Rules of Thumb for Species Occupancy Models

Mark Logie & Tom August

07 February, 2019

Contents

Built with 3.4.1

Introduction

The aim of this document is to start development for rules of thumb around occupancy models. Which species can we make reliable models for? How many records do we need? How large an area? how many visits? Having a greater understanding of how these variables effect the quality of the models produced will allow us to develop rules of thumb for which species can be modelled and at which scale. This will also help us to identify key data gaps.

Metrics

To develop these rules of thumb we need two sets of metrics.

Data quality metrics

The metrics describe the data that we have for a species and can be used to predict the quality of the model:

- The 50th/90th percentile of the number of records of the focal species per year [`median/P90`]
- The 50th/90th percentile of the number of visits to a site each year, for sites where the species has been observed in that year (i.e. including visits where the focal species was not recorded) [`visits_median/P90`]
- The proportion of all site:year combinations, where the focal species is observed, that have > 1 visit [`prop_repeats_grp`]
- Considering all the lists where the focal species was recorded, the proportion of lists that had length 1 (i.e. records only of the focal species) [`prop_list_one`]
- Considering all visits for the 'taxonomic' group in the dataset, the proportion of all visits that did not record the focal species [`prop_abs`]
- Considering the visits for the 'taxonomic' group where the focal species was not observed, this is the proportion of visits with list length > 1 [`prop_abs_list`]*

*Note: this variable is excluded in the final decision trees. See 'Classifying the data: Examining the classification variables'

Model quality metrics

These metrics measure the quality of the model produced and are what we will use to develop the rules of thumb:

- The precision of the trend where `Precision <- 1/(sd(long_term_gr)^2)` and `long_term_gr` is a vector of annual growth rates where the `length()` of long_term_gr is equal to the number of model iterations [`precision_growth_rate`]
- Average precision of the year estimates where precision is defined in the same way as above [`mean_year_precision`]
- Proportion of years with converged Rhat [`PropYrConverged`]

Data

The data are in a few different locations and in most cases we need to go to more than one data source to get both sets of metrics. There is also a difference it the location of outputs, some are on Charlie's Cirrus folder, some her JASMIN folder and some are on LOTUS still.

Read in CIRRUS outputs

First, set up the location of the files to be read in

```
files_are_here <-
  "W:/PYWELL_SHARED/Pywell Projects/BRC/Charlie/1.c. New Model Rerun/4. Outputs/CIRRUS"

rdata_files <- list.files(files_are_here,
                          pattern = '.rdata$',
                          recursive = TRUE,
                          full.names = TRUE)
```

```
# The files we want are those in the 20000_update folder
rdata_files <- rdata_files[grepl('20000_update', rdata_files)]
```

Count up the number of files and set up some variables

```
num_files <- length(rdata_files)
speciesName <- quant25 <- Spnsite <- Spnvisits <- nyears <-
  PropYrConverged <- Totalnvisits <- Totalnsites <- FirstYrConverged <-
  LastYrConverged <- c(rep('',num_files*2))
```

Get what data we can from the model outputs

```
plot_species <- function(new_data, x, species, main = ''){

  new_data <- as.data.frame(new_data)
  new_data[,'year'] <-
    (Year = (x$min_year - 1) +
       as.numeric(gsub("psi.fs", "",gsub("\\[|\\]","",row.names(new_data)))))

  # rename columns, otherwise ggplot doesn't work properly
  names(new_data) <- gsub("2.5%","quant_025", names(new_data))
  names(new_data) <- gsub("97.5%","quant_975", names(new_data))

  # Add rhat T/F column
  new_data$rhat_threshold[new_data$Rhat < 1.1] <- 'Good (<1.1)'
  new_data$rhat_threshold[new_data$Rhat > 1.1] <- 'Bad (>1.1)'

  # plot the yearly predicted proportion of occupied sites
  # plot with error bars based on 95CI
  ggplot(new_data, aes_string(x = "year", y = "mean")) +
    theme_bw() +
    geom_ribbon(data = new_data,
                aes_string(group = 1, ymin = "quant_025", ymax = "quant_975"),
                alpha = 0.2) +
    geom_line(size = 1, col = "black") +
    geom_point(size = 4, aes(col = rhat_threshold)) +
    scale_color_manual(name = 'Rhat',
                       values = c('Bad (>1.1)' = 'red','Good (<1.1)' = 'blue')) +
    ylab("Occupancy") +
    xlab("Year") +
    scale_y_continuous(limits = c(0, 1)) +
    ggtitle(main) +
    theme(plot.title = element_text(lineheight = .8, face = "bold"),
          legend.position = 'bottom')
  ggsave(filename = paste0('metrics/plots/', species, '.png'),
         width = 6.71, height = 4.64)
}

for(i in 1:num_files){
  # Load the file
  load(rdata_files[i])
  # Load the summary table
  bugsSummary <- out$BUGSoutput$summary

  ## ~~~ Useful information ~~~
```

```
  speciesName[i] <- out$SPP_NAME

  ## ~~~ Explanatory variables ~~~
  # Number of sites species is observed at
  Spnsite[i] <- out$species_sites
  # Number of visits on which the species is observed
  Spnvisits[i] <- out$species_observations
  # Number of years
  nyears[i] <- length(out$max_year:out$min_year)
  # Total visits
  Totalnvisits[i] <- out$nvisits
  # Total sites
  Totalnsites[i] <- out$nsites
  ## Rhats ##
  yearValues <- bugsSummary[grepl('^psi.fs\\[', rownames(bugsSummary)), ]
  # Proportion of years converged
  PropYrConverged[i] <- sum(yearValues[, 'Rhat'] < 1.1) / nrow(yearValues)
  # First/Last year converged?
  FirstYrConverged[i] <- yearValues[, 'Rhat'][1] < 1.1
  LastYrConverged[i] <- tail(yearValues[, 'Rhat'], 1) < 1.1

  plot_species(yearValues, out, speciesName[i])

  rm(list = c('out'))

  cat(paste0('First pass - Finished processing file ',i,' of ',num_files, '\n'))
}

# Now do just the last 10 years
for(i in 1:num_files){
  # Load the file
  load(rdata_files[i])

  # Load the summary table
  bugsSummary <- out$BUGSoutput$summary

  ## ~~~ Useful information ~~~
  speciesName[i+num_files] <- paste0(out$SPP_NAME, '_last10yrs')

  nyears[i+num_files] <- 10
  ## Rhats ##
  yearValues <- bugsSummary[grepl('^psi.fs\\[', rownames(bugsSummary)), ]
  # subset to last 10 years
  yearValues <- tail(yearValues, 10)
  # Proportion of years converged
  PropYrConverged[i+num_files] <- sum(yearValues[, 'Rhat'] < 1.1) / nrow(yearValues)
  # First/Last year converged?
  FirstYrConverged[i+num_files] <- yearValues[, 'Rhat'][1] < 1.1
  LastYrConverged[i+num_files] <- tail(yearValues[, 'Rhat'], 1) < 1.1

  plot_species(yearValues, out, speciesName[i+num_files])

  rm(list = c('out'))
```

```
  cat(paste0('Second pass - Finished processing file ',i,' of ',num_files, '\n'))
}

Spnsite   <- as.numeric(Spnsite)
Spnvisits <- as.numeric(Spnvisits)
quant25 <- as.numeric(quant25)
nyears <- as.numeric(nyears)
Totalnvisits <- as.numeric(Totalnvisits)
PropYrConverged <- as.numeric(PropYrConverged)

SpvisitPerSite <- Spnvisits/Spnsite
SpvisitPerSite[is.na(SpvisitPerSite)] <- 0

SpvisitPerYear <- Spnvisits/nyears
SpvisitPerYear[is.na(SpvisitPerYear)] <- 0

TotalvisitPerYear <- Totalnvisits/nyears
TotalvisitPerYear[is.na(TotalvisitPerYear)] <- 0

SpeciesData <- data.frame(speciesName, Spnsite, Spnvisits, SpvisitPerSite,
                          SpvisitPerYear, Totalnvisits, Totalnsites,
                          TotalvisitPerYear, nyears, FirstYrConverged,
                          LastYrConverged, PropYrConverged)
```

```
write.csv(SpeciesData,
          file = 'Results/metrics/model_data.csv', row.names = FALSE)
```

```
model_data <- read.csv('Results/metrics/model_data.csv')
str(model_data)
```

```
## 'data.frame':    21824 obs. of  8 variables:
##  $ speciesName     : Factor w/ 21824 levels "Abrothallus bertianus",..: 11747 11749 11751 11753 1175
##  $ FirstYrConverged: logi  FALSE TRUE TRUE TRUE TRUE TRUE ...
##  $ LastYrConverged : logi  FALSE TRUE TRUE TRUE TRUE TRUE ...
##  $ PropYrConverged : num  0 1 1 1 1 1 1 0 1 1 ...
##  $ mean_means      : num  0.02879 0.17826 0.00563 0.42023 0.12572 ...
##  $ median_means    : num  0.02787 0.19084 0.00504 0.41488 0.11193 ...
##  $ mean_medians    : num  0.0232 0.1763 0.00483 0.41816 0.1246 ...
##  $ median_medians  : num  0.02214 0.18983 0.00451 0.41451 0.11152 ...
```

```
head(model_data)
```

```
##           speciesName FirstYrConverged LastYrConverged PropYrConverged
## 1   FORMICA aquilonia            FALSE           FALSE               0
## 2 FORMICA cunicularia             TRUE            TRUE               1
## 3     FORMICA exsecta             TRUE            TRUE               1
## 4       FORMICA fusca             TRUE            TRUE               1
## 5      FORMICA lemani             TRUE            TRUE               1
## 6    FORMICA lugubris             TRUE            TRUE               1
##    mean_means median_means mean_medians median_medians
## 1 0.028790722  0.027866278  0.023195764    0.022140221
## 2 0.178260764  0.190840130  0.176301763    0.189831898
## 3 0.005626754  0.005036095  0.004832814    0.004510045
## 4 0.420225682  0.414878580  0.418160565    0.414514145
## 5 0.125723942  0.111931324  0.124597629    0.111521115
```

```
## 6 0.026551388  0.023028410  0.026153027    0.022550226
```

Posteriors

We can also get data on the trend estimate from the 1000 posteriors that Charlie has already calculated. This is for all species groups regardless of whether they were run on Cirrus or JASMIN. We have to remove some years full of NAs. This is where the model is run to a year where there is no data. As a consequence some of the 'final 10 years' runs will actually be for a shorter period.

```
path_to_posteriors <-
  file.path('W:/PYWELL_SHARED/Pywell Projects/BRC/Charlie/1.c. New Model Rerun',
            '6. Indicators and other analyses/1000 posterior samples')

rdata_files <- list.files(path = tolower(path_to_posteriors),
                          pattern = '.rdata$',
                          full.names = TRUE)

annual_growth_rate <- function(years){
    years <- na.omit(years)
    (((tail(years, 1)/years[1])^(1/length(years)))-1)*100
  }

modelposterior <- function(sp, mat_j_post, sp_list, suffix = NULL){

  cat('Modelling', sp, '...\n')

    gr1000 <- apply(X = mat_j_post[sp_list == sp, ],
                    MARGIN = 1,
                    FUN = annual_growth_rate)

    mean_growth_rate <- mean(gr1000)
    precision_growth_rate <- 1/(sd(gr1000)^2)

    yrprec <- apply(X = mat_j_post[sp_list == sp, ],
                    MARGIN = 2,
                    FUN = function(x){
                      1/(sd(x)^2)
                    })

    mean_year_precision <- mean(yrprec, na.rm = TRUE)
    return(data.frame(species = paste0(sp, suffix),
                      mean_growth_rate,
                      precision_growth_rate,
                      mean_year_precision))
}

for(data_file in rdata_files){

  # data_file <- rdata_files[1]
  cat('Starting', basename(data_file))

  # Charlie has named some of the objects differently, so I deal with this
  if(exists('j_post')) rm(list = 'j_post')
  if(exists('samp_post')) rm(list = 'samp_post')
```

```r
load(data_file)
if(exists('j_post')){
  samp_post <- j_post
  rm(list = 'j_post')
}

# sp_list <- j_post$spp
sp_list <- samp_post$spp

spp <- unique(sp_list)

cat('\n', length(spp), 'species\n')

# mat_j_post <- as.matrix(j_post[, colnames(j_post)[!colnames(j_post) %in% c('iter', 'spp')]])
mat_j_post <-
  as.matrix(samp_post[, colnames(samp_post)[!colnames(samp_post) %in% c('iter', 'spp')]])

## NOTE ##
# Some of the posterior files contain data for years that the model was not run for
# these appear here as columns of entirely NA values. These should be removed
good_columns <- apply(mat_j_post, MARGIN = 2, FUN = function(x) !all(is.na(x)))
if(any(!good_columns)){
  warning('Removing NA year(s) in ', basename(data_file), ' : ',
          paste(names(good_columns[!good_columns]), collapse = ', '))
  mat_j_post <- mat_j_post[,good_columns]
}

# set year estimates of 0 to 0.0001 to avoid infinite growth rates
mat_j_post[mat_j_post == 0] <- 0.0001
# rm(list = 'j_post')
rm(list = 'samp_post')

df <- lapply(spp, FUN = modelposterior, mat_j_post = mat_j_post, sp_list = sp_list)
df <- do.call(rbind, df)
row.names(df) <- 1:nrow(df)
write.csv(df,
          row.names = FALSE,
          file = file.path('Results/metrics',
                           paste0('posteriorLM_',
                                  gsub('.rdata$', '',
                                       basename(data_file)),
                                  '.csv')))

df <- lapply(spp, FUN = modelposterior,
             mat_j_post = mat_j_post[,(ncol(mat_j_post)-9):ncol(mat_j_post)],
             sp_list = sp_list, suffix = '_last10yrs')
df <- do.call(rbind, df)
row.names(df) <- 1:nrow(df)
write.csv(df, file = file.path('Results/metrics',
                           paste0('posteriorLM_last10yrs_',
                                  gsub('.rdata$', '',
                                       basename(data_file)),
                                  '.csv')))
```

```
}
```

Once we have the data for all species we can add the data together into one file

```
LMfiles <- list.files(path = 'Results/metrics',
                      pattern = '^posteriorLM',
                      full.names = TRUE)

master <- NULL

for(i in LMfiles){

  x <- read.csv(i)
  master <- rbind(master, x[,c('species',
                               'mean_growth_rate',
                               'precision_growth_rate',
                               'mean_year_precision')])

}

write.csv(master, file = 'Results/metrics/ALL_posteriorLM.csv',
          row.names = FALSE)
```

Raw data files

There are some additional stats we can get from the raw data file. Again, these are for all groups not just CIRRUS/JASMIN runs. Given the memory requirements needed to reformat the plant and moth data they have been removed them from this step and will be incorporated after producing the classification trees.

```
dataMetrics <- function(sp, basen, species_obs, hab=NULL,
                        habitat=FALSE, region=FALSE, suffix=NULL, years = FALSE){

  cat(as.character(sp),'... ')

  timetaken <- system.time({
    if(habitat){
      tFDall <- species_obs[(species_obs[,paste0(hab,'_hab')]),]
    } else if(region){
      tFDall <- species_obs[species_obs$nutsname==hab,]
    } else {
      tFDall <- species_obs
    }
    tFD <- tFDall[tFDall[,names(tFDall) == sp],]

    # Extract taxonomic name
    tax_root <- substr(basen,start = 1,stop = (regexpr('_17',basen)[1]-1))

    if(!is.null(years)){
      # There is a limit to the number of year's data we want
      # Set taxonomic name with this information included
      tax_nom <- paste0('last',years,'yrs',tax_root)
      #  Drop everything we don't want
      if(nrow(tFD)!=0){
        boundingyear <- max(tFD$year)-years
```

```
      tFD <- tFD[tFD$year>boundingyear,]
    }
    else
    {
      # There's no observances for this species, so set a bounding year
      # based on the group data
      boundingyear <- max(tFDall$year)-years
    }
    tFDall <- tFDall[tFDall$year>boundingyear,]
  } else {
    tax_nom <- tax_root
  }

  if(nrow(tFD)==0){
  # No data, so set a few variables to 0.  The reason for using 0's as defaults is if
  # there is no data, these parameters should be 0 e.g. median number of visits = 0.
  # Without this default, many parameters return NA, which is less accurate.
    nyears <- Spnsite <- Spnvisits <- SpvisitPerSite <- SpvisitPerYear <-
      avVisitPerYear <- median <- P70 <- P80 <- P90 <- sdVisitsPerYear <-
      coeffVar <- zmedian <- zP70 <- zP80 <- zP90 <- zsdVisitsPerYear <-
      zcoeffVar <- visits_median <- visits_P70 <- visits_P80 <- visits_P90 <-
      prop_repeats_spc <- prop_repeats_grp <- prop_of_years <- prop_list_one <-
      prop_abs_list <- Totalnvisits <- Totalnsites <- TotalvisitPerYear <- 0
    prop_abs <- 1
  } else {
    # range of years with data
    nyears <- (1+max(tFD$year)-min(tFD$year))
    # Number of sites species is observed at
    Spnsite <- length(unique(tFD$site))
    # Number of visits on which the species is observed
    Spnvisits <- length(unique(tFD$visit))
    # Number of visits per site
    SpvisitPerSite <- Spnvisits/Spnsite
    # Number of visits per year
    SpvisitPerYear <- Spnvisits/nyears

    # average visits per year
    visits_year <- tapply(tFD$visit, tFD$year,
                          FUN = function(x) length(unique(x)))
    avVisitPerYear = mean(visits_year)

    # find 70, 80 and 90th percentiles
    percentiles <- quantile(visits_year,c(.5,.7,.8,.9))
    median <- as.numeric(percentiles[1])
    P70 <- as.numeric(percentiles[2])
    P80 <- as.numeric(percentiles[3])
    P90 <- as.numeric(percentiles[4])

    if(nrow(tFD)==1){
      sdVisitsPerYear <- coeffVar <- 1
    } else {
      # sd visits per year
      sdVisitsPerYear = sd(visits_year)
```

```
  # coefficient of variation
  coeffVar <- sd(visits_year)/mean(visits_year)
}

unique_years <- unique(tFD$year)
all_years    <- min(tFDall$year):max(tFDall$year)
missing_years <-  all_years[!(all_years %in% unique_years)]
visits_year <- c(visits_year,rep(0,length(missing_years)))

# find 70, 80 and 90th percentiles
percentiles <- quantile(visits_year,c(.5,.7,.8,.9))
zmedian <- as.numeric(percentiles[1])
zP70 <- as.numeric(percentiles[2])
zP80 <- as.numeric(percentiles[3])
zP90 <- as.numeric(percentiles[4])

# sd visits per year
zsdVisitsPerYear = sd(visits_year)
# coefficient of variation
zcoeffVar <- sd(visits_year)/mean(visits_year)

# repeat visits
# Within each year get the counts of visits to each location
repeats <- count(tFD, site, year)
repeats$concat <- paste0(repeats$site,'-',repeats$year)

# What proportion of these are > 1
prop_repeats_spc <- sum(repeats$n > 1) / nrow(repeats)

# visits within the group for visits to each location within a year
group_repeats <- count(tFDall, site, year)
group_repeats$concat <-
  paste0(group_repeats$site,'-',group_repeats$year)

# Find which of these group visits are to sites in years where the
# species of interest was observed
site_year <- group_repeats$concat %in% repeats$concat
group_repeats <- group_repeats$n[site_year]
prop_repeats_grp <- sum(group_repeats > 1) / length(group_repeats)

# all visits to a site in a year where there was at least one observance
# of species of interest
percentiles <- quantile(group_repeats,c(.5,.7,.8,.9))
visits_median <- as.numeric(percentiles[1])
visits_P70 <- as.numeric(percentiles[2])
visits_P80 <- as.numeric(percentiles[3])
visits_P90 <- as.numeric(percentiles[4])

# proportion of years with data
prop_of_years <-
  length(unique(tFD$year))/(1+max(tFDall$year)-min(tFDall$year))

# lists of length 1
```

```
  prop_list_one <- sum(tFD$L == 1) / nrow(tFD)
}

# proportion of records for the group which did not observe this species
prop_abs <- (nrow(tFDall)-nrow(tFD))/(nrow(tFDall))

# proportion of non-observances with list length > 1
tFDabs <- tFDall[!tFDall[,colnames(tFDall) %in% sp],]
prop_abs_list <- sum(tFDabs$L > 1) / nrow(tFDabs)

# Number of sites for taxonomic group
Totalnsites <- length(unique(tFDall$site))
# Number of visits for taxonomic group
Totalnvisits <- nrow(tFDall)
if(nrow(tFDall)==0){
  total_years <- 0
  TotalvisitPerYear <- 0
} else {
  total_years <- length(min(tFDall$year):max(tFDall$year))
  TotalvisitPerYear <- Totalnvisits/total_years
}

df <- data.frame(species = paste0(sp, suffix),
                 habitat = NA,
                 region = NA,
                 avVisitPerYear = avVisitPerYear,
                 median = median,
                 P70 = P70,
                 P80 = P80,
                 P90 = P90,
                 sdVisitsPerYear = sdVisitsPerYear,
                 coeffVar = coeffVar,
                 zmedian = zmedian,
                 zP70 = zP70,
                 zP80 = zP80,
                 zP90 = zP90,
                 zsdVisitsPerYear = zsdVisitsPerYear,
                 zcoeffVar = zcoeffVar,
                 prop_repeats_spc = prop_repeats_spc,
                 prop_repeats_grp = prop_repeats_grp,
                 visits_median = visits_median,
                 visits_P70 = visits_P70,
                 visits_P80 = visits_P80,
                 visits_P90 = visits_P90,
                 prop_of_years = prop_of_years,
                 prop_list_one = prop_list_one,
                 prop_abs = prop_abs,
                 prop_abs_list = prop_abs_list,
                 nyears = nyears,
                 Spnsite = Spnsite,
                 Spnvisits = Spnvisits,
                 SpvisitPerSite = SpvisitPerSite,
                 SpvisitPerYear = SpvisitPerYear,
```

```
                      Totalnsites = Totalnsites,
                      Totalnvisits = Totalnvisits,
                      TotalvisitPerYear = TotalvisitPerYear,
                      Taxa_Root = tax_root,
                      Taxa = tax_nom,
                      stringsAsFactors = FALSE)
    if(habitat){
      df[, 'habitat'] <- hab
    } else if(region){
      df[, 'region'] <- hab
    }
  })
  cat(as.numeric(timetaken[3]), 'seconds', '\n')
  return(df)
}

data_files_path <- file.path('W:/PYWELL_SHARED/Pywell Projects/BRC/Charlie',
                             '1.c. New Model Rerun/1. Data/Cleaned Datasets')

data_files <- list.files(data_files_path,
                         pattern = '.rdata$',
                         full.names = TRUE)

# Dont do the really big ones.  These will be interpreted later.
data_files <- data_files[!grepl('Moths', data_files)]
data_files <- data_files[!grepl('VascPlants', data_files)]

dataPrep <- function(data_file){
  cat('Starting', basename(data_file))

  load(data_file)

  if('SQ_1KM' %in% colnames(taxa_data)){
    names(taxa_data)[names(taxa_data) == 'SQ_1KM'] <- 'TO_GRIDREF'
  }

  # Filter the data as it is for the occupancy models
  formatted_data <- formatOccData(taxa = taxa_data$CONCEPT,
                                  site = taxa_data$TO_GRIDREF,
                                  time_period = taxa_data$TO_STARTDATE)
  return(formatted_data)
}

for(data_file in data_files){
  formatted_data <- dataPrep(data_file)

  spp <- unique(taxa_data$CONCEPT)
    # one group has a species called '', best to remove this (single) record
  spp <- spp[spp != '']

  cat('\n', length(spp), 'species\n')

  tFDall <- merge(formatted_data$occDetdata,
```

```
                  formatted_data$spp_vis)

  # Remove sites which have not seen a repeat in subsequent years.
  # This is a requirement here as this was a step taken by the model.
  yps <- rowSums(acast(tFDall, site ~ year, length, value.var = 'L') > 0)
  sites_to_include <- names(yps[yps >= 2])
  tFDall <- tFDall[as.character(tFDall$site) %in% sites_to_include,]

  raw_metrics <- lapply(spp,
                        FUN = dataMetrics,
                        species_obs = tFDall,
                        basen = basename(data_file))

  raw_metrics <- do.call(rbind, raw_metrics)

  write.csv(raw_metrics, file = file.path('Results/metrics',
                                          paste0('rawMetrics_',
                                                 gsub('.rdata$', '',
                                                      basename(data_file)),
                                                 '.csv')),
            row.names = FALSE)

  # now with just the last 10 years
  cat('\n', length(spp), 'species\n')
  raw_metrics <- lapply(spp,
                        FUN = dataMetrics,
                        tFDall = tFDall,
                        basen = basename(data_file),
                        suffix = '_last10yrs',
                        years = 10)

  raw_metrics <- do.call(rbind, raw_metrics)

  write.csv(raw_metrics, file = file.path('Results/metrics',
                                          paste0('rawMetrics_last10yrs',
                                                 gsub('.rdata$', '',
                                                      basename(data_file)),
                                                 '.csv')),
            row.names = FALSE)
}
```

