Bat monitoring guidelines using automatic bat detectors

Bat Recovery Group

Version 5 September 2025

This document provides guidance on monitoring of bats using primarily the Department of Conservation AR4 bat detectors. The process, however, is the same for whatever recorder you are using. This document should be read in conjunction with the "Surveying for bats – guidance for community groups". Community groups may want to get some statistical support if they want to follow the monitoring guidelines. For more detailed information on survey and monitoring methods please refer to Best practice manual of conservation techniques for pekapeka/bats in Aotearoa New Zealand.

Monitoring is defined as 'the systematic collection of ecological data in a standardized manner at regular intervals over time' (Cambridge University Press).

The first steps of any monitoring programme are to figure out

1. Why are you doing the monitoring? What are your objectives?
2. What do you need to measure to understand whether you have met your objectives?
3. Which method is suitable for the species you want to measure and the project?

This document focuses on monitoring of bats' echolocation calls using automated bat detectors (also known as Automated bat monitoring units or ABMs). It focuses on the Department of Conservation AR4 bat detectors, but the basic principles can be applied to other detectors. It is not recommended to use a variety of different

detectors for standard monitoring on one project as each detector model will have different sensitivities. Other monitoring techniques are extensively covered in the Best practice manual of conservation techniques for pekapeka/bats in Aotearoa New Zealand

Technology has improved bat detectors so that they are smaller and cheaper than in previous times. This means large monitoring programmes are more achievable. The electronics division of the Department of Conservation developed a simple recorder (AR4) in 2012 that was designed specifically for NZ bats and is relatively cheap compared to international models. It has been so successful that there are now 8000 in circulation and there is trouble meeting demand for the product. Recently a new company (Alato) has taken over production of AR4 detectors and updated the model to the AR5. AR5 Acoustic Recorder. The model is basically the same as the AR4 with minor modifications and can be used interchangeably with the AR4.

Things to consider for monitoring

To minimise variability and ensure consistency, the **same** sites have to be done at the **same** time of year using the **same** type of automatic bat detectors for the length of the project. This commitment needs to be taken into account when considering whether to begin a project. To minimise weather variabilities only use data from nights where the temperature is above 7 degrees C (South Island) or 10 degrees C (North Island).

Calculating the number of points and amount of time required for a bat monitoring programme

Before beginning, you need to calculate the number of detectors you will need and the number of nights you will need to record for to generate meaningful data which can analyse trends in bat activity. Previous studies suggest that 20-25 recorders for 12-15 nights for 7-10 years as a guideline (DOC unpublished data).

To test your own data, you can do a power analysis. Power is related to how confident you are that you are detecting a true effect. A power analysis uses a pre-existing dataset to give an indication of the number of sites required to be monitored to detect a change in activity.

DOC have been following the methods in Power analysis.pdf. Initial studies show that you need a 7 - 10-year project with 20-25 recorders (detectors) over 12-15 nights to detect a 5% annual change with 80% power. This means that you have 80% confidence that the effect is true. This project design accounts for the inherent variability in bat activity and can be used as a basis for your study.

If you think your data is different or unusual you may want to test your own data, the code is in Appendix A and is based on Green and Macleod 2016 otherwise use our recommendations.

Sampling design: The Master Sample

There are a range of options for designing monitoring. The Department of Conservation uses a particular sampling design to aid with coordinating monitoring efforts. The Bat Recovery Group has been testing this spatial design for bat monitoring. The theory is explained in the paper the master sample. The sampling design is designed to avoid duplication of effort and can be used locally, regionally and nationally. This means that a small community group can contribute to national monitoring, if they use this sampling design. If you are a community group, you may need some additional advice and help to set up the programme. This document outlines the steps required so that you can work out what help and equipment that you need.

To set up a spatially designed programme;

1. define the area that you want to make a statement about (as a shapefile - a file that can be used by a mapping programme)
2. generate a surplus of points in R using the code from Appendix B

3. start from number 1. You then go onto the next numerical number until you reach the number of points that you require. You don't need to use them all, but you must have a reason for not using a point.

Running a bat monitoring project using AR4s (or similar model detector e.g., AR5).

