,df2, by='location')
db_drive_compare$compare <- db_drive_compare$wav - db_drive_compare$drive_wav

st=format(Sys.time(),"%Y%m%d_%H%M")
write.xlsx(db_drive_compare,file=paste0(fromFolder,"/", "wav_",year,"_compare_right_",st,".xlsx"),
row.names = F)

db_drive_compare_l <- left_join(db_data,df2, by='location')
db_drive_compare_l$compare <- db_drive_compare_l$wav - db_drive_compare_l$drive_wav

write.xlsx(db_drive_compare_l,file=paste0(fromFolder,"/", "wav_",year,"_compare_left_",st,".xlsx")
, row.names = F)

```

Appendix C. R Script to Compare the Number of Files on Network Drive and Hard Drive

We use the program R to efficiently compare numbers of files on the network drive (M drive) for each site and the number of files located on the hard drive.

Make sure the programs R and RStudio are installed on your computer. We will work in RStudio but R is also required because it will run in the background. Open RStudio and open the script to count the number of files on the hard drive (count_all_files_v05.R). The script contain code that will search your computer for the R packages needed for these scripts and will automatically download and install them if they are needed.

The script will prompt the user to point to the year on the M drive and the hard drive folder to analyze.

```
m:\Monitoring\Bats\R\count_all_files_v05.R
# Script written to compare the number of files on the M drive to the number of files on the
corrected drive
# Written by: Al Kirschbaum
# Date: 3/26/18

# Package check/management -----
list.of.packages <- c("dplyr","xlsx","rChoiceDialogs")
new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]
if(length(new.packages)) install.packages(new.packages)
library(dplyr)
library(xlsx) #exporting and reading xlsx
library(rChoiceDialogs)

# User inputs -----
# choose the root folder containing all of the parks, such as 'Bat_corrected_2017'
year.list <- c('2015','2016','2017','2018','2019')
i <- menu(year.list,graphics = T,title = "Choose year")
year <- year.list[i]

MDriveFolder <- rchoose.dir(default = paste0("//inpglknbrs1/Public/Monitoring/bats/data/field
data/",year," Summer Field Data"),caption="Bat field season on M drive, Ex: data/field data/2016
Summer Field Data")
MDriveFolder <- gsub("\\\\","/",MDriveFolder,fixed=T)
HardDriveFolder <- rchoose.dir(default = "f:/",caption="HD folder with wav files, Ex:
f:/Bat_2016_corrected")
HardDriveFolder <- gsub("\\\\","/",HardDriveFolder,fixed=T)

# create a list of folders w/in the folder from above, this list will be used in the loop below
listOfFolders <- list.dirs(path=MDriveFolder,full.names = TRUE, recursive = FALSE)

# create empty lists to store information during the loop process
list_loc <- list()
list_wav <- list()
list_txt <- list()
list_pdf <- list()
list_jpg <- list()
list_tif <- list()
list_doc <- list()
list_log <- list()
list_dmp <- list()
list_pgm <- list()
list_db <- list()
list_csv <- list()
list_xls <- list()
list_fil <- list()
```