Once we have the data for all species we can add the data together into one file

```
rawdataFiles <- list.files(path = 'Results/metrics',
                       pattern = '^rawMetrics',
                       full.names = TRUE)

master <- NULL

for(i in rawdataFiles){

  x <- read.csv(i)
  master <- rbind(master, x)
```

```
}

write.csv(master, file = 'Results/metrics/ALL_rawMetrics.csv',
          row.names = FALSE)
```

Classification of Models

Create metrics

We can do some simple decision tree statistics to develop rules of thumb once we have decided what constitutes a 'bad' model. Here I choose some metrics that make for a bad model and use the `rpart` package to create a decision tree.

First, combine all the data

```
RM <- read.csv('Results/metrics/ALL_rawMetrics.csv')
LM <- read.csv('Results/metrics/ALL_posteriorLM.csv')
MM <- read.csv('Results/metrics/model_data.csv')

# Remove repeat variables from model data
MM <- MM[,c('speciesName','FirstYrConverged',
            'LastYrConverged','PropYrConverged',
            'mean_means')]

# These merges drop a few species that don't match. That is OK as the ones that don't
# match do not meet the data minimum criteria or we do not have model data for them, so
# cannot use them for predictions.
trendsData <- merge(x = LM, y = RM,
                    by.x = 'species',
                    by.y = 'species')
trendsData <- merge(x = MM, y = trendsData,
                    by.x= 'speciesName', by.y = 'species')
head(trendsData)
```

```
##                         speciesName FirstYrConverged LastYrConverged
## 1              Abrothallus bertianus               NA              NA
## 2    Abrothallus bertianus_last10yrs               NA              NA
## 3           Abrothallus caerulescens               NA              NA
## 4 Abrothallus caerulescens_last10yrs               NA              NA
## 5              Abrothallus cetrariae               NA              NA
## 6    Abrothallus cetrariae_last10yrs               NA              NA
##   PropYrConverged mean_means mean_growth_rate precision_growth_rate
## 1              NA 0.13435213        1.5362514            0.19439949
## 2              NA 0.19201512        1.1963530            0.03742833
## 3              NA 0.18045817        3.0448637            0.03252791
## 4              NA 0.12971493        0.6296349            0.02091444
## 5              NA 0.09710464       -0.6514195            0.19909366
## 6              NA 0.12577490        2.3128789            0.01950330
##   mean_year_precision avVisitPerYear Spnvisits Spnsite SpvisitPerSite
## 1            57.15791       1.846154        24      24              1
## 2            29.41260       2.000000        14      14              1
## 3            16.18282       1.000000         3       3              1
## 4            27.37129       1.000000         3       3              1
## 5            35.33673       1.000000         4       4              1
```

```
## 6            33.61787       1.000000         2       2              1
##   SpvisitPerYear Totalnvisits Totalnsites TotalvisitPerYear median P70 P80
## 1      0.6000000        15545        4952          337.9348      1   3   3
## 2      1.5555556         5061        2479          460.0909      2   3   3
## 3      0.7500000        15545        4952          337.9348      1   1   1
## 4      0.7500000         8380        3740          441.0526      1   1   1
## 5      0.1025641        15545        4952          337.9348      1   1   1
## 6      0.6666667         5061        2479          460.0909      1   1   1
##   P90 sdVisitsPerYear  coeffVar zmedian zP70 zP80 zP90 zsdVisitsPerYear
## 1   3       0.9870962 0.5346771       0    0    1  2.5        0.9829464
## 2   3       1.0000000 0.5000000       1    2    3  3.0        1.2720778
## 3   1       0.0000000 0.0000000       0    0    0  0.0        0.2496374
## 4   1       0.0000000 0.0000000       0    0    0  1.0        0.3746343
## 5   1       0.0000000 0.0000000       0    0    0  0.0        0.2848849
## 6   1       0.0000000 0.0000000       0    0    0  1.0        0.4045199
##   zcoeffVar prop_repeats_spc prop_repeats_grp visits_median visits_P70
## 1 1.8839806                0        0.3333333           1.0        2.0
## 2 0.9994897                0        0.4285714           1.0        2.0
## 3 3.8277737                0        0.0000000           1.0        1.0
## 4 2.3726841                0        0.0000000           1.0        1.0
## 5 3.2761766                0        0.2500000           1.0        1.3
## 6 2.2248595                0        0.5000000           2.5        3.1
##   visits_P80 visits_P90 prop_of_years prop_abs_list  prop_abs
## 1        2.4        4.0    0.28260870     0.8302944 0.9984561
## 2        2.4        3.7    0.63636364     0.8080048 0.9972337
## 3        1.0        1.0    0.06521739     0.8305237 0.9998070
## 4        1.0        1.0    0.15789474     0.8235645 0.9996420
## 5        2.2        3.1    0.08695652     0.8305128 0.9997427
## 6        3.4        3.7    0.18181818     0.8084602 0.9996048
##   prop_list_one             Taxa Taxa_Root
## 1             0          Lichens   Lichens
## 2             0 last10yrsLichens   Lichens
## 3             0          Lichens   Lichens
## 4             0 last10yrsLichens   Lichens
## 5             0          Lichens   Lichens
## 6             0 last10yrsLichens   Lichens
```

```
str(trendsData)
```

```
## 'data.frame':    20070 obs. of  40 variables:
##  $ speciesName          : Factor w/ 21824 levels "Abrothallus bertianus",..: 1 2 3 4 5 6 7 8 9 10 ..
##  $ FirstYrConverged     : logi  NA NA NA NA NA NA ...
##  $ LastYrConverged      : logi  NA NA NA NA NA NA ...
##  $ PropYrConverged      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ mean_means           : num  0.1344 0.192 0.1805 0.1297 0.0971 ...
##  $ mean_growth_rate     : num  1.536 1.196 3.045 0.63 -0.651 ...
##  $ precision_growth_rate: num  0.1944 0.0374 0.0325 0.0209 0.1991 ...
##  $ mean_year_precision  : num  57.2 29.4 16.2 27.4 35.3 ...
##  $ avVisitPerYear       : num  1.85 2 1 1 1 ...
##  $ Spnvisits            : int  24 14 3 3 4 2 3 2 48 27 ...
##  $ Spnsite              : int  24 14 3 3 4 2 3 2 48 27 ...
##  $ SpvisitPerSite       : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ SpvisitPerYear       : num  0.6 1.556 0.75 0.75 0.103 ...
##  $ Totalnvisits         : int  15545 5061 15545 8380 15545 5061 15545 7151 15545 5061 ...
##  $ Totalnsites          : int  4952 2479 4952 3740 4952 2479 4952 3333 4952 2479 ...
```

```
##  $ TotalvisitPerYear    : num  338 460 338 441 338 ...
##  $ median               : num  1 2 1 1 1 1 1 1 1 3 ...
##  $ P70                  : num  3 3 1 1 1 1 1 1 3 4 ...
##  $ P80                  : num  3 3 1 1 1 1 1 1 4 4 ...
##  $ P90                  : num  3 3 1 1 1 1 1 1 4 4.6 ...
##  $ sdVisitsPerYear      : num  0.987 1 0 0 0 ...
##  $ coeffVar             : num  0.535 0.5 0 0 0 ...
##  $ zmedian              : num  0 1 0 0 0 0 0 0 0 2 ...
##  $ zP70                 : num  0 2 0 0 0 0 0 0 1 4 ...
##  $ zP80                 : num  1 3 0 0 0 0 0 0 2 4 ...
##  $ zP90                 : num  2.5 3 0 1 0 1 0 0.5 3.5 4 ...
##  $ zsdVisitsPerYear     : num  0.983 1.272 0.25 0.375 0.285 ...
##  $ zcoeffVar            : num  1.884 0.999 3.828 2.373 3.276 ...
##  $ prop_repeats_spc     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ prop_repeats_grp     : num  0.333 0.429 0 0 0.25 ...
##  $ visits_median        : num  1 1 1 1 1 2.5 2 1.5 1 2 ...
##  $ visits_P70           : num  2 2 1 1 1.3 3.1 2 1.7 2 3 ...
##  $ visits_P80           : num  2.4 2.4 1 1 2.2 3.4 2 1.8 2.6 3 ...
##  $ visits_P90           : num  4 3.7 1 1 3.1 3.7 2 1.9 3.3 4 ...
##  $ prop_of_years        : num  0.2826 0.6364 0.0652 0.1579 0.087 ...
##  $ prop_abs_list        : num  0.83 0.808 0.831 0.824 0.831 ...
##  $ prop_abs             : num  0.998 0.997 1 1 1 ...
##  $ prop_list_one        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ Taxa                 : Factor w/ 64 levels "Ants","AquaticBugs",..: 50 34 50 34 50 34 50 34 50 34
##  $ Taxa_Root            : Factor w/ 32 levels "Ants","AquaticBugs",..: 18 18 18 18 18 18 18 18 18 18
```

```
nrow(trendsData)
```

```
## [1] 20070
```

There are some species with no data in this dataset. We will remove these species.

```
trendsData$last10yr <- substring(trendsData$Taxa,1,8) == "last10yr"

sum(trendsData$P90==0)
```

```
## [1] 658
```

```
trendsData <- trendsData[trendsData$P90>0,]
```

Consultation on model outputs

Next, define which of the species have 'good' models and which are 'bad'.

This was done by consultation with 3 experts on the models, testing them on 100 models which were sampled to provide a spread of examples across the dataset, with a cluster of models focused around the range in which there was likely to be controversial.

```
# Lets have a look at the distribution
quantile(trendsData$precision_growth_rate)
```

```
##           0%          25%          50%          75%         100%
## 4.259870e-04 2.960039e-02 9.274431e-02 3.223825e-01 2.702034e+02
```

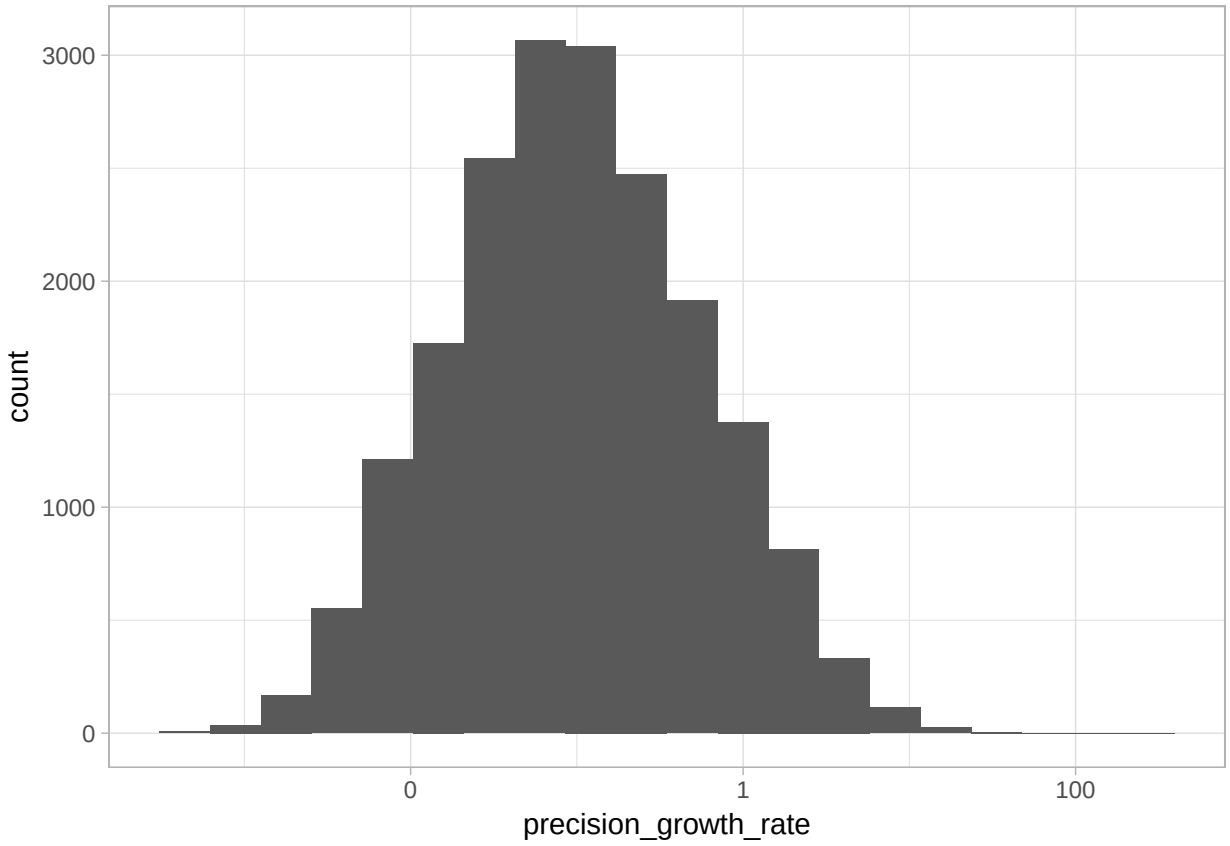


Figure 1: Histogram of the growth rate precision for all models

```
quantile(trendsData$mean_year_precision)
```

```
##           0%          25%          50%          75%         100%
## 4.907666e+00 2.901984e+01 2.456900e+02 1.704495e+03 1.378302e+07
```

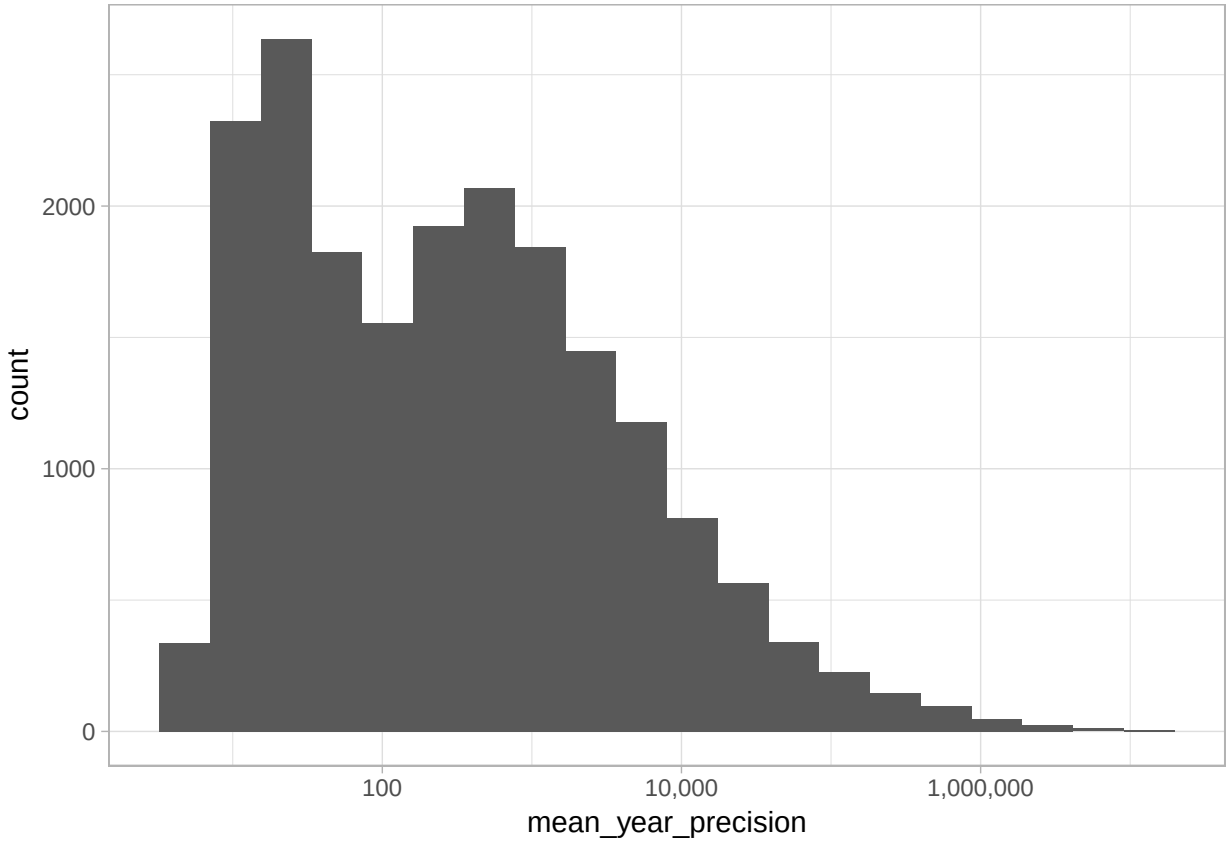


Figure 2: Histogram of the mean year precision for all models

Below are the results of the consultation. This graph shows the score given to each model where the score is the number of experts who thought the model was of good quality. The models not used in the assessment are shown in the background in grey

```
# Lets look at the results of the consultation with model experts
fsdf <- read.csv(file = 'Results/Consultation/fsdf_full.csv')
```