The instructions below refer to projects that are specifically using AR4s – if you have other types of detectors, then a different process will be required that is not covered in this document.

AR4s can be checked using the DOC bat recorder tester app DOC - Bat Recorder Tester – Apps on Google Play that is available from google play store. It is important that recorders are checked before each survey as sometimes the microphones or clock batteries need replacing. Refer to the Electronics team support and service if you need to send any units back to be fixed and for maintenance tips.

Processing data from AR4s can be done using the Bat search 3 programme (found in the Electronics website). This programme allows you to look at the spectrogram picture and decide whether or not is a bat. There is a library of identified calls on the Electronics team support and service to help you in this process. Alternatively, you can use AviaNZ that processes the data using AI but there will be some false positives. Either way a csv file is produced with the results of the processed data, and this is automatically saved with the raw data. If you are using another type of recorder, you may process the data in a different way, but you will still produce a file with positive detections.

Managing data so that it can be analysed

Processing each AR4 will produce a csv file for each site. If you have 25 recorders, then this is a lot of data. The data all needs to be collated so that it can be analysed in R (a stats package). Ideally all the processed data for a survey needs to be in one csv file and then it can be summarised per night using the code in Appendix C. The format to analyse the data in R is Year, Station, Day, Passes (Table 1). The days need to be consistent so day 1 needs to be the same date across the survey. Each station

also needs to be consistent across the years, so Station 1 is always the same site. Note that to use R, you will need to understand coding to some level.

Year	Station	Day	Passes
1	1	1	12
1	1	2	3
1	1	3	0
1	1	4	0
1	1	5	0
1	1	6	13
1	2	1	149
1	2	2	66

Table 1. example of the format needed for the analysis

Once the data is in one csv file in the format of Table 1 then the data can be analysed. Appendix D shows the code that can be used.

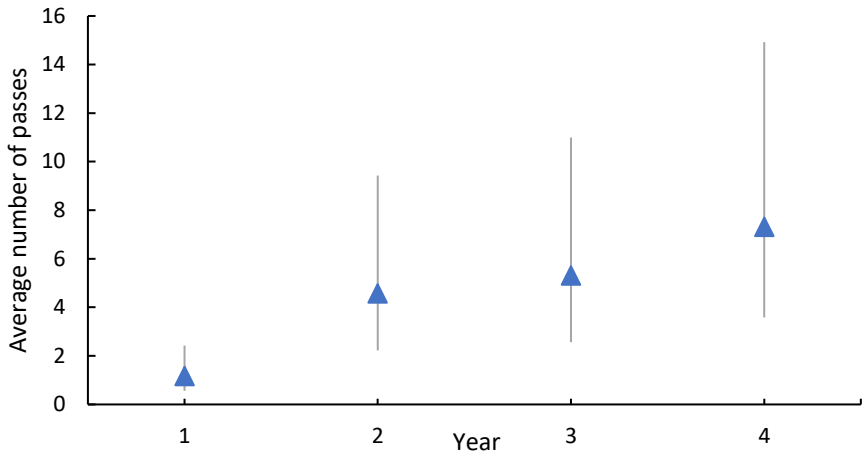
Example of analyses

Karen is responsible for the monitoring of long-tailed bats for her community group. The group decided that they wanted to look to see if there are changes in the long-tailed bat activity rate (the number of calls per night) over time, because they are trapping rats, stoats, possums and cats – all predators of bats.

They did a power analysis and figured out that they need 20 detectors recording for 10 nights a year and to be able to detect an annual change in activity rates of 5% with 80% certainty and it will take 7 years to detect that change.

Karen set up their data in a CSV file and then used the code in Appendix D to analyse the data using the programme R on her laptop.

Karen notes that the year was significant, and the acoustic activity is increasing with year (see Appendix E for details of the analysis and the outputs).



A simpler analysis could just be to use the mean number of passes per year, per station, per night. This would be less accurate as it does not take into account the variation in year, station and the date of the survey but it is still useful and produces a similar trend.