```

# the loop goes into the first folder (i=APIS), then w/in this folder creates a list (sub) of
those folders (sites)
for (i in listOfFolders){
  print (i)
  #"/inpglknmbrs1/Public/Monitoring/Bats/Data/Field Data/2016 Summer Field Data/APIS"
  subs <- list.dirs(path=i, full.names=TRUE,recursive=FALSE)
  print (subs)
  for (s in subs){
    print (s)
    #"/inpglknmbrs1/Public/Monitoring/Bats/Data/Field Data/2016 Summer Field
Data/APIS/APIS026A"
    path_split <- strsplit(s,'[/]')
    loc_name <- tail(path_split[[1]],1) #retrieves the last item in a string after it has been
split
    #loc_name <- path_split[[1]][[1]]
    print (loc_name)
    allFiles <- list.files (s,recursive = T,full.names = T)
    wavCount <- length(allFiles[grepl('wav', allFiles, ignore.case = T)])
    txtCount <- length(allFiles[grepl('txt', allFiles, ignore.case = T)])
    pdfCount <- length(allFiles[grepl('pdf', allFiles, ignore.case = T)])
    jpgCount <- length(allFiles[grepl('jpg', allFiles, ignore.case = T)])
    tifCount <- length(allFiles[grepl('tif', allFiles, ignore.case = T)])
    docCount <- length(allFiles[grepl('doc', allFiles, ignore.case = T)]) #this will include .docx
    logCount <- length(allFiles[grepl('log', allFiles, ignore.case = T)])
    dmpCount <- length(allFiles[grepl('dump', allFiles, ignore.case = T)]) #this will include 3
and 4 dump files
    pgmCount <- length(allFiles[grepl('pgm', allFiles, ignore.case = T)])
    dbCount <- length(allFiles[grepl('db', allFiles, ignore.case = T)])
    xlsCount <- length(allFiles[grepl('xls', allFiles, ignore.case = T)]) #this will include .xlsx
    csvCount <- length(allFiles[grepl('csv', allFiles, ignore.case = T)])
    allCount <- length(allFiles)
    list_loc <- c(list_loc,loc_name)
    list_wav <- c(list_wav,wavCount)
    list_txt <- c(list_txt,txtCount)
    list_pdf <- c(list_pdf,pdfCount)
    list_jpg <- c(list_jpg,jpgCount)
    list_tif <- c(list_tif,tifCount)
    list_doc <- c(list_doc,docCount)
    list_log <- c(list_log,logCount)
    list_dmp <- c(list_dmp,dmpCount)
    list_pgm <- c(list_pgm,pgmCount)
    list_db <- c(list_db,dbCount)
    list_xls <- c(list_xls,xlsCount)
    list_csv <- c(list_csv,csvCount)
    list_fil <- c(list_fil,allCount)
  }
}

# You have to do a fair amount of programming to convert the list into a normal dataframe. This
was the path of
# least resistance when I was writing this program. If I had the time, I would try to populate
data frames instead of lists.

# the length the future data frame is needed
len <- length(list_loc)
# convert the lists to a dataframe
df_loc <- data.frame(matrix(unlist(list_loc),nrow=len,byrow = T),stringsAsFactors = F)
df_wav <- data.frame(matrix(unlist(list_wav),nrow=len,byrow = T),stringsAsFactors = F)
df_txt <- data.frame(matrix(unlist(list_txt),nrow=len,byrow = T),stringsAsFactors = F)
df_pdf <- data.frame(matrix(unlist(list_pdf),nrow=len,byrow = T),stringsAsFactors = F)
df_jpg <- data.frame(matrix(unlist(list_jpg),nrow=len,byrow = T),stringsAsFactors = F)
df_tif <- data.frame(matrix(unlist(list_tif),nrow=len,byrow = T),stringsAsFactors = F)
df_doc <- data.frame(matrix(unlist(list_doc),nrow=len,byrow = T),stringsAsFactors = F)
df_log <- data.frame(matrix(unlist(list_log),nrow=len,byrow = T),stringsAsFactors = F)
df_dmp <- data.frame(matrix(unlist(list_dmp),nrow=len,byrow = T),stringsAsFactors = F)
df_pgm <- data.frame(matrix(unlist(list_pgm),nrow=len,byrow = T),stringsAsFactors = F)
df_db <- data.frame(matrix(unlist(list_db),nrow=len,byrow = T),stringsAsFactors = F)
df_xls <- data.frame(matrix(unlist(list_xls),nrow=len,byrow = T),stringsAsFactors = F)
df_csv <- data.frame(matrix(unlist(list_csv),nrow=len,byrow = T),stringsAsFactors = F)
df_fil <- data.frame(matrix(unlist(list_fil),nrow=len,byrow = T),stringsAsFactors = F)

```

```

df <- cbind
(df_loc,df_wav,df_txt,df_pdf,df_jpg,df_tif,df_doc,df_log,df_dmp,df_pgm,df_db,df_xls,df_csv,df_fil
)
m_drive <- df %>%
  rename (loc=matrix.unlist.list_loc...nrow...len..byrow...T.,
    m_wav=matrix.unlist.list_wav...nrow...len..byrow...T.,
    m_txt=matrix.unlist.list_txt...nrow...len..byrow...T.,
    m_pdf=matrix.unlist.list_pdf...nrow...len..byrow...T.,
    m_jpg=matrix.unlist.list_jpg...nrow...len..byrow...T.,
    m_tif=matrix.unlist.list_tif...nrow...len..byrow...T.,
    m_doc=matrix.unlist.list_doc...nrow...len..byrow...T.,
    m_log=matrix.unlist.list_log...nrow...len..byrow...T.,
    m_dmp=matrix.unlist.list_dmp...nrow...len..byrow...T.,
    m_pgm=matrix.unlist.list_pgm...nrow...len..byrow...T.,
    m_db=matrix.unlist.list_db...nrow...len..byrow...T.,
    m_xls=matrix.unlist.list_xls...nrow...len..byrow...T.,
    m_csv=matrix.unlist.list_csv...nrow...len..byrow...T.,
    m_fil=matrix.unlist.list_fil...nrow...len..byrow...T.) %>%
  mutate(m_file_tot = m_wav + m_txt + m_pdf + m_jpg + m_tif + m_doc + m_log + m_dmp + m_pgm +
    m_db + m_csv + m_xls,
    m_check1 = m_fil - m_file_tot,
    m_non_wav = m_txt + m_pdf + m_jpg + m_tif + m_doc + m_log + m_dmp + m_pgm + m_db + m_csv
    + m_xls)