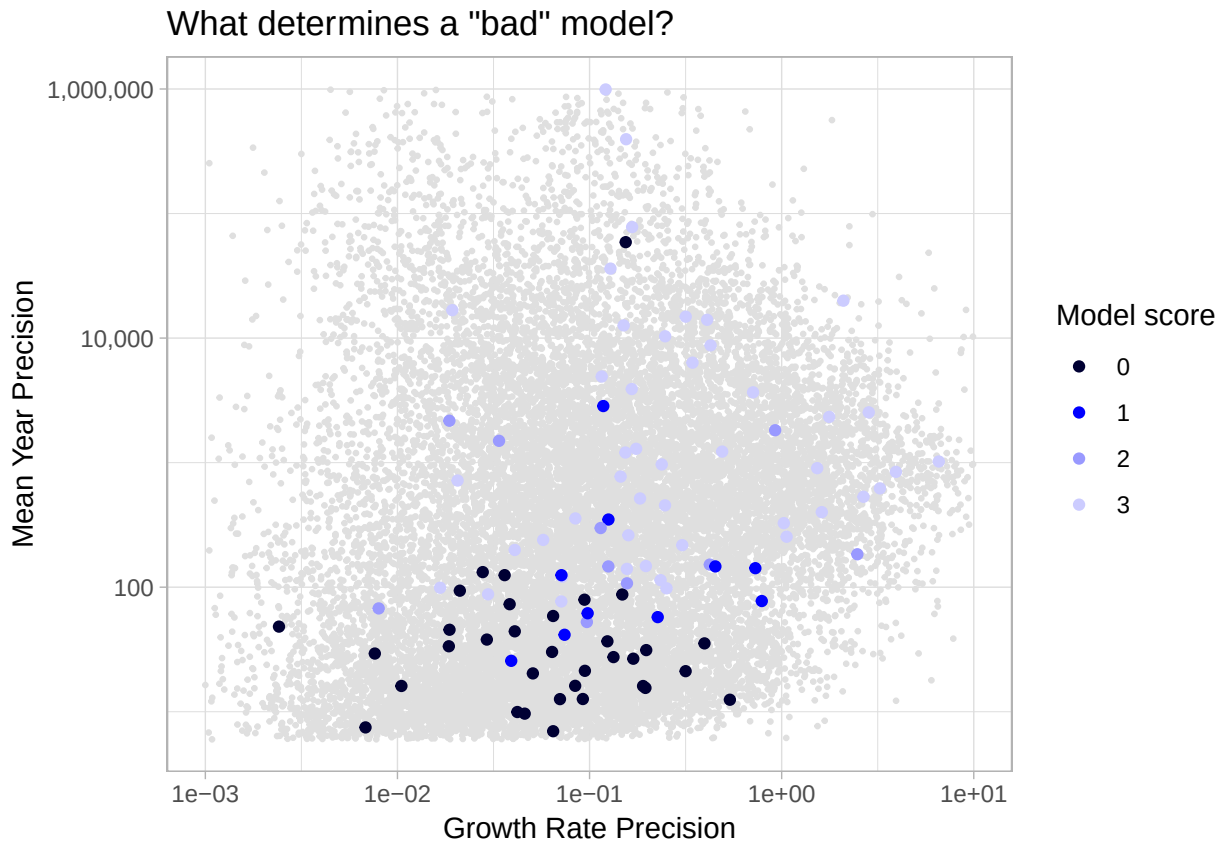


Figure 3: Graph showing results of consulation with model experts, plotting mean year precision vs growth rate precision

```
## Warning: Removed 4005 rows containing missing values (geom_point).
```

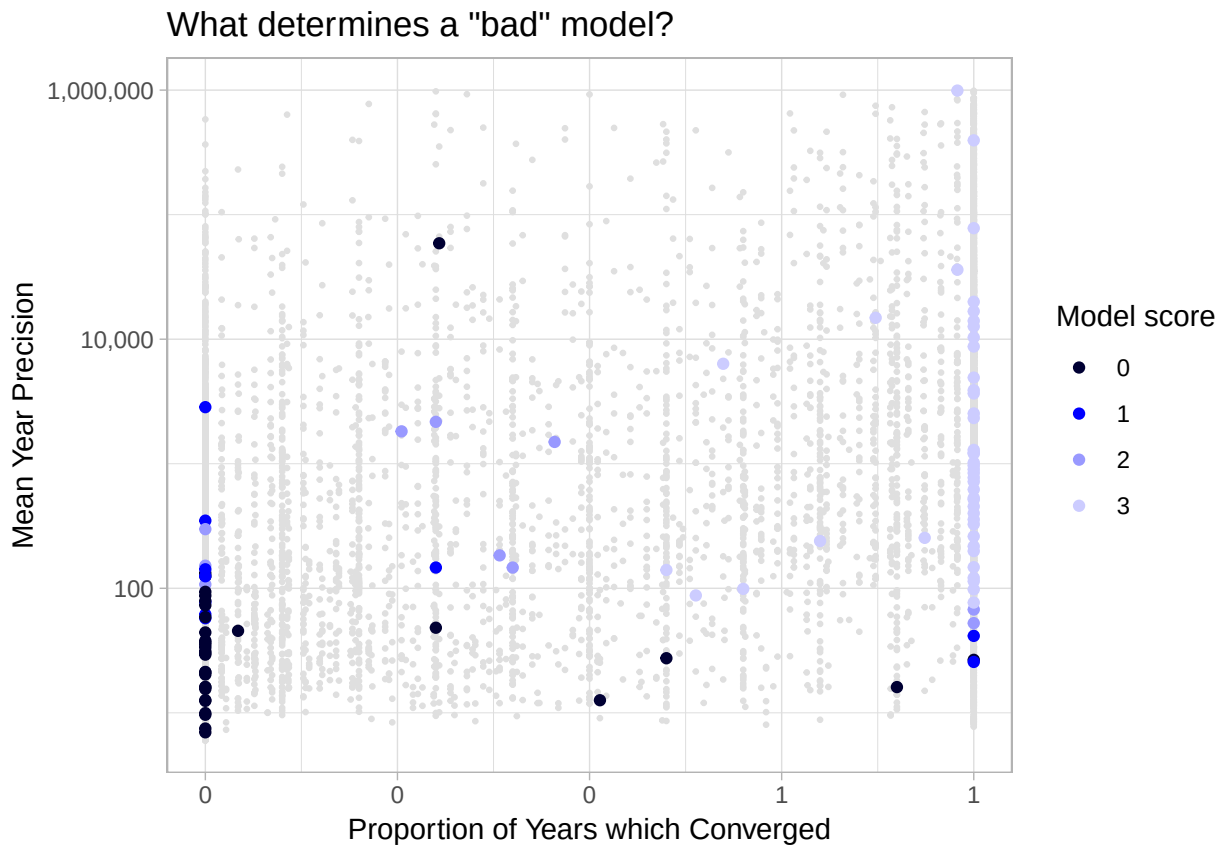


Figure 4: Graph showing results of consulation with model experts, plotting mean year precision vs the proportion of years which converged. Coloured points show the score assigned to models: light blue represents a score of 3 or unanimously a good model; black shows a score of 0, or unanimously a bad model. Light grey points in the background show all models which were not part of the consultation

The graphs above show the results of the consultation with experts on the model. There was unanimous agreement on 80% of the models about which were 'good' and 'bad'. The models chosen for this consultation were deliberately difficult to tell apart, so a 80% unanimous agreeement is considered very good.

Plotting these scores against proportion of years which converged, mean year precision and growth rate precision show that these parameters are correlated with quality, but it is not obvious how to perform this split.

Classification tree

To decide which models would be classified as good and bad, a decision tree was created to automatically classify the 100 models tested on. The models with a score of 2 or 3 were classified as good, while the models with a score of 0 or 1 were classified as bad.

Two decision trees are shown below, one using the proportion of years converged, and the other not, for comparison.

```
fsdf$good <- rep('bad',nrow(fsdf))
fsdf$good[fsdf$score>=2] <- 'good'
fsdf$good <- as.factor(fsdf$good)

fit_fsdf <- rpart(good ~ mean_year_precision + precision_growth_rate,
                  method = 'class',
                  data = fsdf)

rpart.plot(fit_fsdf, extra = 108, type = 3, clip.right.labs = FALSE)
```

```
fit_fsdf <- rpart(good ~ mean_year_precision + precision_growth_rate + PropYrConverged,
                  method = 'class',
                  data = fsdf)
```

```
#plot(fit_fsdf, uniform=TRUE,
#     main = "Which models are good or bad (precision and convergence)?",
#     margin = .1)
#text(fit_fsdf, use.n = TRUE,
#     all = TRUE,
#     cex = .7)
rpart.plot(fit_fsdf, extra = 108, type = 4, clip.right.labs = FALSE)
```

```
## Warning: Removed 4005 rows containing missing values (geom_point).
```

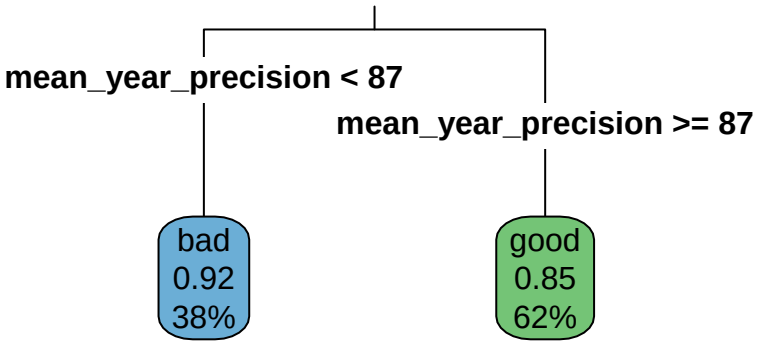


Figure 5: Decision tree for deciding which models are good or bad based on precision only

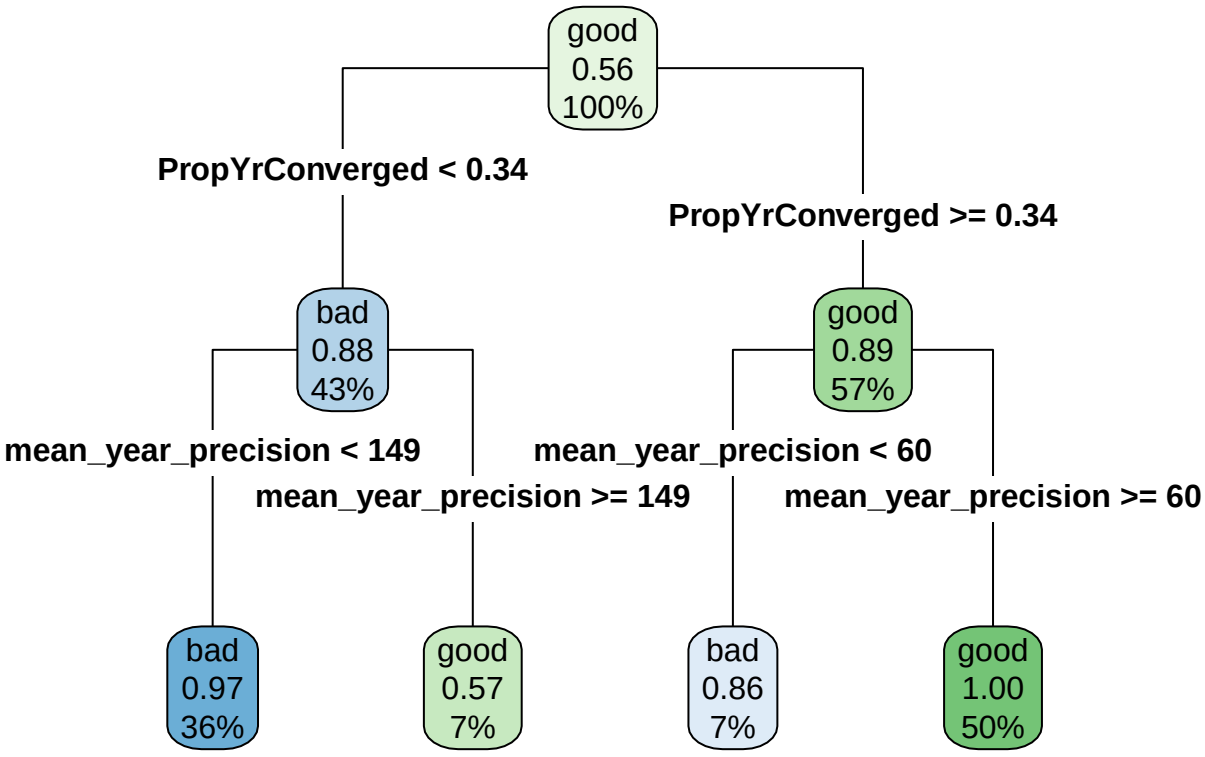


Figure 6: Decision tree for deciding which models are good or bad based on precision and convergence

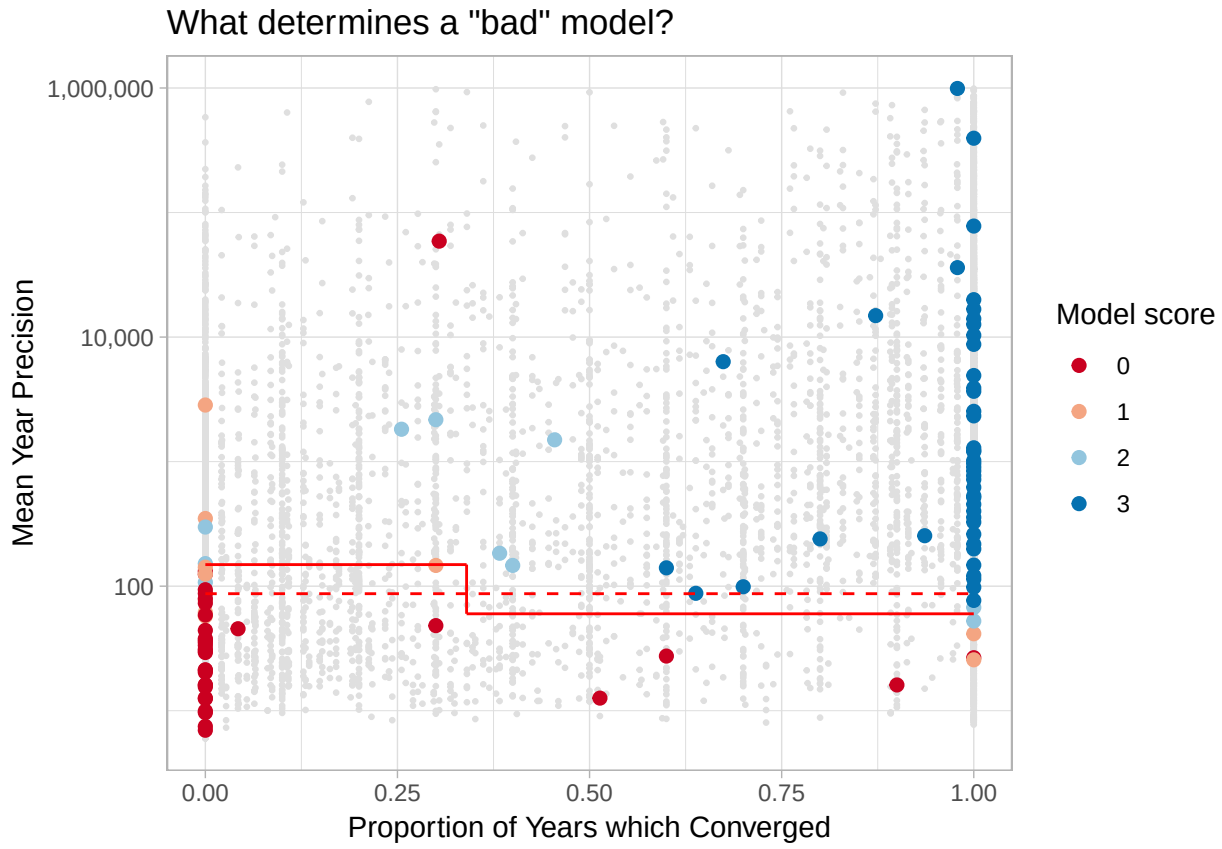


Figure 7: Graph showing results of consulation with model experts, including the results of the decision tree. Light grey points in the background show all models which were not part of the consultation

The success of the decision tree for classifying bad models is:

- Using precision only: 92% for bad models, 85% for good models
- Using precision and convergences: 95% for bad models, 95% for good models

Using both metrics is clearly better for splitting good and bad models, therefore this split will be used. The split is as follows:

- If the proportion of years converged is < 0.3437, the model will be classified as 'bad' if the mean year precision is < 149.3
- If the proportion of years converged is $>= 0.3437$, the model will be classified as 'bad' if the mean year precision is < 60.2

Classifying the data

Examining the classification variables

Before running the classifiers, it is important to determine whether or not any of the variables are highly taxa dependent. If they are, they are much less generalisable across data sets.

This was done for all variables and the results from two of the variables are shown below:

- First is the violin plot for prop_repeats_grp: the proportion of site:year combinations for the species of interest which have more than one visit.
- Second is the violin plot for prop_abs_list: the proportion of data for the taxonomic group *not including the species of interest* which have a list length >1.