Appendix A – code to do a power analysis based on the work by Green & MacLeod and modified by Ellen Ciaraad (DOC statistician).

```
remove(list=ls())

# this script assumes all relevant files are in the folder associated with this project

# set this to where your data is

setwd("C:\\mpryde\\bat detectors\\Eglinton bats acoustic")

# load libraries ----

library(car)

library(MuMIn)

# library(AICcmodavg)

# library(httr)

# library(DHARMa)

library(lme4)

library(simr)

library(readxl)

library(tidyverse)

#includes dplyr, ggplot2; load most used package last

#### PREPARE DATA ############################

## brings in all data, included in this data are zero species,

# make sure missing data comes in as NA

dat<-read.table("simrlongpower.csv",

          header=TRUE, sep=",", na.strings=c("NA"),

          colClasses = c("numeric","numeric","numeric","numeric"))

str(dat)    # check structure

head(dat)   #

#list(dat)

summary(dat)

histogram(Passes~Year|Station,data=dat)
```

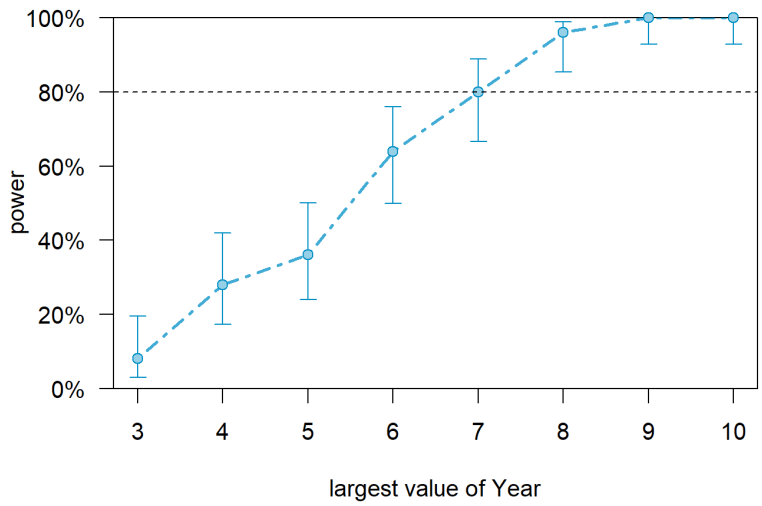
```
histogram(Passes~Year|Day,data=dat)

table(dat$Station)

table(dat$Year)

table(dat$Day)

# see below for practice example simr package

### BAT DATA MODEL & SIMS ######################

dat$YearDay <- paste(dat$Year, dat$Day)

dat<-na.omit(dat)

trend.to.detect <- 0.05 # this is what you want to detect

# start simple:

 model1 <- glmer.nb(Passes ~ Year + (1|Station), data=dat)

 model1

 # this takes into account that observations at any one station have

# lower variance (are more similar) than obs between stations


model2 <- glmer.nb(Passes ~ Year + (1|Station) + (1|YearDay), data=dat)

model2

# having a YearDay combination as a random effect

# creates a structure that obs at any one night in the dataset

# are treated as being more similar than between nights


# simr package tricky to extend data appropriately

# could create custom extend function:

# https://github.com/pitakakariki/simr/blob/master/R/extend.R


# ideally model would take into account that observations

# are nested within night and within year,

# but with only 2 years of data the model is too complicated & doesn't converge
```