# write.xlsx(m_drive,file="b:/Data/Field Data/2016 Summer Field
Data/file_summary_m2.xlsx",row.names = F)

```

```

# create a list of folders w/in the folder from above, this list will be used in the loop below
listOfFolders <- list.dirs(path=HardDriveFolder,full.names = TRUE, recursive = FALSE)

```

```

# create empty lists to store information during the loop process
list_loc <- list()
list_wav <- list()
list_txt <- list()
list_pdf <- list()
list_jpg <- list()
list_tif <- list()
list_doc <- list()
list_log <- list()
list_dmp <- list()
list_pgm <- list()
list_db <- list()
list_csv <- list()
list_xls <- list()
list_fil <- list()

```

```

# the loop goes into the first folder (i=APIS), then w/in this folder creates a list (sub) of
those folders (sites)
for (i in listOfFolders){
  print (i)
  #"F:\\Bat_corrected_2017//APIS"
  subs <- list.dirs(path=i, full.names=TRUE,recursive=FALSE)
  print (subs)
  for (s in subs){
    print (s)
    #"F:\\Bat_corrected_2017//APIS/APIS026A"
    path_split <- strsplit(s,'[/]')
    loc_name <- tail(path_split[[1]],1) #retrieves the last item in a string after it has been
split
    #loc_name <- path_split[[1]][4]
    print (loc_name)
    allFiles <- list.files (s,recursive = T,full.names = T)
    wavCount <- length(allFiles[grepl('wav', allFiles, ignore.case = T)])
    txtCount <- length(allFiles[grepl('txt', allFiles, ignore.case = T)])
    pdfCount <- length(allFiles[grepl('pdf', allFiles, ignore.case = T)])
  }
}

```

```

jpgCount <- length(allFiles[grepl('jpg', allFiles, ignore.case = T)])
tifCount <- length(allFiles[grepl('tif', allFiles, ignore.case = T)])
docCount <- length(allFiles[grepl('doc', allFiles, ignore.case = T)]) #this will include .docx
logCount <- length(allFiles[grepl('log', allFiles, ignore.case = T)])
dmpCount <- length(allFiles[grepl('dump', allFiles, ignore.case = T)])
pgmCount <- length(allFiles[grepl('pgm', allFiles, ignore.case = T)])
dbCount <- length(allFiles[grepl('db', allFiles, ignore.case = T)])
xlsCount <- length(allFiles[grepl('xls', allFiles, ignore.case = T)]) #this will include .xlsx
csvCount <- length(allFiles[grepl('csv', allFiles, ignore.case = T)])
allCount <- length(allFiles)
list_loc <- c(list_loc, loc_name)
list_wav <- c(list_wav, wavCount)
list_txt <- c(list_txt, txtCount)
list_pdf <- c(list_pdf, pdfCount)
list_jpg <- c(list_jpg, jpgCount)
list_tif <- c(list_tif, tifCount)
list_doc <- c(list_doc, docCount)
list_log <- c(list_log, logCount)
list_dmp <- c(list_dmp, dmpCount)
list_pgm <- c(list_pgm, pgmCount)
list_db <- c(list_db, dbCount)
list_xls <- c(list_xls, xlsCount)
list_csv <- c(list_csv, csvCount)
list_fil <- c(list_fil, allCount)
}
}
# There are now two populated lists, one with site names, one with the number of wav files in
that site folder
# You have to do a fair amount of programming to convert the list into a normal dataframe. This
was the path of
# least resistance when I was writing this program. If I had the time, I would try to populate
data frames instead of lists.