The `prop_repeats_grp` data shows a broad range of results from 0 to 1 for all taxonomic groups. By contrast, the `prop_abs_list` data shows very clear division between taxonomic groups. Therefore, any partitioning on the basis of the `prop_abs_list` variable is taxonomically biased. For this reason, `prop_abs_list` will not be used for producing decision trees.

```
# Subset data to remove last 10 yr data, as it just makes the plot messier
td_taxa <- trendsData[as.character(trendsData$Taxa)==
                      as.character(trendsData$Taxa_Root),]
```

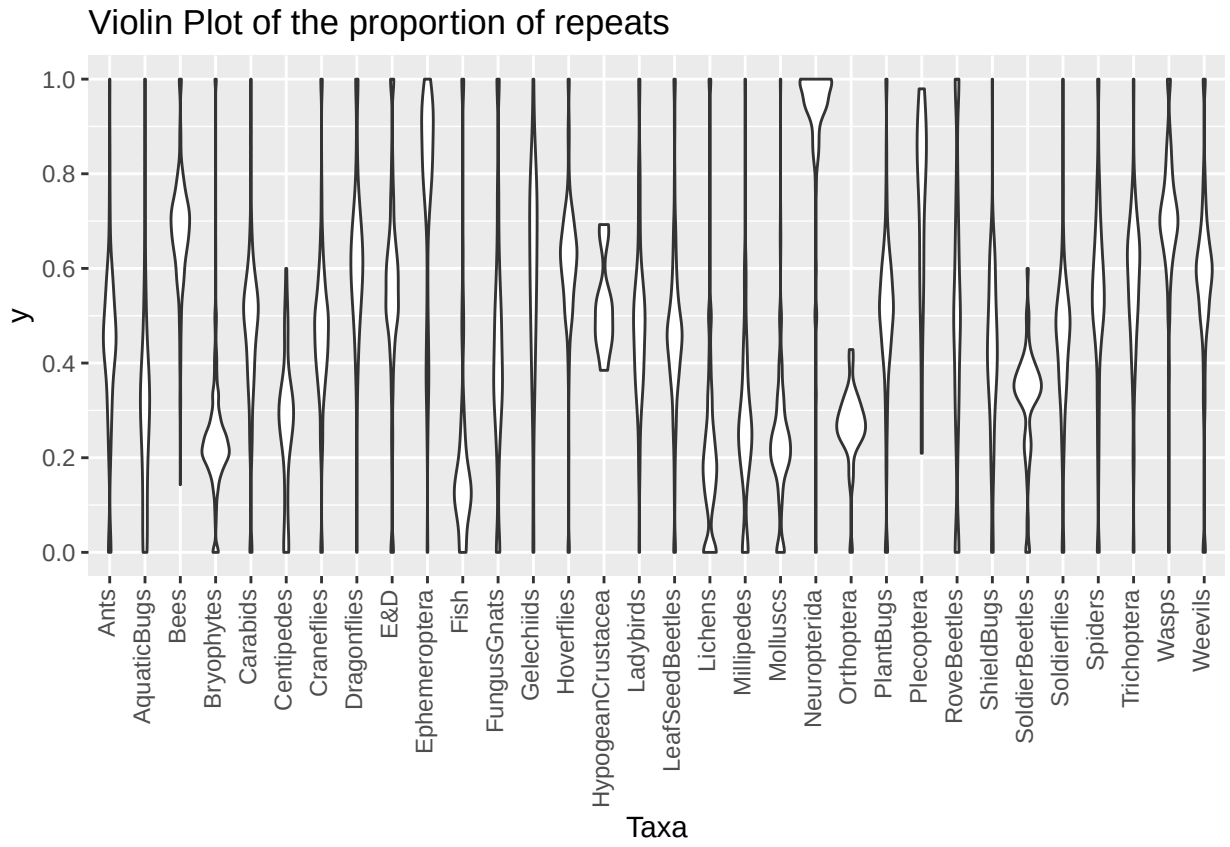


Figure 8: Violin plot for the prop_repeats_grp variable, defined as the proportion of all site:year combinations, where the focal species is observed, that have more than 1 visit for that taxonomic group

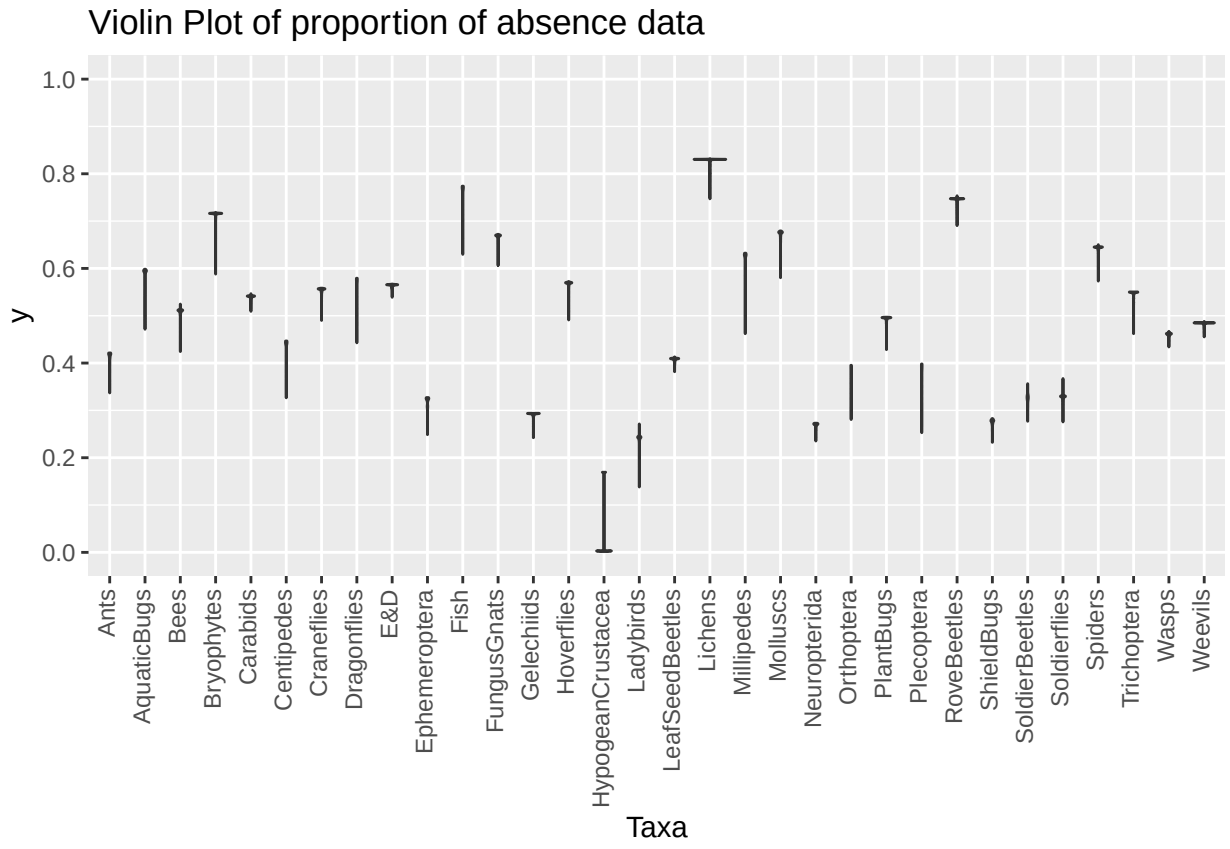


Figure 9: Violin plot for the prop_abs_list variable, defined as the proportion of records with list length greater than 1, for all records where the focal species was not observed

Correlation between variables

To examine the between all the variables, a pairs plot is produced below. As can be seen, P90 is strongly correlated with number of records (Spnvisits), as expected. Prop_abs is negatively correlated with number of records, as the fewer the records for a species, the higher the proportion of taxonomic data without the species of interest.

Apart from number of records, which is not included in the classification tree, the variables show little cross correlation, suggesting they are a good choice for building a decision tree.

```
td_sub <- trendsData[,c('Spnvisits','P90','visits_P90',
                        'prop_repeats_grp','prop_abs')]
pairs.panels(td_sub, hist.col = 'blue',smooth = TRUE)
```

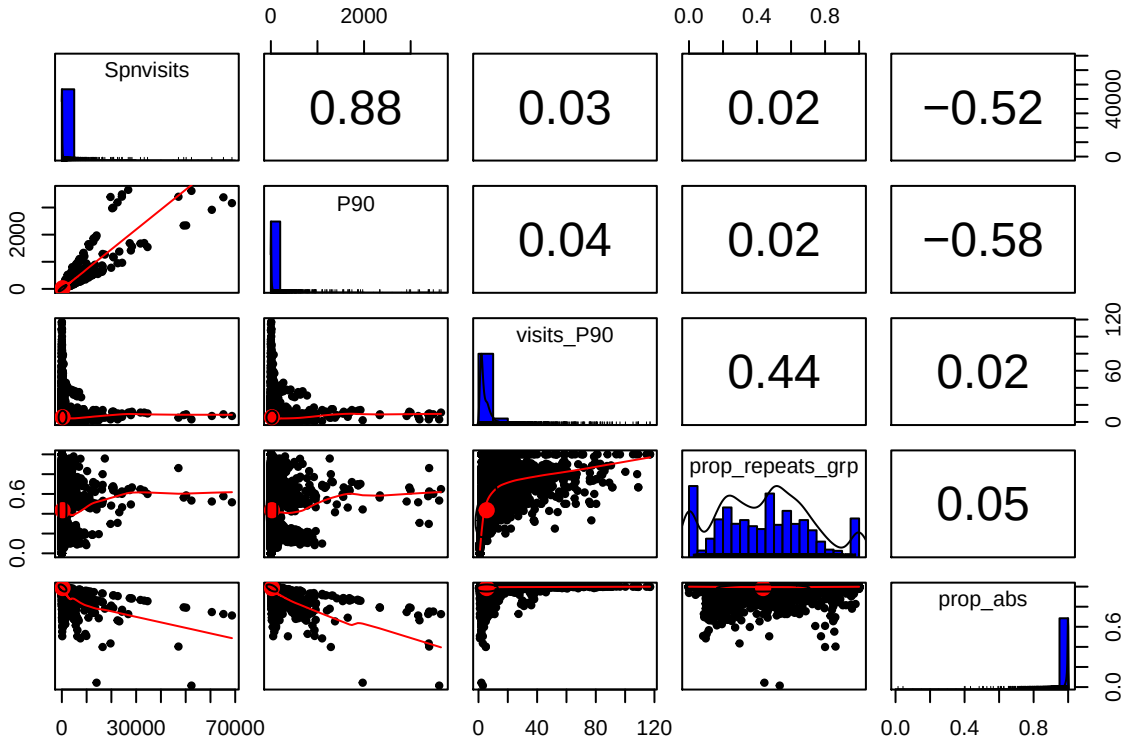


Figure 10: Pairs plot for all variables used to construct the classification tree, plus the Spnvisits variable, defined as the total number of records for each species of interest

```
td_sub2 <- log10(trendsData[,c('Spnvisits','P90','visits_P90')]+1)
td_sub2 <- cbind (td_sub2, trendsData[,c('prop_repeats_grp','prop_abs')] )

pairs.panels(td_sub2, hist.col = 'blue',smooth = TRUE)
```

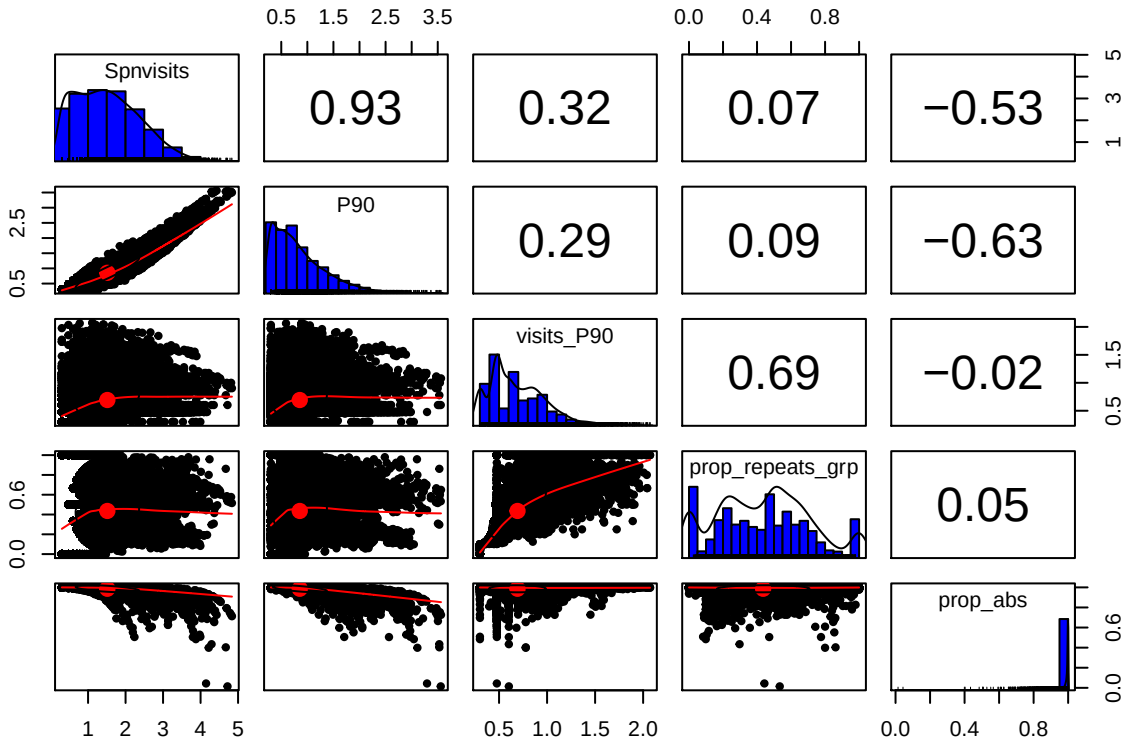


Figure 11: Pairs plot for all variables used to construct the classification tree, plus the Spnvisits variable, defined as the total number of records for each species of interest. Spnvisits, P90 and visits_P90 have been log transformed for this version.

Input data requirement

In order to be included in the dataset, one preliminary filtering step was carried out to remove some of the data. This step was to remove all records from visits to sites that never saw a repeat visit in any other years, for that taxonomic group. This step removed 33% of all records across our database.

- For instance, if a dragonfly was recorded in one site in 1990, if there was one more record of any dragonfly at that site in any year except 1990, both records would be included.
- If there was another record for another dragonfly to the same site in 1990, but no other records for that site, both of these records would be removed from the database.

This step was carried out as it was carried out on the data prior to running the models. Failing to do this step would make the results invalid.

This means that any decision tree below has the pre-requisite that all records in the database must meet this requirement.

To show the effect of this step, below is a plot showing the proportion of the records removed for each taxonomic group.

```
# Read in the metrics
RM <- read.csv('Results/metrics/ALL_rawMetrics.csv')
RM <- RM[RM$Taxa==as.character(RM$Taxa_Root),]
# Read in the metrics calculated for all excluded records
RM_1rec <- read.csv('Results/metrics/ALL_1rec.csv')
colnames(RM_1rec)[2] <- 'numrec_removed'
RM <- merge(RM,RM_1rec)
RM$Taxa <- as.character(RM$Taxa)
df <- NULL

# Calculate proportion of records removed for each taxonomic group
for(taxa in sort(unique(RM$Taxa))){
  num_inc <- sum(RM$Spnvisits[RM$Taxa==taxa])
  num_exc <- sum(RM$numrec_removed[RM$Taxa==taxa])
  df <- rbind(df,
              data.frame(taxa = taxa,
                         Removed = (num_exc/(num_inc+num_exc)),
                         Included = (num_inc/(num_inc+num_exc))))
}

taxa_melt <- melt(df, id=c('taxa'))
```

```
df2 <- NULL

for(taxa in sort(unique(RM$Taxa))){
  num_inc <- sum(RM$Spnvisits[RM$Taxa==taxa])
  num_exc <- sum(RM$numrec_removed[RM$Taxa==taxa])
  df2 <- rbind(df2,
              data.frame(taxa = taxa,
                         RecordsIncluded = num_inc,
                         RecordsExcluded = num_exc,
                         TotalRecords = num_inc + num_exc,
                         ProportionIncluded = num_inc/(num_inc+num_exc)))
}
```

```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

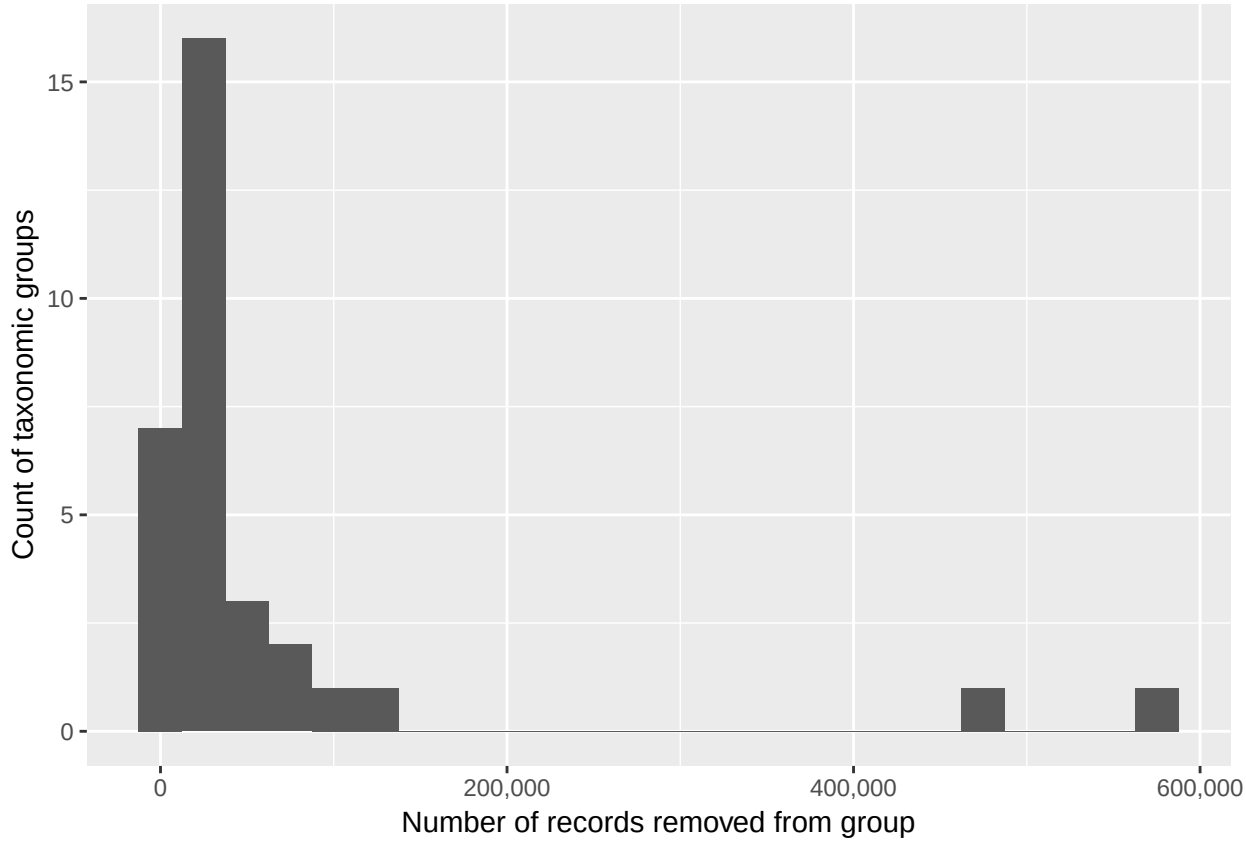


Figure 12: Histogram showing number of records removed from each taxonomic group due to the input data requirement of repeat visits. As can be seen, most taxonomic groups have less than 100,000 records removed due to this step, while a couple of groups have in excess of 400,000 removed. These groups are shown below

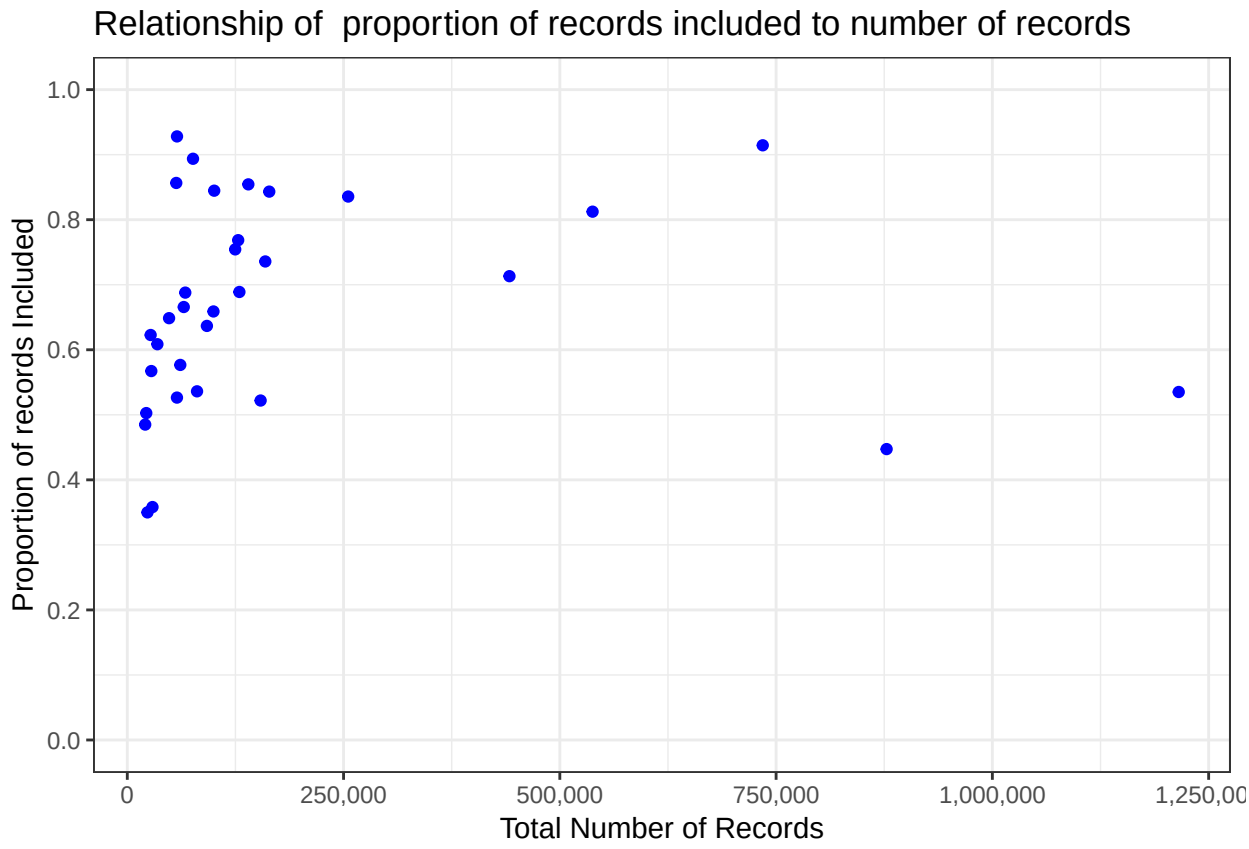


Figure 13: Plot showing proportion of records included vs total number of records. As can be seen, most taxonomic group retain at least half of their records. Those which retain fewer than 50% of their recdords are shown below

Table 1: List of taxa with more than 400,000 records removed

taxa	RecordsIncluded	RecordsExcluded	TotalRecords	ProportionIncluded
Bryophytes	650324	565077	1215401	0.5350695
Lichens	392598	485143	877741	0.4472823

Table 2: List of taxa with more than half of records removed

taxa	RecordsIncluded	RecordsExcluded	TotalRecords	ProportionIncluded
Centipedes	8159	15156	23315	0.3499464
Lichens	392598	485143	877741	0.4472823
Millipedes	10429	18689	29118	0.3581633
Neuropterida	10045	10668	20713	0.4849611