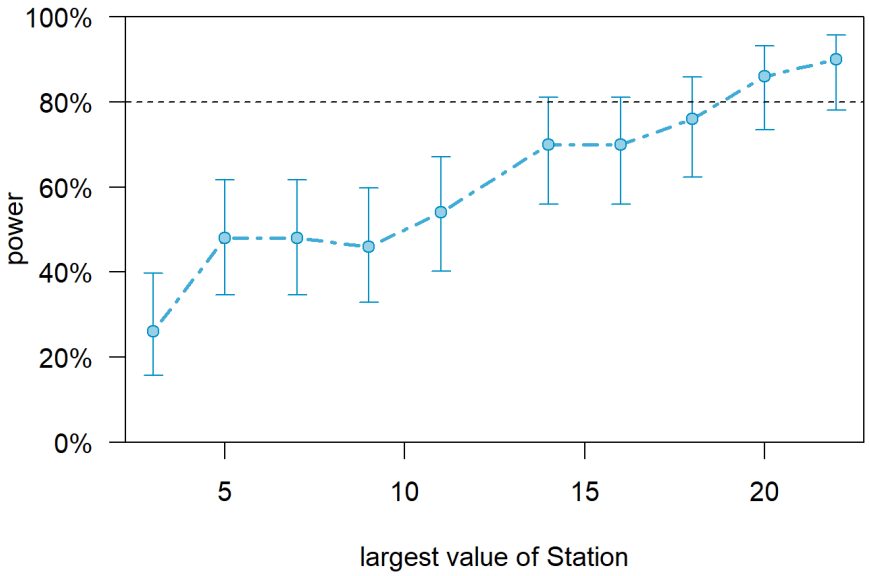
```
model3 <- glmer.nb(Passes ~ Year + (1|Station) + (1|Year/Day), data=dat)

model3

# next best model we can use

# year nested within station: this accounts for repeated measurements at each station

model4 <- glmer.nb(Passes ~ Year + (1|Station/Year), data=dat)

model4

fixef(model1)["Year"] <- trend.to.detect

(pc1 <- powerCurve(model1, nsim = 5))

(ps1 <- powerSim(model1, nsim = 5))

model2 <- extend(model1, along="Year", n=7)

nrow(getData(model2))

summary(getData(model2))

(ps2 <- powerSim(model2, nsim = 50))

(pc2 <- powerCurve(model2, nsim = 50))

plot(pc2)

model3 <- extend(model2,  along="Station", n=22)

pc3 <- powerCurve(model3, along="Station", nsim = 50)

plot(pc3)

model4 <- extend(model2, within="Year+Station", n=12)

pc4 <- powerCurve(model4, within="Year+Station", nsim = 50)

plot(pc4)

# increase simulations when you're doing it for real

# default = nsim = 1000

# or decrease to run it fast and having an explore

# compare whether this nested model fits better than simple Station random effect

# model with lowest AIC fits best

model0 <- glmer.nb(Passes ~ 1 + (1|Station) , data=dat)

model1a <- glmer.nb(Passes ~ Year + (1|Station) , data=dat)
```

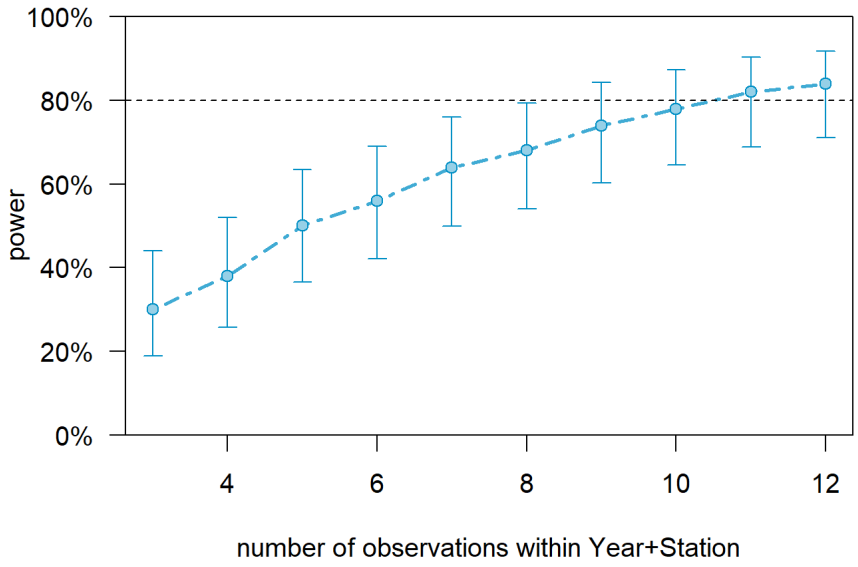
```
model1b <- glmer.nb(Passes ~ Year + (1|Station/Year) , data=dat)

dplyr::arrange(anova(model0, model1a, model1b), AIC)

r2year <- MuMIn::r.squaredLR(model1b.0, model1b)

# in the MuMIn package this function

# provides pseudo-R2 for the variable year in the model1b

# (by comparing it to the same model without the year as a fixed effect

############ !

####### EXTRACT MODEL COEFFICIENTS FOR PLOTTING ETC.


model0 <- glmmTMB(Passes ~ 1 + (1|Station/Year) , data=dat, family=nbinom2)

model1 <- glmer.nb(Passes ~ Year + (1|Station/Year), data=dat)


modelA <- glmmTMB(Passes ~ Year - 1 + (1|Station/Year), data=dat, family=nbinom2)

# By adding argument - 1, this indicates that there should be no

# intercept term in the model. This means that the regression line is

# forced to go through the origin (0,0) rather than allowing for a non-zero intercept.

# if you want year as a factor: i.e. don't want a linear trend over time

# but independent estimate of count per year (taking into account random effects)

# then need to turn Year into a categorical variable

# this makes more sense when you have > 2 years worth of data

# allows you to plot the means as estimated by the model

# (rather than simply calculating arithmetic mean without random effect structure etc)

# modelB <- glmer.nb(Passes ~ as.factor(Year)  + (1|Station/Year), data=dat)

## i would suggest you use the emmeans package to extract & backtransform the coefficients

library(emmeans)

emmeans(model1b, specs = "Year", type = "response")

# this turns the model coefficients (the fixed terms component, population based estimate,

# so setting the random effects to zero)
```

```
# type = "response" returns the back-transformed log coefficients

# if you leave that off then it's returned on the transformed, in this case log, scale

pairs(emmeans(model1b, specs = "Year", type = "response"))

# provides tests between the different levels of year


library(car)    # use Anova() on a model output for overall significance value for the fixed effects

car::Anova(model1)
```