# the length the future data frame is needed
len <- length(list_loc)
# convert the lists to a dataframe
df_loc <- data.frame(matrix(unlist(list_loc), nrow=len, byrow = T), stringsAsFactors = F)
df_wav <- data.frame(matrix(unlist(list_wav), nrow=len, byrow = T), stringsAsFactors = F)
df_txt <- data.frame(matrix(unlist(list_txt), nrow=len, byrow = T), stringsAsFactors = F)
df_pdf <- data.frame(matrix(unlist(list_pdf), nrow=len, byrow = T), stringsAsFactors = F)
df_jpg <- data.frame(matrix(unlist(list_jpg), nrow=len, byrow = T), stringsAsFactors = F)
df_tif <- data.frame(matrix(unlist(list_tif), nrow=len, byrow = T), stringsAsFactors = F)
df_doc <- data.frame(matrix(unlist(list_doc), nrow=len, byrow = T), stringsAsFactors = F)
df_log <- data.frame(matrix(unlist(list_log), nrow=len, byrow = T), stringsAsFactors = F)
df_dmp <- data.frame(matrix(unlist(list_dmp), nrow=len, byrow = T), stringsAsFactors = F)
df_pgm <- data.frame(matrix(unlist(list_pgm), nrow=len, byrow = T), stringsAsFactors = F)
df_db <- data.frame(matrix(unlist(list_db), nrow=len, byrow = T), stringsAsFactors = F)
df_xls <- data.frame(matrix(unlist(list_xls), nrow=len, byrow = T), stringsAsFactors = F)
df_csv <- data.frame(matrix(unlist(list_csv), nrow=len, byrow = T), stringsAsFactors = F)
df_fil <- data.frame(matrix(unlist(list_fil), nrow=len, byrow = T), stringsAsFactors = F)
df <- cbind
(df_loc, df_wav, df_txt, df_pdf, df_jpg, df_tif, df_doc, df_log, df_dmp, df_pgm, df_db, df_xls, df_csv, df_fil
)
hard_drive <- df %>%
  rename (loc=matrix.unlist.list_loc...nrow...len..byrow...T.,
    hd_wav=matrix.unlist.list_wav...nrow...len..byrow...T.,
    hd_txt=matrix.unlist.list_txt...nrow...len..byrow...T.,
    hd_pdf=matrix.unlist.list_pdf...nrow...len..byrow...T.,
    hd_jpg=matrix.unlist.list_jpg...nrow...len..byrow...T.,
    hd_tif=matrix.unlist.list_tif...nrow...len..byrow...T.,
    hd_doc=matrix.unlist.list_doc...nrow...len..byrow...T.,
    hd_log=matrix.unlist.list_log...nrow...len..byrow...T.,
    hd_dmp=matrix.unlist.list_dmp...nrow...len..byrow...T.,
    hd_pgm=matrix.unlist.list_pgm...nrow...len..byrow...T.,
    hd_db=matrix.unlist.list_db...nrow...len..byrow...T.,
    hd_xls=matrix.unlist.list_xls...nrow...len..byrow...T.,
    hd_csv=matrix.unlist.list_csv...nrow...len..byrow...T.,
    hd_fil=matrix.unlist.list_fil...nrow...len..byrow...T.) %>%
  mutate(hd_file_tot = hd_wav + hd_txt + hd_pdf + hd_jpg + hd_tif + hd_doc + hd_log + hd_dmp +
hd_pgm + hd_db + hd_csv + hd_xls,
    hd_check1 = hd_fil - hd_file_tot,

```

```

      hd_non_wav = hd_txt + hd_pdf + hd_jpg + hd_tif + hd_doc + hd_log + hd_dmp + hd_pgm +
hd_db + hd_csv + hd_xls)

# write.xlsx(hard_drive,file="b:/Data/Field Data/2016 Summer Field
Data/file_summary_hd.xlsx",row.names = F)

drive_compare <- right_join(m_drive,hard_drive, by='loc')
drive_compare <- drive_compare %>%
  mutate(txt=m_txt-hd_txt, pdf=m_pdf-hd_pdf, jpg=m_jpg-hd_jpg, tif=m_tif-hd_tif, doc=m_doc-
hd_doc, log=m_log-hd_log,
         dmp=m_dmp-hd_dmp, pgm=m_pgm-hd_pgm, xls=m_xls-hd_xls, csv=m_csv-hd_csv)

st=format(Sys.time(),"%Y%m%d %H%M")
write.xlsx(drive_compare,file=(paste0(HardDriveFolder,"/","file_count_",year,"_compare","_",st,".
xlsx")),row.names = F)

```


Appendix D. R Script to Compare the Modification Times and Sizes of Files on the Network Drive and Hard Drive

We use the program R to efficiently compare size and modification date of files on the network drive (M drive) to the files on the corrected hard drive.

Make sure the programs R and RStudio are installed on your computer. We will work in RStudio but R is also required because it will run in the background. Open RStudio and open the script to compare file information (get_file_info_v03.R). The script contain code that will search your computer for the R packages needed for these scripts and will automatically download and install them if they are needed.

The script will prompt the user to point to the year on the M drive and the hard drive folder to analyze.