```
stack_records(taxa_melt, colours = c('red', '#9999FF'),
              ylabel = 'Proportion of records',
              title = 'Proportion of records which are included')
```

Overall, we found that 0.33 of all records were removed, varying from 0.07 to 0.65 across projects

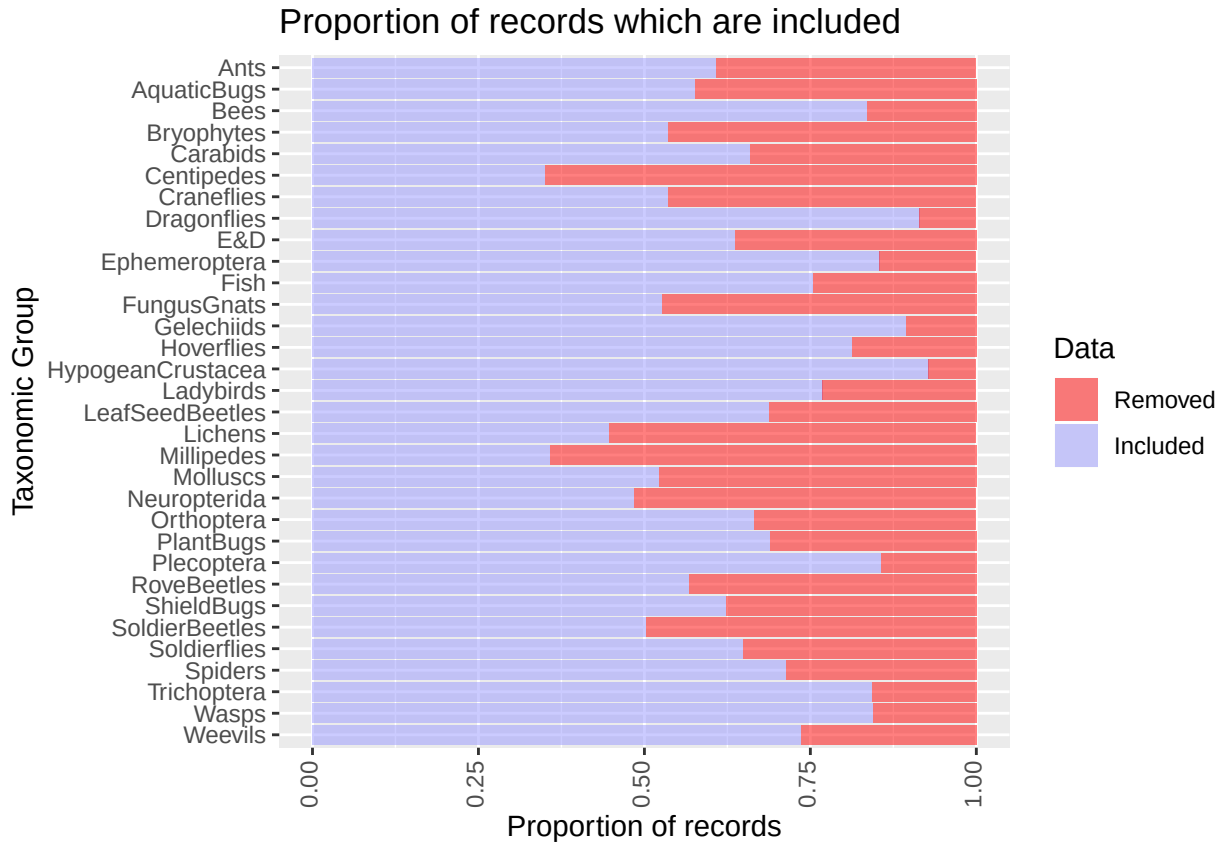


Figure 14: Graph showing the proportion of records which are included in the models. Those which are not are excluded or to sites which only received visit(s) in one year within the taxonomic group

Splitting the output data into good and bad models

Using the decision tree above for splitting output models into good and bad, the model outputs can be classified as good or bad. For some of the species we do not have the proportion of years converged. For these we use the simple decision tree: mean_year_precision >= 87.

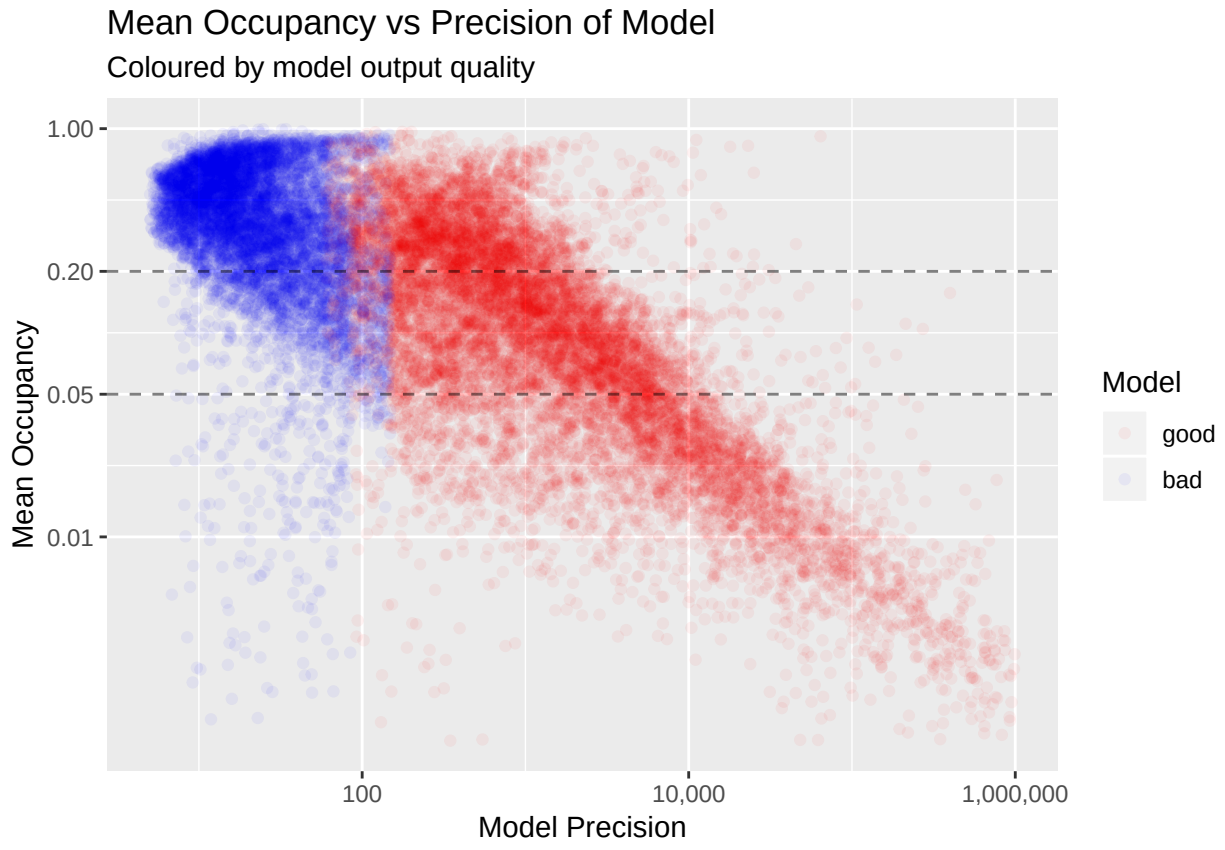
```
trendsData$model_good <- rep('bad',nrow(trendsData))
trendsData$model_good[(trendsData$PropYrConverged < 0.3437 &
                   trendsData$mean_year_precision >= 149.3) |
                   (trendsData$PropYrConverged >= 0.3437 &
                    trendsData$mean_year_precision >= 60.02) |
                   (is.na(trendsData$PropYrConverged) &
                    trendsData$mean_year_precision >= 87)] <- 'good'

trendsData$model_good <- factor(trendsData$model_good, levels = c('good','bad'))
```

Effect of rarity of species on precision

For species which have a very low occupancy, the absolute precision on the occupancy estimate can be very high even if there is very little data. Conversely, a common or high occupancy species is is likely to have a low precision estimate even with a decent number of records. This is due to the fact that uncertainty on a high occupancy number equates to a greater absolute range in occupancy estimate when compared to the same relative uncertainty on a lower occupancy number.

This effect is shown in the below graph showing mean occupancy vs precision.



After discussion, it was decided to drop all species with an occupancy of <0.05 from the data before running the rules of thumb classification tree. By doing so, roughly 20% of all species are dropped. These species are all rare species, and generally easy to model with relatively few records owing to their rarity i.e. even if very little data the model can be confident that the species is rare. By dropping these rare species, the resulting rules of thumb will be more applicable to common species.

```
trendsData <- trendsData %>% filter(mean_means > 0.05)
```

Classification trees

Equally weighted decision tree

Using the split in the models between 'good' and 'bad', you can create a classification tree that attempts to use the variables we have extracted to partition the data to the best of its ability using a decision tree.

For this decision tree the evtree function is being used. This function creates a globally optimised decision tree, rather than a greedy decision tree. This results in more accuracy though the processing time is longer.

```
# Perform the fit
goodweight <- 1
badweight  <- 1
set.seed(1)
ev <- evtree(model_good ~ median + P90 + visits_median + visits_P90 +
               prop_repeats_grp + prop_abs + prop_list_one,
             data = trendsData,
             weights = ifelse(trendsData$model_good == 'bad',
                              goodweight, badweight),
             control = evtree.control(maxdepth = 2))
```

```
load(file = 'ev_trees.Rdata')
ev <- ev_trees$ev
ev
```

```
##
## Model formula:
## bad ~ median + P90 + visits_median + visits_P90 + prop_repeats_grp +
##     prop_abs + prop_list_one
##
## Fitted party:
## [1] root
## |   [2] prop_abs < 0.98737
## |   |   [3] P90 < 7.4: bad (n = 513, err = 38.6%)
## |   |   [4] P90 >= 7.4: good (n = 3925, err = 5.0%)
## |   [5] prop_abs >= 0.98737
## |   |   [6] P90 < 4.3: bad (n = 7765, err = 16.6%)
## |   |   [7] P90 >= 4.3: good (n = 3090, err = 17.5%)
##
## Number of inner nodes:    3
## Number of terminal nodes: 4
```

```
plot(ev)
```

The first node in this decision tree is `prop_abs` >= 0.987. To meet this requirement, at least 98.7% of the data for the taxonomic group needs to come from species other than the species of interest. This is equivalent to 76 additional records for other species for every 1 record of the species of interest.

For example, if there were 2000 records within a group, and 40 records of the species of interest, prop_abs

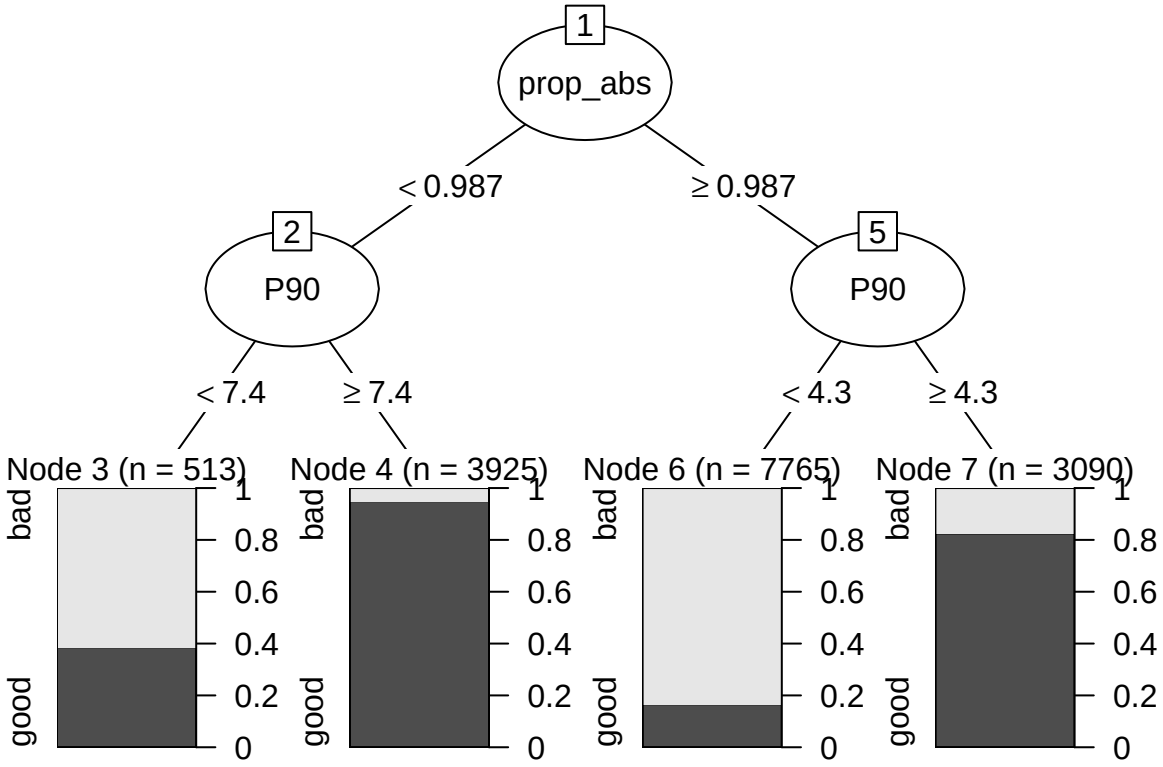


Figure 15: Decision tree for rules of thumb

would be (2000-40)/2000 = .98. This species would therefore not meet this criterion. However, a species with 40 records within a group of 4000 records would meet this criterion (prop_abs = .99)

The second node is `P90`. This is the 90th percentile of number of records for years which have records.

To be classisified as worth running a model:

After removing all data from sites which never saw a repeat visit in subsequent year for any species within the taxonomic group (see *Input Data Requirement*):

If:

- `prop_abs` >= 0.987 and `P90` >= 4.3 OR
- `prop_abs` < 0.987 and `P90` >= 7.4

The data will probably produce an acceptable model.