Example output files






100%
80%
60%
40%
20%
0%
power
4
6
8
10
12
number of observations within Year+Station

Appendix B – code to produce a spatially balanced set of points

```
# code written by Ollie Gansell, DOC

#Code to install and run spbal to get

#sample points for the NZ mastersample


#Install spbal dev version as current R CRAN version has a bug – this bug has been sorted
#  remotes::install_github("docgovtnz", ref = "iteratorbug")

#Install companion package which has master sample seeds and bounding boxes for #NZ
 remotes::install_github("ogansell/spbalRef")

#Load required packages

 library(dplyr)
 library(sf)
 library(spbal)
 library(spbalRef)
 library(mapview)

#Load shapefile. Change filepath to location for your shapefile

 shp <-st_read("filepath/sample_frame.shp")

#Make sure shapefile representing sample frame is the right coordinate system.
#Here I'm converting it to the same projection as the North island bounding box
 shp <-st_transform(shp, st_crs(north_nz$bb))

#Generate sample for sample frame
 shp_points <- spbal::BAS(shapefile = shp,
                          n = 30,
                          #Call bounding box for north island mastersample.
                          #Change to south_nz$bb if in the South Island
                          boundingbox = north_nz$bb,
                          #Call bounding box for north island mastersample.
                          #Change to south_nz$seed if in the South Island
                          seeds = north_nz$seed,
                          verbose = FALSE)

#Apply unique ID for North Island master sample
 shp_points <-uniqueID(north_nz,shp_points)

#Have a look
 mapview(shp_points$sample) + mapview(shp, alpha.regions = 0.3)

#Prepare to export
 shp_points$sample$Easting <- st_coordinates(shp_points$sample)[,1]
 shp_points$sample$Northing <- st_coordinates(shp_points$sample)[,2]

#Write to csv,shp,gpkg etc

 st_write(shp_points$sample,"sample_points.csv")
 st_write(shp_points$sample,"sample_points.shp")
```

```
 st_write(shp_points$sample,"sample.gpkg", layer = "sample_points")
#ENDS
```