```
m:\Monitoring\Bats\R\get_file_info_v03.R
# Script written to compare the size and modification date of files on the M drive, to files on
our 'corrected' external drive
# Written by: Al Kirschbaum
# Date: 4/18/2018

# Package check/management -----
list.of.packages <- c("dplyr","xlsx","stringr","data.table","reshape2")
new.pkgs <- list.of.pkgs[!(list.of.pkgs %in% installed.packages()[,"Package"])]
if(length(new.pkgs)) install.packages(new.pkgs, repos='http://cran.rstudio.com/')
library (stringr)
library (dplyr)
library (xlsx)
library (data.table)
library (reshape2)

# User inputs -----
# Folder containing wav files for current year
year.list <- c('2015','2016','2017','2018','2019')
i <- menu(year.list,graphics = T,title = "Choose year to compare")
data.year <- year.list[i]
MDriveFolder <- choose.dir(default = (paste0("M:/Monitoring/bats/data/field data/",data.year,"
Summer Field Data")),caption="Ex: 2017 Summer Field Data on M drive")
MDriveFolder <- gsub("\\\\","/",MDriveFolder,fixed=T)
MDriveFolder <- gsub("//","/",MDriveFolder,fixed=T)
HardDriveFolder <- choose.dir(default = "F:/",caption="HD folder with correct structure")
HardDriveFolder <- gsub("\\\\","/",HardDriveFolder,fixed=T)
HardDriveFolder <- gsub("//","/",HardDriveFolder,fixed=T)

# this example file will be used to create a dataframe, below
create_empty_df <- choose.files(default = (paste0("M:/Monitoring/bats/data/field
data/",data.year," Summer Field Data/APIS/APIS001A")),caption="Choose an example pdf file from M
drive")

# M drive -----
# one easy way to create an empty dataframe w/ the correct column names and types, is to create
it w/ one record, then delete the record
# now you have an empty dataframe, ready to populate
build_str <- file.info(create_empty_df)
# now remove the row
m_files <- build_str[FALSE,]
# create a list of files sitting in the folder. I will loop through these files and extract file
information
listOfFiles_M <- list.files(MDriveFolder,recursive = T, full.names = T)
listOfFiles_M <- listOfFiles_M[lapply(listOfFiles_M,function(x) length(grep(".wav",x,value =
F)))==0]
```

```

# The loop, short and simple, get the file, extract info and put this info into the empty
dataframe, adding a row for each iteration
for (i in listOfFiles_M){
  print (i)
  m_files[i,] = c(file.info(i)) #this is where the row gets appended to the dataframe
}

# use data.table to give the first column a title
m_files2 <- setDT(m_files,keep.rownames = TRUE)[]
# convert back to a dataframe
m_files2 <- as.data.frame(m_files2)

# there is information in the first column, called 'rn', here I break the values in this column
into additional columns
# since there are three different separators, I need to do this three times
# all of these could change depending on the folder depth...which shouldn't change for the M
drive, unless we change the folder structure
# I'm adding checks
splitCheck1 <- strsplit(i,"/")
#if (length(splitCheck1[[1]])!=6){stop("You need to make sure there are 6 elements")}
cols_split1_m <-
colsplit(m_files2$rn,"/",c("drive","fold1","fold2","fold3","fold4","fold5","park","site","fn"))

splitCheck2 <- strsplit(i,"[.]")
#if (length(splitCheck3[[1]])!=2){stop("You need to make sure there are 2 elements")}
cols_split2_m <- colsplit(m_files2$rn,"[.]",c("fold7","ext"))

m_files3 <- cbind(m_files2,cols_split1_m,cols_split2_m)
m_files4 <- m_files3 %>%
  select (size,mtime,ctime,drive,park,site,fn,ext)%>%
  rename (size_m=size, mtime_m=mtime, ctime_m=ctime, drive_m=drive)