```
TP <- sum(predict(ev)==trendsData$model_good & predict(ev)=='good')
FP <- sum(predict(ev)!=trendsData$model_good & predict(ev)=='good')
TN <- sum(predict(ev)==trendsData$model_good & predict(ev)=='bad')
FN <- sum(predict(ev)!=trendsData$model_good & predict(ev)=='bad')
```

This decision tree has a positive precision (TP/(TP+FP)) of 89.5%, meaning 89.5% of the time, if the decision tree predicts the data will be able to produce a good model, it will.

This decision tree has a negative precision (TN/(TN+FN)) of 82%, meaning 82% of the time, if the decision tree predicts the data will not be able to produce a good model, it will indeed not produce a good model.

The recall/sensitivity/true positive rate (TP/(TP+FN)) is 80.8%, meaning 80.8% of all datasets which would produce a good model are classified as good datasets.

The specificity/true negative rate (TN/(TN+FP)) is 90.2%, meaning 90.2% of all datasets which would NOT produce a good model are classified as bad.

A graphic way of considering the decision tree is in the following graphs. These graphs have the decision tree cutoffs shown in a black line, where any data below the line are predicted as not producing a good model and any above the line are predicted as having enough data. Points coloured blue did in fact produce a good model, while those below the line did not:

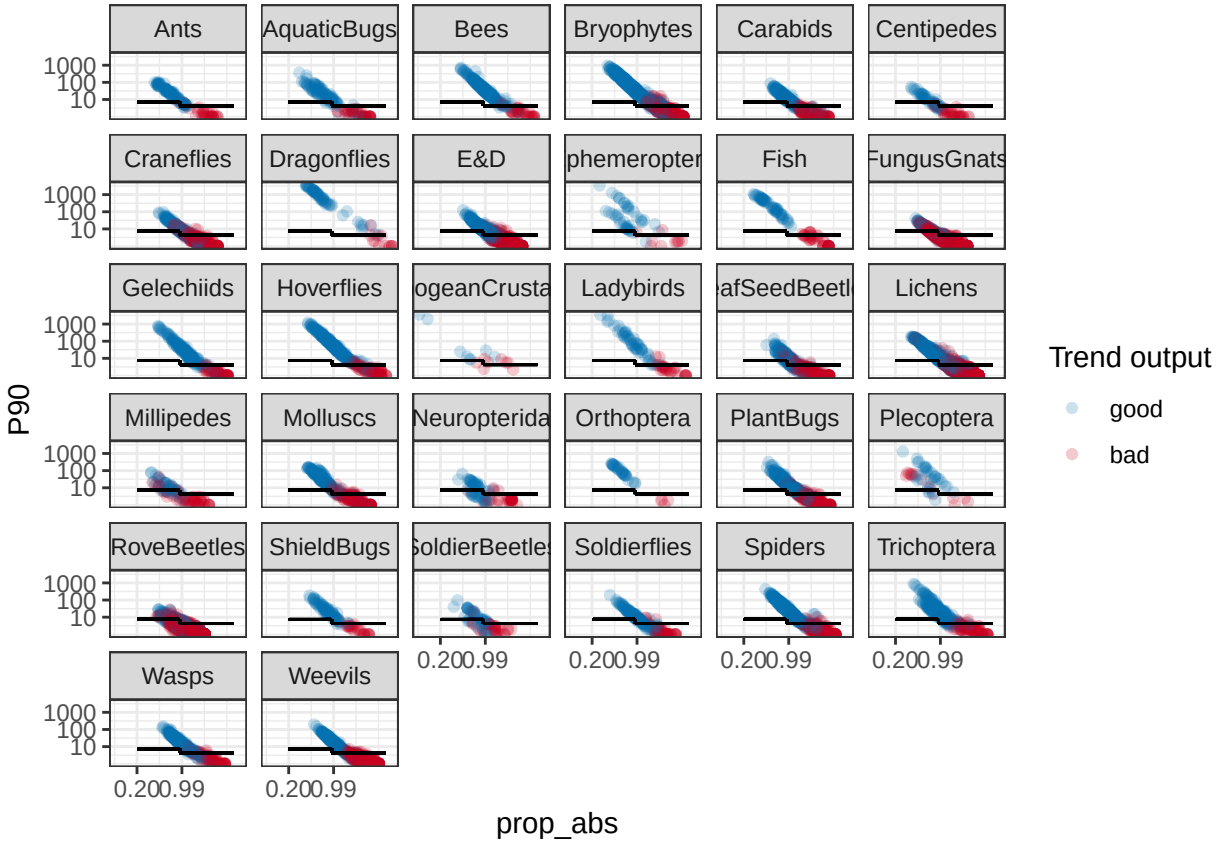


Figure 16: The results of the decision tree with P90 on a log-scale, and prop_abs on a logit scale, by taxonomic group

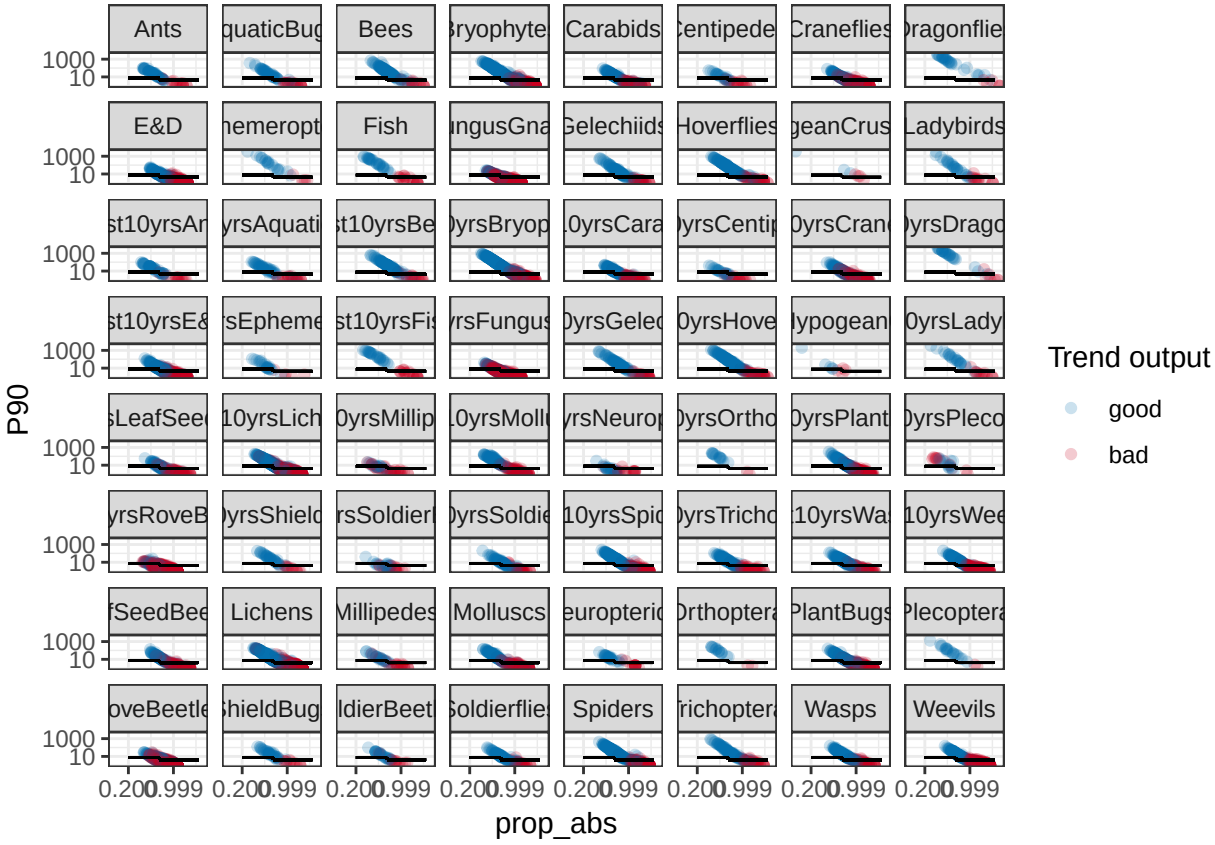


Figure 17: The results of the decision tree with P90 on a log-scale, and prop_abs on a logit scale, by taxonomic group and splitting 10 year and all year trends

10:1 Good:bad tree - Aspirational Target

The decision tree is computed again twice more. In these models, the penalty for misclassifying good models as bad, or vice versa, is varied.

The first one below is with a 10:1 good:bad ratio i.e. it is 10x more important to correctly classify good models as good. This weighting makes the decision tree much more specific, at the expense of sensitivity.

By using a 10:1 good:bad ration, models which are classified as good are very likely to be good. As such, this decision tree can be considered an aspirational target.

Note that in the decision tree the values for n are not correct. This is due to the method evtree uses for weighting decisions: by replicating data of one output, in this case it replicates the 'bad' output 10x, such that assigning each of these outputs as bad becomes 10x more signigicant.

```
# produce a decision tree for goodweight = 10
goodweight <- 10
badweight  <- 1
set.seed(1)
ev_aspire <- evtree(model_good ~ median + P90 + visits_median + visits_P90 +
                      prop_repeats_grp + prop_abs + prop_list_one,
                    data = trendsData,
                    weights = ifelse(trendsData$model_good == 'bad',
                                     goodweight, badweight),
                    control = evtree.control(maxdepth = 2))
save(x = ev_aspire, file = 'ev_aspire.Rdata')
```

```
ev_aspire <- ev_trees$ev_aspire
ev_aspire
```

```
##
## Model formula:
## bad ~ median + P90 + visits_median + visits_P90 + prop_repeats_grp +
##     prop_abs + prop_list_one
##
## Fitted party:
## [1] root
## |   [2] prop_abs < 0.95748
## |   |   [3] P90 < 22.4: bad (n = 1426, err = 12.3%)
## |   |   [4] P90 >= 22.4: good (n = 1428, err = 11.9%)
## |   [5] prop_abs >= 0.95748
## |   |   [6] P90 < 9.6: bad (n = 75619, err = 3.7%)
## |   |   [7] P90 >= 9.6: good (n = 4590, err = 23.1%)
##
## Number of inner nodes:    3
## Number of terminal nodes: 4
```

```
plot(ev_aspire)
```

```
TP_aspire <- sum(predict(ev_aspire)==trendsData$model_good & predict(ev_aspire)=='good')
FP_aspire <- sum(predict(ev_aspire)!=trendsData$model_good & predict(ev_aspire)=='good')
TN_aspire <- sum(predict(ev_aspire)==trendsData$model_good & predict(ev_aspire)=='bad')
FN_aspire <- sum(predict(ev_aspire)!=trendsData$model_good & predict(ev_aspire)=='bad')
```

The first node in this decision tree is `prop_abs` $>= 0.957$. To meet this requirement, at least 95.7% of the data for the taxonomic group needs to come from species other than the species of interest. This is equivalent to 23 additional records for other species for every 1 record of the species of interest.

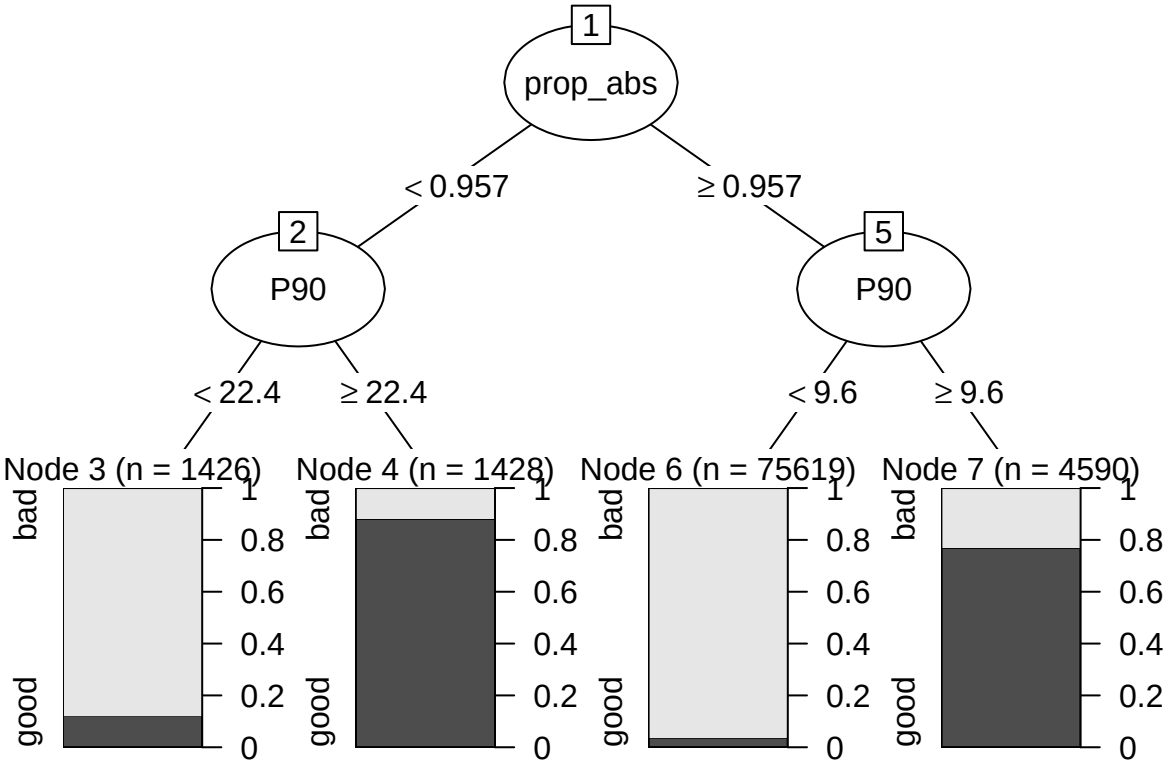


Figure 18: Decision tree for aspirational targets

To be classisified as worth running a model:

After removing all data from sites which never saw a repeat visit in subsequent year for any species within the taxonomic group (see *Input Data Requirement*):

If:

- `prop_abs` >= 0.957 and `P90` >= 9.6 OR
- `prop_abs` < 0.957 and `P90` >= 22.4

The data will probably produce an acceptable model.

The recall/sensitivity/true positive rate (TP/(TP+FN)) is 61.7%, meaning 61.7% of all datasets which would produce a good model are classified as good datasets (compare with 89.5% for the first decision tree).

The specificity/true negative rate (TN/(TN+FP)) is 98.4%, meaning 98.4% of all datasets which would NOT produce a good model are classified as bad (compare with 90.2% for the first decision tree).

This decision tree is shown on the below plot:

1:10 Good:bad tree - bare minimum to try running a model

The decision tree below is with a 1:10 good:bad ratio i.e. it is 10x more important to correctly classify bad models as bad This means that any models which are classified as bad are very likely to be bad, while a model classified as good may or may not be actually good. As such, this decision tree can be considered a bare minimum threshold.

In testing, it was found that trees using all variables, P90 and prop_abs and just P90 produced similar sensitivity and specificity scores. For simplicity, just P90 was used for this tree.

```
# produce a decision tree for badweight = 10
goodweight <- 1
badweight  <- 10
set.seed(1)
ev_bm <- evtree(model_good ~ P90,
                data = trendsData,
                weights = ifelse(trendsData$model_good == 'bad',
                                 goodweight, badweight),
                control = evtree.control(maxdepth = 2))
ev_trees <- list(ev,ev_aspire,ev_bm)
names(ev_trees) <- c('ev','ev_aspire','ev_bm')
save(x = ev_trees, file = 'ev_trees.Rdata')
```

```
ev_bm <- ev_trees$ev_bm
ev_bm
```

```
##
## Model formula:
## bad ~ P90
##
## Fitted party:
## [1] root
## |   [2] P90 < 1.4: bad (n = 5699, err = 43.5%)
## |   [3] P90 >= 1.4: good (n = 79461, err = 5.4%)
##
## Number of inner nodes:    1
## Number of terminal nodes: 2
```

```
plot(ev_bm)
```

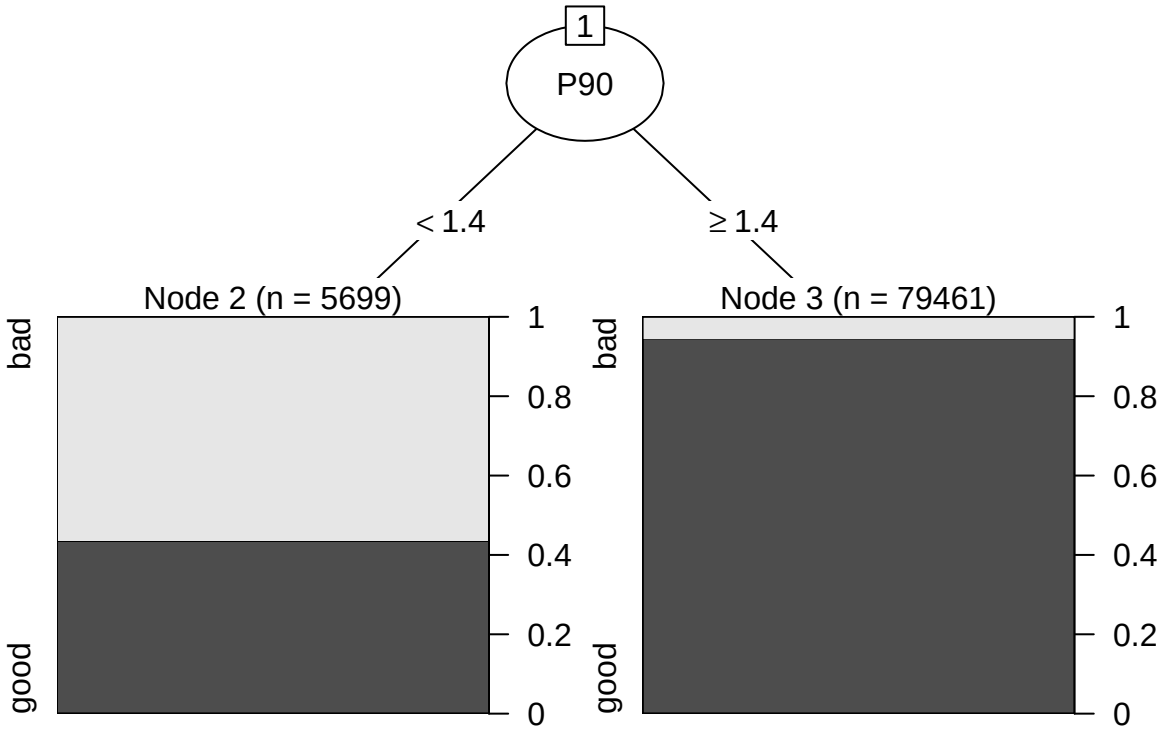


Figure 19: Decision tree for aspirational targets

```
TP_bm <- sum(predict(ev_bm)==trendsData$model_good & predict(ev_bm)=='good')
FP_bm <- sum(predict(ev_bm)!=trendsData$model_good & predict(ev_bm)=='good')
TN_bm <- sum(predict(ev_bm)==trendsData$model_good & predict(ev_bm)=='bad')
FN_bm <- sum(predict(ev_bm)!=trendsData$model_good & predict(ev_bm)=='bad')
```

The sensitivity is 96.8%, while the specificity is 42.7%.

As can be seen in the sensitivity and specificity scores above, this model creates a large number of false positives but a very low number of false negatives. Therefore, if a dataset does not meet this very low threshold of P90 >= 1.4, it will almost certainly not result in a good model. The only exceptions to this rule are likely to be the rarest of species with a rich taxonomic dataset.

Table 3: Sensitivity and Specificity for the decision trees

Decision_Tree	Sensitivity	Specificity
Base	80.8	90.2
Aspirational	61.7	98.4
Bare Minimum	96.8	42.7

A table of the sensitivity and specificity scores for these decision trees is shown in Table 3.

All these decision trees can be plotted on the same graph. This is shown below. All data points below the lines are predicted to not produce a good model. Those models coloured blue did produce a good model, those below did not.

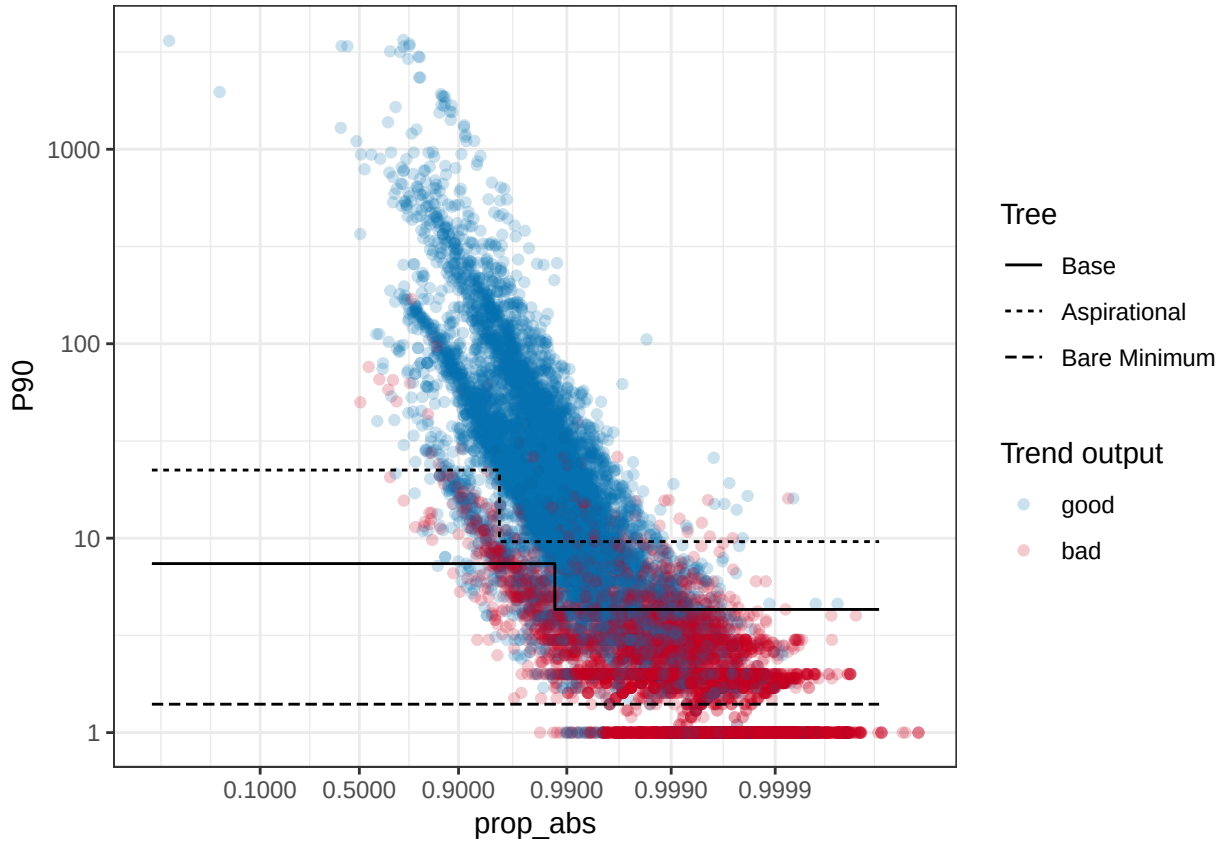


Figure 20: The results of the decision trees with P90 on a log-scale, and prop_abs on a logit scale

As can be seen, the aspirational tree line is the hardest to get above (it is highly specific), but accordingly there are a large number of blue (successful) models below the line.

Similarly, the bare minimum tree line is very easy to get above (it is highly sensitive), but there are a lot of red (unsuccessful) models above the line.

The Base model is a good balance between these two.

Applying the decision tree to all the data

In order to apply the decision trees to the data, including species and datasets for which we do not currently

have a model, the function below creates a new parameter `data_good` which returns `bad` if the data are likely to produce a bad model, and `good` if the data are likely to produce a good model