Appendix C – summarising data so that it can used in analysis

Code written by Elle Ciaraad

```
# load libraries ----
library(writexl)
library(lubridate)
library(tidyverse)
# Suppress summarise info from dplyr
options(dplyr.summarise.inform = FALSE) # suppress messages about grouping

setwd("set the file path for your data")
"allbats2023.csv"

### INPUT is a Bat Search output
#1. filepath: path pointing to a .csv file outputted by BatSearch, must include columns:
#   e.g. "Egbats2020.csv"
   # Date
   # Time
   # AssignedSite = Station
   # Category = either Short tail, Long tail or Both (all other categories will be ignored)
   # Foldername (not used)
   # Filename (not used)
   # Observer (not used)
#2. location: used to name the output files/worksheets, e.g. "Eglinton"
# e.g.
# batsearch_summarise("Egbats2020.csv", "Eglington")
# this will take the file, and wrangle it, extract
# short tail and long tail data, summarise per night per station (for the whole file/year)
# and summarise per station across the sampling period (year)
#
batsearch_summarise <- function(filepath, location) {
```

```
bat <- read.csv(filepath, na.strings = c("","NA"))
print(summary(bat))

bat1 <- bat |>
  filter(!is.na(Date)) |> # remove any obs which don't have a date
  mutate(Date     = dmy(Date),
         Year     = year(Date),
         Time     = case_when(is.na(Date) ~ NA,
                              .default = paste(hour(hms(Time)), minute(hms(Time)), sep = ":")),
         Datetime = case_when(is.na(Date) ~ NA,
                              .default = (ymd_hm(paste(Date, Time, sep = "_"), quiet = TRUE))),
         Nightof  = case_when(is.na(Date) ~ NA,
                              hour(hm(Time))<12 ~ Date - 1,
                              .default = Date),
         AssignedSite = as.factor(AssignedSite),
         Category     = as.factor(Category)) |>
  filter(Category %in% c("Both", "Short tail", "Long tail")) |>  ## remove all categories not in one of these three
  droplevels()

batlocation <- location
batyear <- bat1$Year[1]   # assign for use in output file name

## warning: some strings do not parse (NA in time or date column) + that is ok

# if there is a category = "Both" in the dataset, then duplicate those observations, and give one Category: "Short tail", and one "Long tail"
if(sum(bat1$Category == "Both") > 0) {
  bothsp <- bat1 |>
  filter(Category == "Both")

  short <- bothsp |>
    mutate(Category = "Short tail")

  long <- bothsp |>
    mutate(Category = "Long tail")
```

```
  print(paste(nrow(bothsp), "rows were labelled 'Both'. These rows have been copied for 'Long tail'
and 'Short tail' categories"))


  bat1 <- bat1 |>
  filter(Category %in% c("Short tail", "Long tail") ) |>
  bind_rows(short) |>
  bind_rows(long)  |>
  mutate(Category = as.factor(Category)) |>
  droplevels()}
# drop unused levels
# summarise calls per night and per site and per species category
# insert sites which did not have any counts
bat_bynight <- bat1 |>
  group_by(Year, AssignedSite, Nightof, Category) |>
  summarise(Calls = n()) |>
  ungroup() |>
  complete(Year, AssignedSite, Nightof, Category, fill = list(Calls = 0)) |>
  mutate(AssignedSite = as.character(AssignedSite)) |>
  arrange(Year, Category, Nightof, AssignedSite)
bat_byyear <- bat_bynight |>
  group_by(Year, AssignedSite, Category) |>
  summarise(Calls_mean = mean(Calls, na.rm = TRUE),
            Nights_n = n(),
            Calls_se = sd(Calls, na.rm = TRUE)/sqrt(Nights_n)) |>
  arrange(Year, Category, as.character(AssignedSite))


## QA CHECK:
## ensure all species, sites and nights are present in the dataset
## so that counts are summarised as expected
## also needed for (including zeroes) to be generated as expected
## if zero calls are logged on a station across all nights,
## this is not captured in the data!!
## if at least 1 call at a station, then zeroes created for the other nights / categories etc


(uniqueNights = unique(bat1$Nightof[!is.na(bat1$Nightof)]))
message(paste0("Check number of unique nights = ", length(uniqueNights),
               ": Unique nights = "))
print(uniqueNights)
```

```
(uniqueSites = unique(bat1$AssignedSite[!is.na(bat1$AssignedSite)]) )
message(paste0("Check number of site = ", length(uniqueSites),
           ": Unique sites = "))
print(as.character(uniqueSites))

path = paste0(location, "_", batyear, ".xlsx")

if (file.exists(path)) {
  message(paste(path, " already exists! - rename path or delete file and try again"))
  stop()}  else {
  message(paste0("summary by night&station and by station saved to ", path)) }
writexl::write_xlsx(list(year = bat_byyear, bynight = bat_bynight), path = path)
}
batsearch_summarise("allbats2023.csv","Eglinton")
```