# Hard drive -----
hd_files <- build_str[FALSE,]

listOfFiles_HD <- list.files(HardDriveFolder,recursive = T, full.names = T)
listOfFiles_HD <- listOfFiles_HD[lapply(listOfFiles_HD,function(x) length(grep(".wav",x,value =
F)))!=0]
listOfFiles_HD <- gsub("\\","/",listOfFiles_HD,fixed=T)
listOfFiles_HD <- gsub("//","/",listOfFiles_HD,fixed=T)

# listOfFiles_HD_pdf <- list.files(HardDriveFolder,pattern="\\.pdf$",recursive = T, full.names =
T, ignore.case = T)
# listOfFiles_HD_doc1 <- list.files(HardDriveFolder,pattern="\\.doc$",recursive = T, full.names =
T, ignore.case = T)
# listOfFiles_HD_doc2 <- list.files(HardDriveFolder,pattern="\\.docx$",recursive = T, full.names
= T, ignore.case = T)
# listOfFiles_HD_xl1 <- list.files(HardDriveFolder,pattern="\\.xlsx$",recursive = T, full.names =
T, ignore.case = T)
# listOfFiles_HD_xl2 <- list.files(HardDriveFolder,pattern="\\.xls$",recursive = T, full.names =
T, ignore.case = T)
# listOfFiles_HD_csv <- list.files(HardDriveFolder,pattern="\\.csv$",recursive = T, full.names =
T, ignore.case = T)
# listOfFiles_HD_jpg <- list.files(HardDriveFolder,pattern="\\.jpg$",recursive = T, full.names =
T, ignore.case = T)
# listOfFiles_HD_tif <- list.files(HardDriveFolder,pattern="\\.tif$",recursive = T, full.names =
T, ignore.case = T)
# listOfFiles_HD_txt <- list.files(HardDriveFolder,pattern="\\.txt$",recursive = T, full.names =
T, ignore.case = T)
# listOfFiles_HD_log <- list.files(HardDriveFolder,pattern="\\.LOG$",recursive = T, full.names =
T, ignore.case = T)
# listOfFiles_HD <-
combine(list(listOfFiles_HD_doc1,listOfFiles_HD_doc2,listOfFiles_HD_pdf,listOfFiles_HD_tif,listOf
Files_HD_jpg,
#
listOfFiles_HD_csv,listOfFiles_HD_xl2,listOfFiles_HD_xl1,listOfFiles_HD_txt,listOfFiles_HD_log))

for (i in listOfFiles_HD){
  print (i)
  hd_files[i,] = c(file.info(i))
}

```

```

}

hd_files2 <- setDT(hd_files,keep.rownames = TRUE)[ ]
hd_files2 <- as.data.frame(hd_files2)

splitCheck4 <- strsplit(i,"/")
# if (length(splitCheck4[[1]])!=2){stop("You need to make sure there are 2 elements")}
cols_split1_hd <- colsplit(hd_files2$rn,"/",c("drive","fold1","park","site","fn"))

# splitCheck5 <- strsplit(i,"/")
# if (length(splitCheck5[[1]])!=4){stop("You need to make sure there are 4 elements")}
# cols_split2_hd <- colsplit(hd_files2$rn,"/",c("fold2","park","site","fn"))

splitCheck5 <- strsplit(i,"[.]")
# if (length(splitCheck5[[1]])!=2){stop("You need to make sure there are 2 elements")}
cols_split2_hd <- colsplit(hd_files2$rn,"[.]",c("fold3","ext"))

hd_files3 <- cbind(hd_files2,cols_split1_hd,cols_split2_hd)
hd_files4 <- hd_files3 %>%
  select (size,mtime,ctime,drive,park,site,fn,ext)%>%
  rename (size_hd=size, mtime_hd=mtime, ctime_hd=ctime, drive_hd=drive)

# Combine and compare -----
compare1 <- left_join(m_files4,hd_files4, by=c('park','site','fn','ext'))
compare2 <- compare1 %>%
  mutate(size_diff = size_m - size_hd, mtime_diff = mtime_hd - mtime_m, ctime_diff = ctime_m -
  ctime_hd)
# %>%
# filter(ext == "PDF" | ext=="pdf" | ext=="docx")

st=format(Sys.time(),"%Y%m%d_%H%M")
write.xlsx(compare2,file=(paste0(HardDriveFolder,"/","file_info_",data.year,"_compare_",st,".xlsx
")),row.names = F)

```