```
calc_bad <- function(df){
  df$Node1 <- df$prop_abs>=.987
  df$Node2 <- df$P90>=4.3
  df$Node3 <- df$P90>=7.4
  df$data_good <- df$data_aspire <- rep('bad',length(df$species))
  df$data_good[(df$Node1&df$Node2)|(df$Node3)] <- 'good'
  df$data_good <- factor(df$data_good,levels=c('good','bad'))

  df$asp_Node1 <- df$prop_abs>=.957
  df$asp_Node2 <- df$P90>=9.6
  df$asp_Node3 <- df$P90>=22.4
  df$data_aspire[(df$asp_Node1&df$asp_Node2)|(df$asp_Node3)] <- 'good'
  df$data_aspire <- factor(df$data_aspire,levels=c('good','bad'))

  if(!is.null(df$habitat)){
    df$habitat <- as.character(df$habitat)
  }
  if(!is.null(df$region)){
    df$region <- as.character(df$region)
  }
  if(!is.null(df$code)){
    df$code <- as.character(df$code)
  }
  return(df)
}

trendsData <- calc_bad(trendsData)
```

More investigation about the parameters in the decision tree

The relationship between P90 and P50

One of the possible concerns in applying this more widely is that while we found that P90 was an important parameter in the decision tree, it could also be strongly correlated with P50 (i.e. the median number of visits per year). We note that we have good prior reasons for thinking that P90 is useful (because it allows good estimation of detection), but here we test the relationship between the two.

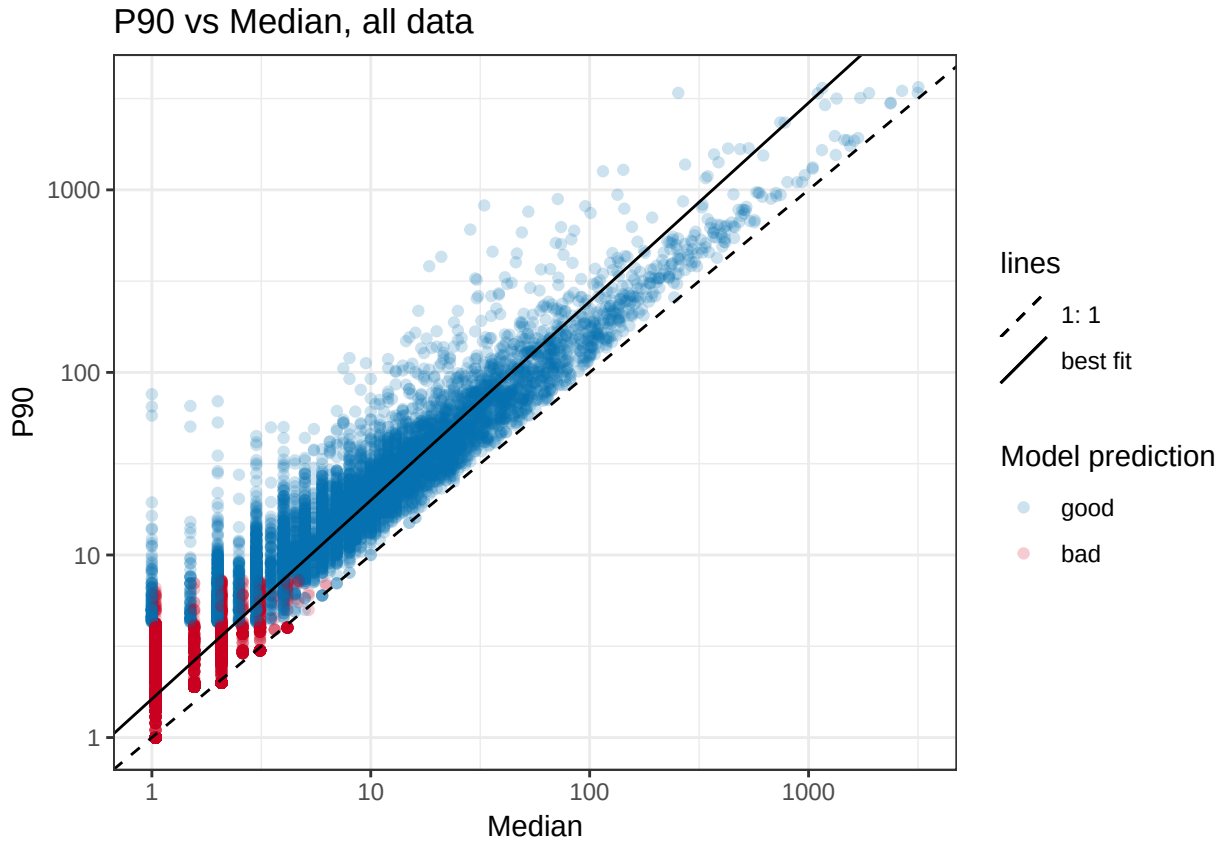


Figure 21: This graph shows that there is a linear trend of the P50 against the P90, so although the P90 represents the best years in the dataset (and this is indeed a better predictor of the adequacy of the data than P50) this is broadly related to the median years. Over the range of the bulk of these data (i.e. up to median = 100) the P90 is two to three times greater than the median

The relationship between precision and P90

We have used a decision tree to model when the data are minimal adequate. However, does more data automatically mean better trends?

One way to answer this is to model the lower limit of P90 against mean year precision. This can be done using quantile regression

```
trendsData.p90.7_4 <- subset(trendsData, P90>7.4)

mod.limit.of.precision <- rq(
```

```
      log10(mean_year_precision) ~
        log10(P90),
      tau = 0.05, data = trendsData.p90.7_4)

# predict the points at P90 = 5.85 and max P90
segment.ys <-
  10^(predict(mod.limit.of.precision,
              newdata = data.frame('P90' = c(7.4,
                                             max(trendsData.p90.7_4$P90)))))
```

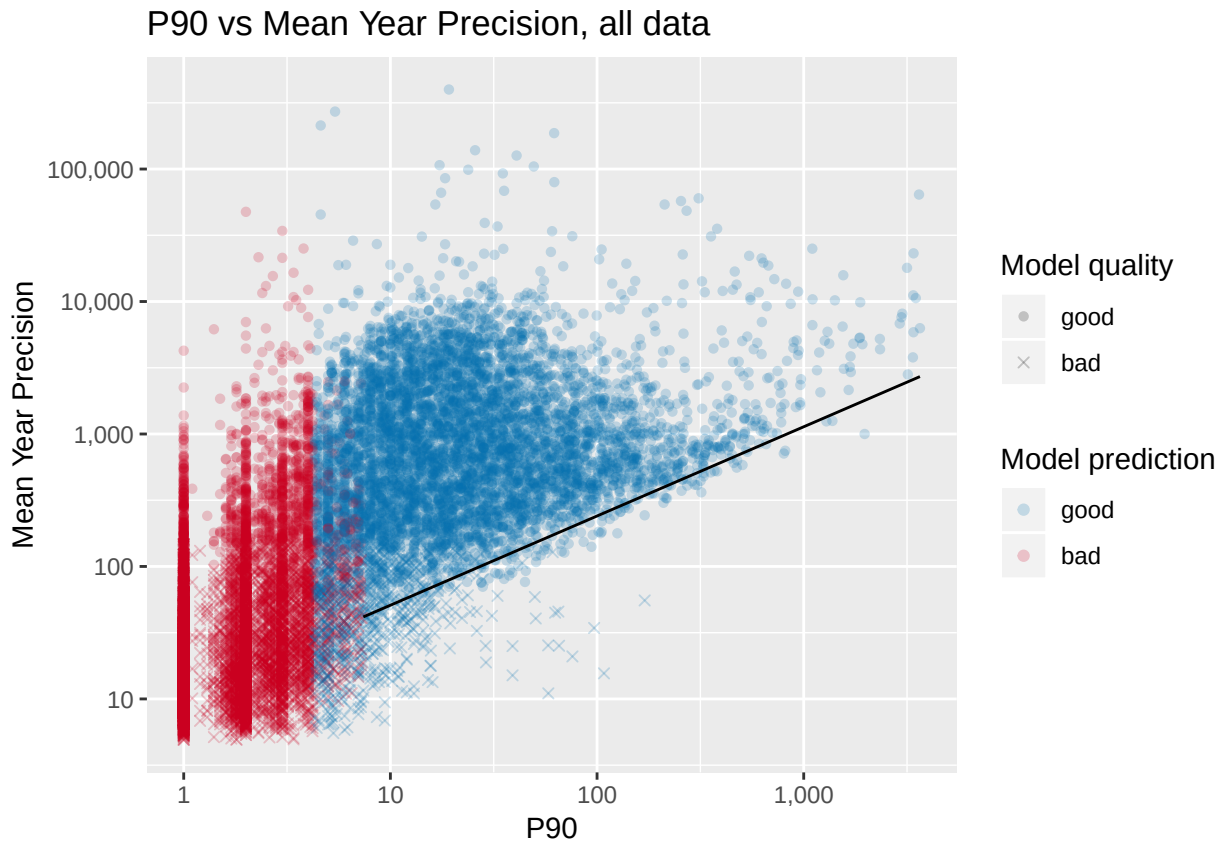


Figure 22: This shows that after a cut-off of P90 > 7.4 (the higher of the P90 cut-offs from the best guess decision tree), the more data there are (represented here by the increase in P90) the better the precision on the resulting occupancy model. The line represents the quantile regression through the 5th percentile. This shows that precision can be high with low P90, but the lowest possible precision is higher with higher P90, i.e. more data

How many species can we model?

For the UK

Based on the work above, we can estimate how many species we can model for taxa across the whole of the UK and across multiple habitat and regional areas.

Extracting butterfly and moth data

Before estimating how many species we can model, the data for the moth and butterfly records can be extracted and added to the metric output data table. Just the metrics which came out from the decision trees will be included in this extraction, so it runs much faster than the earlier data extraction.

```
# Location of the jasmin butterfly and moth data
jasmin <- file.path('W:/PYWELL_SHARED/Pywell Projects/BRC/Mark Logie/TSDA/Jasmin_data')
data_files <- list.files(jasmin,full.names = TRUE)

# Function to extract just the P90 and prop_abs metrics
simpleDataMetrics <- function(sp,tFDall,num_rec,Taxa,habitat=NA,region=NA){
  tFD <- tFDall[tFDall$CONCEPT==sp,]
  year_count <- table(tFD$year)
  P90 <- as.numeric(quantile(year_count,probs = c(.9)))
  if(is.na(P90)){P90 <- 0}
  prop_abs <- 1-(nrow(tFD)/num_rec)
  Spnvisits <- nrow(tFD)
  data.frame(species = sp,
             habitat = habitat,
             region = region,
             P90 = P90,
             prop_abs = prop_abs,
             Spnvisits = Spnvisits,
             Taxa = Taxa,
             Taxa_Root = Taxa)
}

for(data_file in data_files){
  cat('Starting', basename(data_file))
  load(data_file)

  spp <- unique(taxa_data$CONCEPT)
  # one group has a species called '', best to remove this (single) record
  spp <- spp[spp != '']

  cat('\n', length(spp), 'species\n')
  num_rec <- nrow(taxa_data)
  taxa_data$year <- lubridate::year(taxa_data$TO_STARTDATE)

  # Remove sites which have not seen a repeat in subsequent years.
  # This is a requirement here as this was a step taken by the model.
  yps <- rowSums(acast(taxa_data, SQ_1KM ~ year, length, value.var = 'year') > 0)
  sites_to_include <- names(yps[yps >= 2])
  taxa_data <- taxa_data[taxa_data$SQ_1KM %in% sites_to_include,]

  Taxa <- substr(basename(data_file),1,regexpr('_',basename(data_file))[1]-1)
```

```
  # Calculate metrics for all records
  raw_metrics <- lapply(spp,
                        FUN = simpleDataMetrics,
                        tFDall = taxa_data,
                        num_rec = num_rec,
                        Taxa = Taxa)
  raw_metrics <- do.call(rbind, raw_metrics)
  write.csv(raw_metrics, file = file.path('Results/metrics',
                                          paste0('jasminMetrics_',
                                                 gsub('.rdata$', '',
                                                      basename(data_file)),
                                                 '.csv')),
            row.names = FALSE)
}

# Now add these data to the combined metrics table
jasminDataFiles <- list.files(path = 'Results/metrics',
                              pattern = '^jasminMetrics',
                              full.names = TRUE)
master <- NULL
for(i in jasminDataFiles){

  x <- read.csv(i)
  master <- rbind(master, x)

}

# Remove all metrics not required
RM <- read.csv('Results/metrics/ALL_rawMetrics.csv')
RM <- RM[,colnames(RM) %in% colnames(master)]
master <- master[,colnames(master) %in% colnames(RM)]

master <- rbind(RM, master)

write.csv(master, file = 'Results/metrics/ALL_combinedRawMetrics.csv',
          row.names = FALSE)
```

Plotting the data

In order to plot the taxa data, functions are written to plot the graphs when given an input data frame. The functions for running this are below.

```
# Extract raw data metrics, as we want to look at all species, not just those that we had matching post
RM <- read.csv('Results/metrics/ALL_combinedRawMetrics.csv')
# Remove all last 10 yr data
RM$Taxa <- as.character(RM$Taxa)
RM <- RM[RM$Taxa==as.character(RM$Taxa_Root),]

# Function to extract data from data frame
#  df = a trendsData data frame
#  proportional = is the graph to show proportion of species (vs absolute numbers)
#  habitat = habitat of interest
#  region = region of interest, either full name or NUTS code
```

```
#  zeroes = are species with no records (Spnvisits==0) to be included?
#  aspirational = is the cutoff the aspirational target (10:1 good:bad)?  If this is
#    false, the default 1:1 decision tree is used
#  long_table = is the long table for plotting required?
num_spec <- function(df, habitat=NULL, region=NULL, proportional=FALSE,
                     long_table=TRUE, zeroes=TRUE, aspirational=FALSE){
  taxa_num <- NULL
  if(!zeroes){
    df <- df[df$Spnvisits != 0,]
  }
  for(taxa in sort(unique(df$Taxa))){
    num_spec <- length(unique(df$species[df$Taxa == taxa]))
    taxa_num <- rbind(taxa_num,
                      data.frame(taxa = taxa,
                                 num_spec = num_spec))
  }

  if(!is.null(habitat)){
    df <- df[as.character(df$habitat) == habitat,]
  }
  if(!is.null(region)){
    if(grepl('UK[A-Z]', region)){
      df <- df[as.character(df$code) == region,]
    } else {
      df <- df[as.character(df$region) == region,]
    }
  }
  taxa_count <- NULL
  for(taxa in sort(unique(taxa_num$taxa))){
    if(aspirational){
      data_good <- length(df$Taxa[df$Taxa == taxa &
                                  df$data_aspire == 'good'])
      data_bad  <- length(df$Taxa[df$Taxa == taxa &
                                  df$data_aspire == 'bad'])
    } else {
      data_good <- length(df$Taxa[df$Taxa == taxa &
                                  df$data_good == 'good'])
      data_bad  <- length(df$Taxa[df$Taxa == taxa &
                                  df$data_good == 'bad'])
    }
    no_data <- length(df$Taxa[df$Taxa == taxa &
                              df$Spnvisits == 0])
    tmpdf <- data.frame(taxa = taxa,
                        no_data = no_data,
                        bad = data_bad - no_data,
                        good = data_good)
    taxa_count <- rbind(taxa_count, tmpdf)
  }

  taxa_count <- merge(taxa_count, taxa_num)
  taxa_count$no_data <- taxa_count$num_spec - taxa_count$bad - taxa_count$good

  # Convert data table for turning into graphs
```

```
  taxa_melt <- melt(taxa_count, id=c('num_spec','taxa'))
  taxa_prop <- taxa_count
  taxa_prop$bad <- taxa_prop$bad/taxa_prop$num_spec
  taxa_prop$good <- taxa_prop$good/taxa_prop$num_spec
  taxa_prop$no_data <- taxa_prop$no_data/taxa_prop$num_spec
  taxa_prop$no_data[taxa_prop$num_spec==0] <- 1
  taxa_prop$bad[taxa_prop$num_spec==0] <- 0
  taxa_prop$good[taxa_prop$num_spec==0] <- 0
  taxa_prop_melt <- melt(taxa_prop, id=c('num_spec','taxa'))
  if(proportional){
    taxa_melt <- taxa_prop_melt
  }
  if(long_table){
    return(taxa_melt)
  } else {
    return(taxa_count)
  }
}

# Function to plot data extracted from data frame from num_spec function
#  df = a data frame from num_spec
#  prefix = a prefix to the title to describe the area or habitat of interest
stack_taxa <- function(df,prefix=NULL){
  if('no_data' %in% as.character(unique(df$variable))){
    colours = c('#CCCCCC','red', '#9999FF')
  } else {
    colours = c('red', '#9999FF')
  }
  if(max(df$value)<=1){
    ylabel <- 'Proportion of Species'
  } else {
    ylabel <- 'Number of Species'
  }
  if(!is.null(prefix)){
    title <- paste0(prefix,': ',ylabel,' which can be modelled')
  } else {
    title <- paste(ylabel,'which can be modelled')
  }
  stack_records(df,colours,ylabel,title)
}

RM <- calc_bad(RM)
```

To provide a benchmark, below are two plots for all taxa for all UK data. The first graphs shows absolute values, while the second shows the proportion of species for each taxonomic group.

```
num_spec(RM) %>% stack_taxa()
```

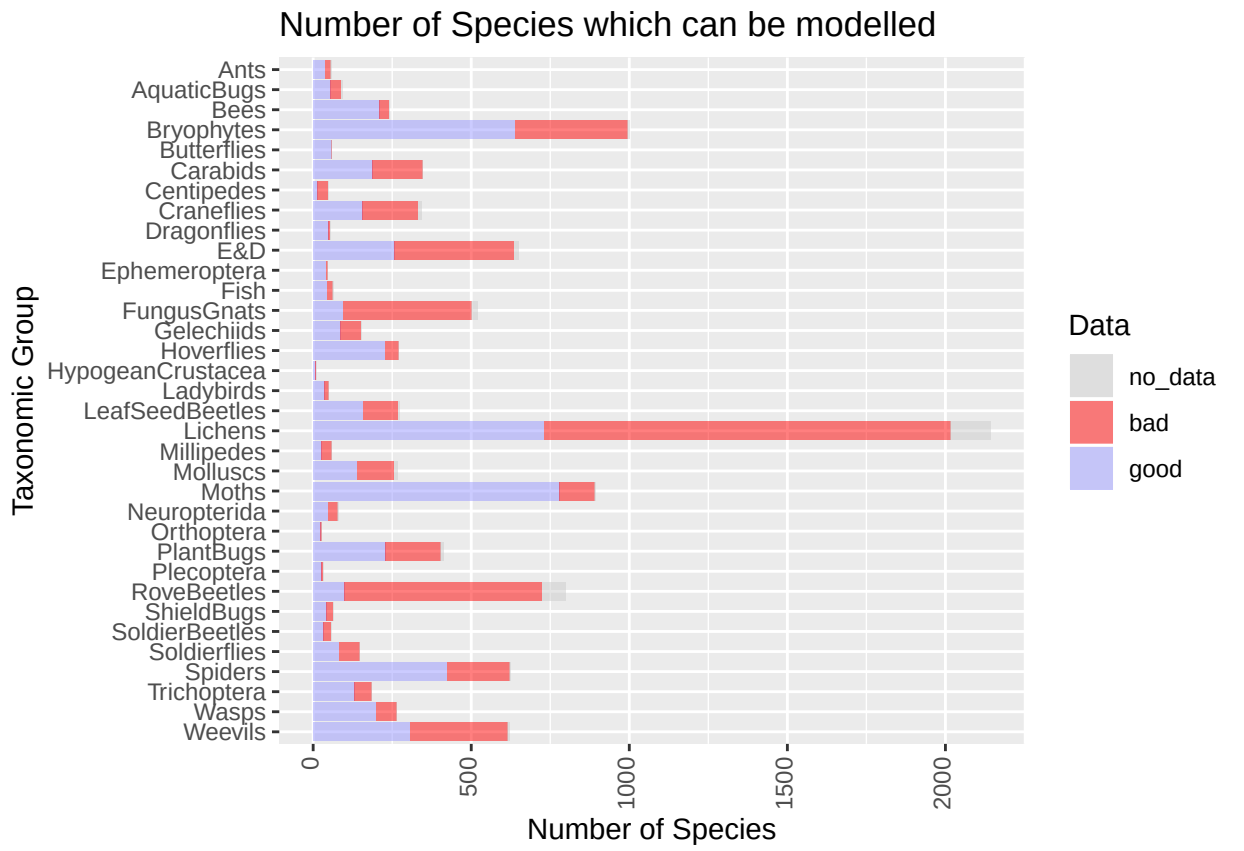


Figure 23: Plot of number of species which can be modelled, split by taxonomic group. Blue shows species which can be modelled, red those which cannot. Grey are those for which there is no data, due to there being no records which meet the requirement of coming from sites with repeat visits in any subsequent years.