Appendix D – analysing the data over the years to detect a trend

Code written by Graeme Elliott (DOC) and modified by Moira Pryde (DOC)

```
rm(list=ls())
library(data.table)   #Better than data frames for subsetting and working with data
library(reshape2)
library(lattice)
library(glmmTMB)
library(ggplot2)
library(car)
library(zoo)
library(boot)
library(MASS)
library(gamm4)
library(AICcmodavg)
library(simr)
setwd("C:/mpryde/bat detectors/Eglinton bats acoustic")
dat<-read.table("eglong.csv",
          header=TRUE, sep=",", na.strings=c("NA")
          ,colClasses = c("numeric","numeric","numeric","numeric"))
View(dat)
list(dat)
summary(dat)
table(dat$Station)
table(dat$Year)
table(dat$Day)
dat$year_day <- paste(dat$Year, dat$Day)
dat<-na.omit(dat)
model1 <- glmmTMBB(Passes ~ Year  + (1|year_day) + (1|Station), data=dat, family=nbinom2)
summary(model1)
#use model 1 for numeric
modelB <- glmmTMB(Passes ~ 1 + (1|year_day) + (1|Station), data=dat, family=nbinom2)
summary(modelB)

anova(model1,modelB,test="Chisq")
# the results of this show whether there is a significant trend over the years
```

```
dat<-read.table("eglong.csv",
          header=TRUE, sep=",", na.strings=c("NA")
          ,colClasses = c("factor","numeric","numeric","numeric"))
#use this for factor
modelA <- glmmTMB(Passes ~ Year -1 + (1|year_day) + (1|Station), data=dat, family=nbinom2)
summary(modelA)
modelB <- glmmTMB(Passes ~ 1 + (1|year_day) + (1|Station), data=dat, family=nbinom2)
summary(modelB)
anova(modelA,modelB,test="Chisq")
##this tests whether there is a significant difference between years
# this looks at Year as a factor to give some actual results to use
u95<-exp(summary(modelA)$coefficients$cond[,1]+1.96*summary(modelA)$coefficients$cond[,2])
mean<-exp(summary(modelA)$coefficients$cond[,1])
l95<-exp(summary(modelA)$coefficients$cond[,1]-1.96*summary(modelA)$coefficients$cond[,2])
the.predictions2<-cbind(l95,mean,u95)
the.predictions2
View(the.predictions2)
plot(the.predictions2)
```

Appendix E – an example using the code from Appendix D

To test whether there is a trend over time compare a model with Year (model 1) and one without Year (Model B)

Example – in this case the year model is significantly better – this is because the AIC is smaller for model 1 with a significance code of *** which means that the model is **very** sure that it is true

This is the output from the Rcode

```
> anova(model1,modelB,test="Chisq")
Data: dat
Models:
modelB: Passes ~ 1 + (1 | year_day) + (1 | Station), zi=~0, disp=~1
model1: Passes ~ Year + (1 | year_day) + (1 | Station), zi=~0, disp=~1
          Df    AIC    BIC       logLik   deviance  Chisq Chi    Df
modelB  4 5020.1 5039.2 -2506.0   5012.1
model1  5 4996.5 5020.4 -2493.3   4986.5      25.543       1
     Pr(>Chisq)
modelB
model1  4.326e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

If the model with Year has a lower AIC (Akaikes Information Criteria) and the difference is significant ($p<0.05$) then there is a trend. To calculate the average pass rate per year for output graphs the Year needs to be changed to a factor so that you get a mean result for every year.

Example output file

	l95	mean	u95
Year1	0.5611068	1.166390	2.424611
Year2	2.2245864	4.579513	9.427342
Year3	2.5657950	5.311228	10.994308
Year4	3.5826859	7.311509	14.921254