```
num_spec(RM,proportional=TRUE) %>% stack_taxa()
```

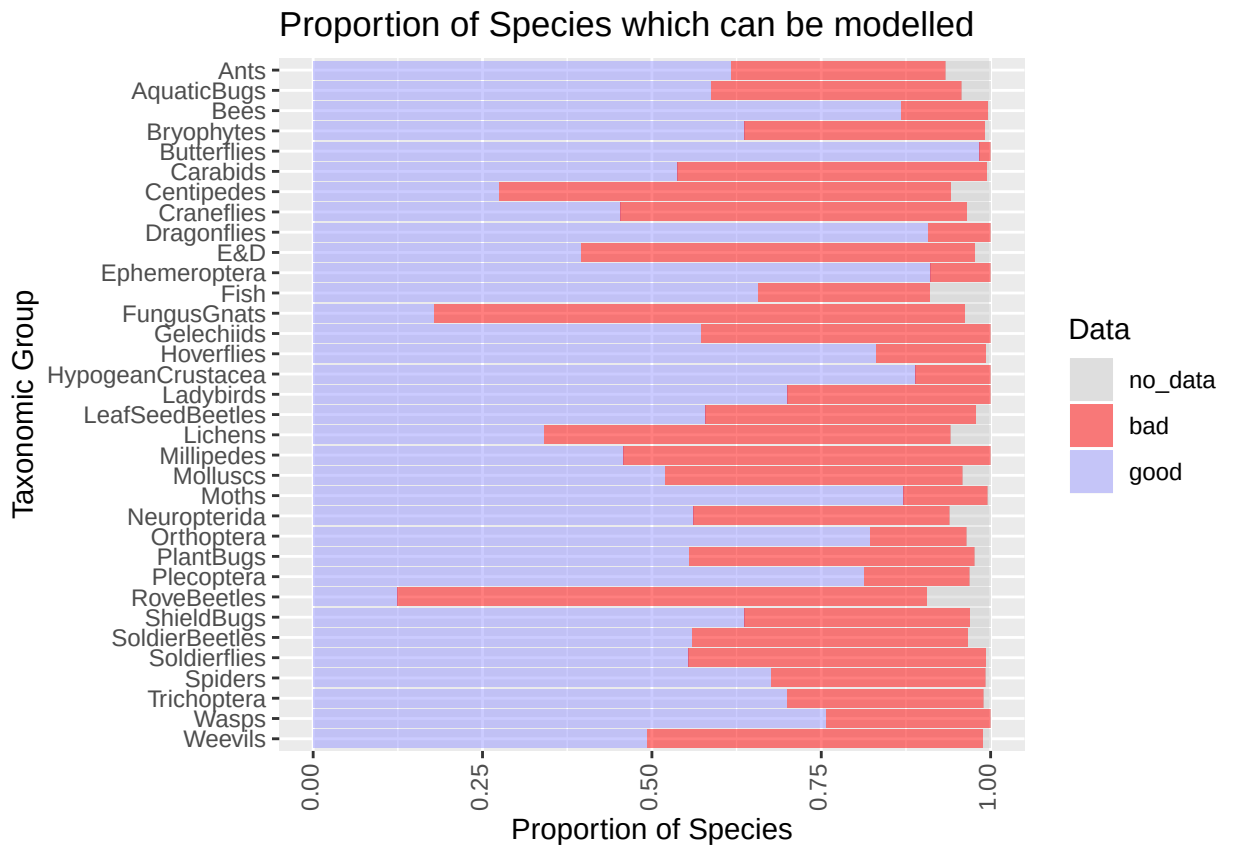


Figure 24: Plot of number of species which can be modelled, as a proportion of the total number of species for each taxonomic group.

```
# Plot aspirational graph
num_spec(RM,aspirational=TRUE,proportional=TRUE) %>%
  stack_taxa(prefix = 'Aspirational')
```

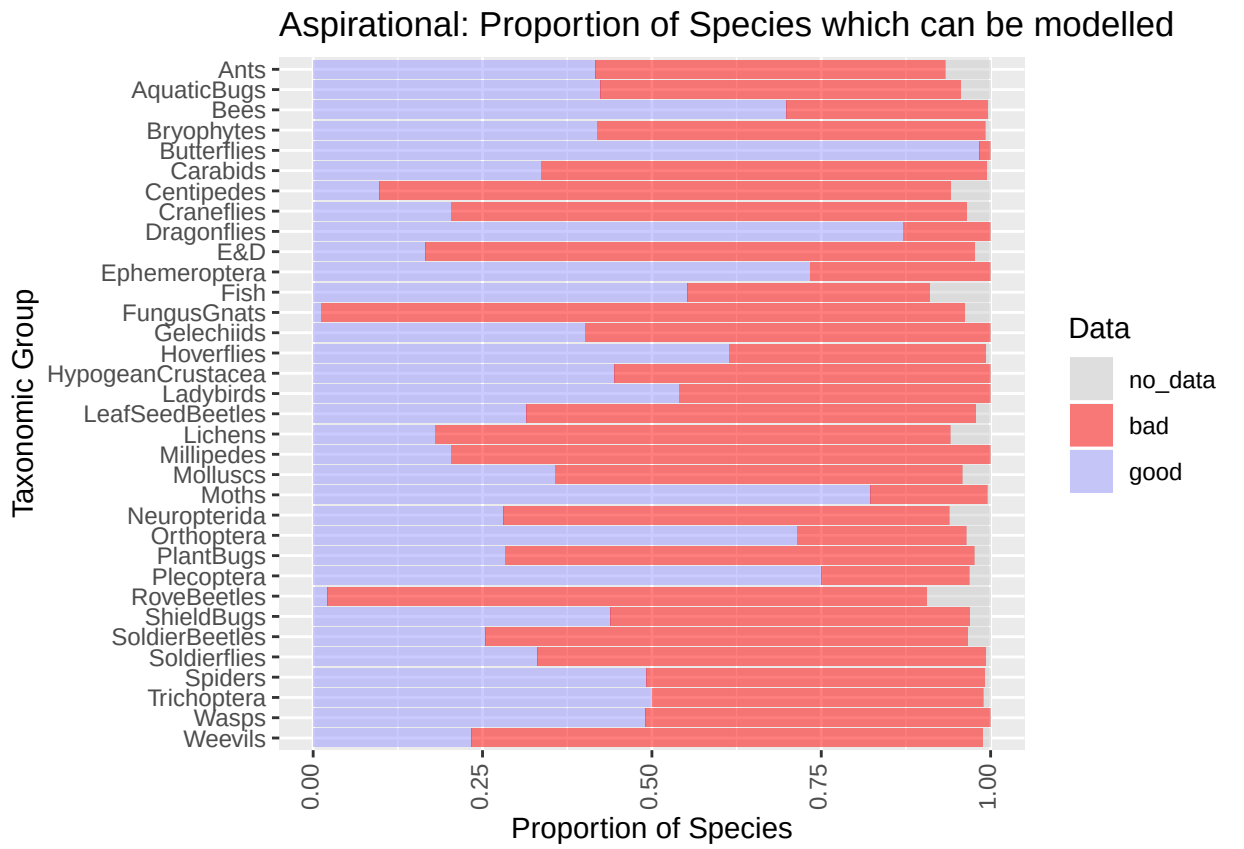


Figure 25: Plot of number of species which can be modelled using the aspirational target. As this is the aspirational target, the number of species which can be modelled is lower

Table 4: List of available habitats

Habitat	Description	Habitat_code
Broadleaf woodland	Broadleaved, Mixed and Yew Woodland	BLW
Coniferous woodland	Coniferous Woodland	CWL
Arable	Arable and Horticultural	A
Improved grassland	Improved Grassland	IG
Semi-natural grassland	Neutral Grassland, Calcareous Grassland, Acid Grassland, Fen, Marsh and Swamp	SNG
Mountain, heath, bog	Heather, Heather Grassland, Bog, Inland Rock	MHB
Coastal	Supra-littoral Rock, Supra-littoral Sediment, Littoral Rock, Littoral Sediment, Saltmarsh	C
Freshwater	Freshwater	FW
Built-up areas and gardens	Built-up Areas and Gardens	BU

By habitat

The possible habitats are shown in Table 4.

```
# Display the list of possible habitats
hab_list <- read.csv('Habitat/Habitat_List.csv')
hab_list %>%
  kable(format = 'latex',
        caption = 'List of available habitats') %>%
  column_spec(2, width = "8cm")
```

To calculate the raw metrics for each habitat, the squares in the UK all need to be assigned to a habitat. To do so, a habitat datafile is read in and the range of percentage cover for each habitat is extracted. This produces a list of coverage for each habitat such as:

- Habitat: BLW; Coverage: 1%, 1%, 2%, 3% ... 95%, 98%, 99%

From this list, the median coverage for each habitat is calculated. Each square in the UK which has above this median coverage is assigned to that habitat. Therefore it is possible for a square to contain multiple habitats or not represent any habitat.

```
# Read in land habitat coverage data file
UK_land <- readRDS(file.path('./Habitat/UK_land.rds'))
UK_land_GR <- UK_land
UK_land_GR$easting = UK_land$easting + 500
UK_land_GR$northing = UK_land$northing + 500

UK_land_GR$site <- gr_num2let(easting = UK_land_GR$easting,
                              northing = UK_land_GR$northing)
UK_land_GR$site <- paste0(substr(x = UK_land_GR$site, start = 1, stop = 4),
                          substr(x = UK_land_GR$site, start = 8, stop = 9))

## Coverage is in km squared
total_coverage <- colSums(UK_land[,3:ncol(UK_land)])

## We also want number of kmsq with any cover of each habitat
UK_land_any_hab <- ifelse(UK_land[,3:ncol(UK_land)] > 0, 1, 0)
total_coverage_squares <- colSums(UK_land_any_hab)

## Now look at coverage for each of quantiles
quants <-
```

```
  data.frame(apply(UK_land[,3:ncol(UK_land)], 2, function(x){
    quantile(x[x>0], probs = c(0.25, 0.50, 0.75))
  }))
```

Now calculate the metrics for each habitat

```
hab_names <- colnames(UK_land[,3:ncol(UK_land)])
for(data_file in data_files){
  formatted_data <- dataPrep(data_file)

  spp <- unique(taxa_data$CONCEPT)
    # one group has a species called '', best to remove this (single) record
  spp <- spp[spp != '']

  # Create combined dataframe
  species_obs <-
    formatted_data$occDetdata %>% left_join(formatted_data$spp_vis,by="visit")
  species_obs$site <- as.character(species_obs$site)
  # Combine with habitat data
  species_obs <- species_obs %>% left_join(UK_land_GR,by="site")
  ## Remove NA's
  species_obs <- species_obs[!is.na(species_obs$CWL),]

  yps <- rowSums(acast(species_obs, site ~ year, length, value.var = 'L') > 0)
  sites_to_include <- names(yps[yps >= 2])
  species_obs <- species_obs[(species_obs$site %in% sites_to_include),]

  spp_ch <- as.character(spp)
  raw_metrics <- data.frame()

  # At the group level, determine records with above average habitat coverage
  for(hab in hab_names){
    ## Find visits with above median coverage for this habitat
    above_av <- species_obs[,colnames(species_obs) == hab]>=
      quants[rownames(quants) == '50%',colnames(quants) == hab]
    species_obs[,paste0(hab,'_hab')] <- above_av
  }

  # For each species, find the visits with just that species recorded
  i <- 1
  for(sp in spp_ch){
    if((i %% 10) == 0){cat('  Species',i,'of',length(spp_ch),'\n')}
    if(sp %in% names(species_obs)){
      taxa_rec <-
        species_obs[,names(species_obs) %in% c('visit','site','L','year',
                                               sp,hab_names,
                                               paste0(hab_names,'_hab'))]
      for(hab in hab_names){
        ## Extract the metrics for each habitat
        function(sp, species_obs, basen, suffix = NULL, years = NULL)
        speciesMetric <- dataMetrics(hab = hab,
                                     sp = sp,
                                     basen = basename(data_file),
                                     species_obs = taxa_rec,
```

```
                                    habitat = TRUE)
        raw_metrics <- rbind(raw_metrics, speciesMetric)
      }
    }
    i <- i+1
  }
  group_name <- regmatches(basename(data_file),
                           regexpr('[A-z]+(?=_)',
                                   basename(data_file),
                                   perl=TRUE))
  write.csv(raw_metrics,
            file = file.path('./Habitat/HabMetrics',
                             paste0('habMetrics_',
                                    gsub('.rdata$', '.csv',
                                         basename(data_file)))),
            row.names = FALSE)
}

td <- data.frame()
file_list <- list.files(file.path('./Habitat/HabMetrics'),
                        pattern = '^habMetrics.*csv$',
                        full.names = TRUE)

for(i in file_list){
  tmp <- read.csv(i)
  td <- rbind(td,tmp)
}

write.csv(x = td, file.path('./Habitat/HabMetrics/ALL_habMetrics.csv'))
```

Now calculate the metrics for butterflies in moths. These are formatted differently and only the basic metrics are extracted, to save computational power.

```
for(data_file in list.files(jasmin,full.names = TRUE)){
  cat('Starting', basename(data_file))
  load(data_file)

  spp <- unique(taxa_data$CONCEPT)
  # one group has a species called '', best to remove this (single) record
  spp <- spp[spp != '']

  cat('\n', length(spp), 'species\n')
  num_rec <- nrow(taxa_data)
  taxa_data$year <- lubridate::year(taxa_data$TO_STARTDATE)

  # Remove sites which have not seen a repeat in subsequent years.
  # This is a requirement here as this was a step taken by the model.
  yps <- rowSums(acast(taxa_data, SQ_1KM ~ year, length, value.var = 'year') > 0)
  sites_to_include <- names(yps[yps >= 2])
  taxa_data <- taxa_data[taxa_data$SQ_1KM %in% sites_to_include,]

  Taxa <- substr(basename(data_file),1,regexpr('_',basename(data_file))[1]-1)
  # First create a habitat lookup table and use it to generate species observances
  # by habitat
```

```
  hab_names <- colnames(UK_land[,3:ncol(UK_land)])
  hab_lookup <- data.frame(site = UK_land_GR$site,
                           stringsAsFactors = FALSE)
  species_obs <- list()
  for(hab in hab_names){
    cat('..',hab,'...\n')
    above_av <-
      UK_land_GR[,names(UK_land_GR) == hab]>=
      quants[rownames(quants) == '50%',colnames(quants) == hab]
    hab_lookup[,match(hab, hab_names) + 1] <- above_av
    names(hab_lookup)[match(hab, hab_names) + 1] <- hab
    species_obs[[match(hab, hab_names)]] <-
      taxa_data[taxa_data$SQ_1KM %in% hab_lookup$site[names(hab_lookup) == hab],]
  }
  names(species_obs) <- hab_names

  # Calculate metrics by species, by habitat
  hab_metrics <- NULL
  i <- 0
  for(sp in spp){
    if((i %% 5) == 0){cat('  Habitat: Species',i,'of',length(spp),'\n')}
    for(hab in hab_names){
      num_rec <- nrow(species_obs[[match(hab, hab_names)]])
      hab_metrics <- rbind(hab_metrics,
                           simpleDataMetrics(sp = sp,
                                             tFDall =
                                               species_obs[[match(hab, hab_names)]],
                                             num_rec = num_rec,
                                             Taxa = Taxa,
                                             habitat = hab))
    }
    i <- i + 1
  }

  write.csv(hab_metrics, file = file.path('Habitat/HabMetrics',
                                          paste0('jasminHabMetrics_',
                                                 gsub('.rdata$', '',
                                                      basename(data_file)),
                                                 '.csv')),
            row.names = FALSE)
}

file_list <- list.files(file.path('./Habitat/HabMetrics'),
                        pattern = '^jasminHabMetrics.*csv$',
                        full.names = TRUE)
master <- NULL
for(i in c(file_list)){
  tmp <- read.csv(i)
  master <- rbind(master,tmp)
}
habRM <- read.csv('./Habitat/HabMetrics/ALL_habMetrics.csv')
# Remove all metrics not required
habRM <- habRM[,colnames(habRM) %in% colnames(master)]
```

```
td <- rbind(habRM,master)

write.csv(x = td, file.path('./Habitat/HabMetrics/ALL_combinedHabMetrics.csv'))
```

The raw metrics for each habitat have now been calculated from the raw data. Using these data, the proportion of species which can be modelled for each habitat can be assessed.

```
# Read in the habitat raw metrics
RM_hab <-
  read.csv(file = file.path('Habitat/HabMetrics/ALL_combinedHabMetrics.csv'))
```

```
# Calculate which datasets are likely to produce good or bad models
RM_hab <- calc_bad(RM_hab)
```

```
# Plot up a sample graph for broad leaf woodland
num_spec(RM_hab,habitat='BLW',proportional=TRUE) %>%
  stack_taxa(prefix='B-L Woodland')
```

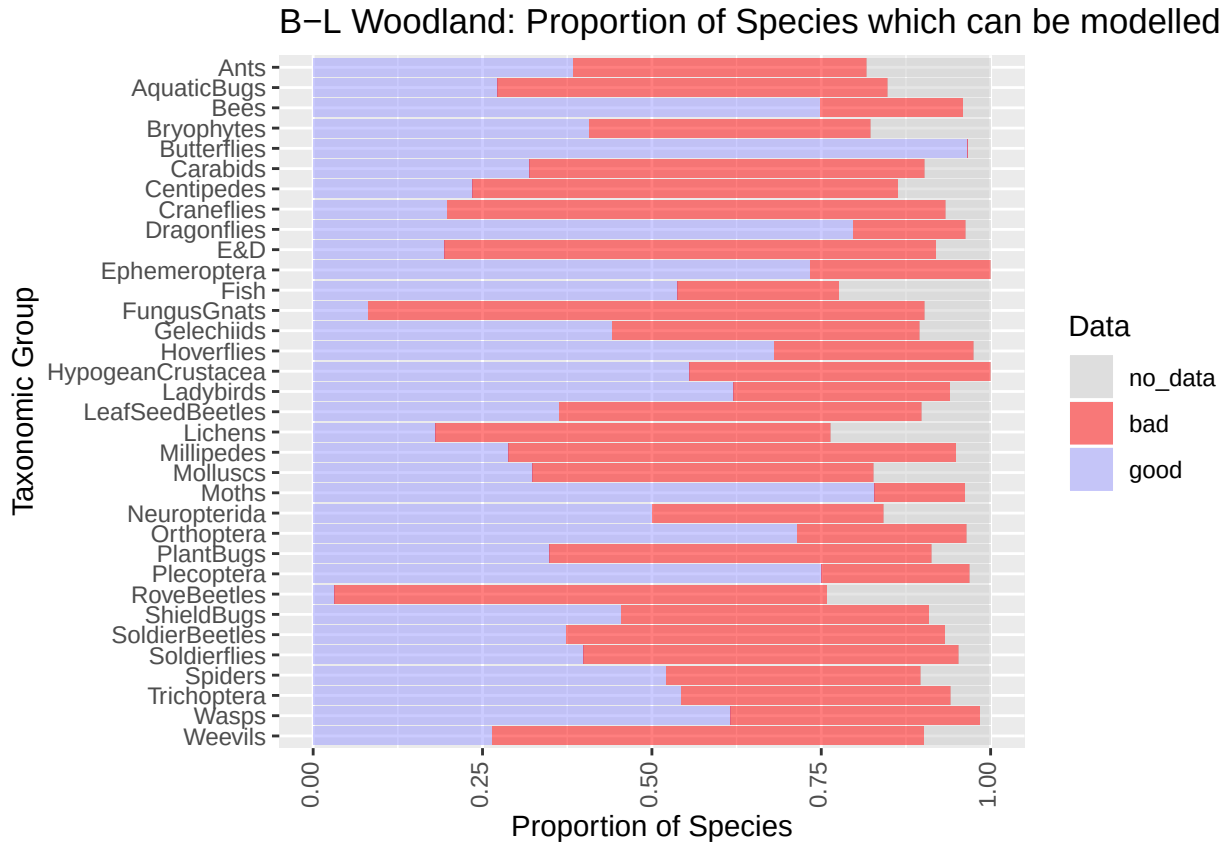


Figure 26: Proportion of species which can be modelled in broad-leaf woodland

```
# And for coastal
num_spec(RM_hab,habitat='C',proportional=TRUE) %>% stack_taxa(prefix='Coastal')
```

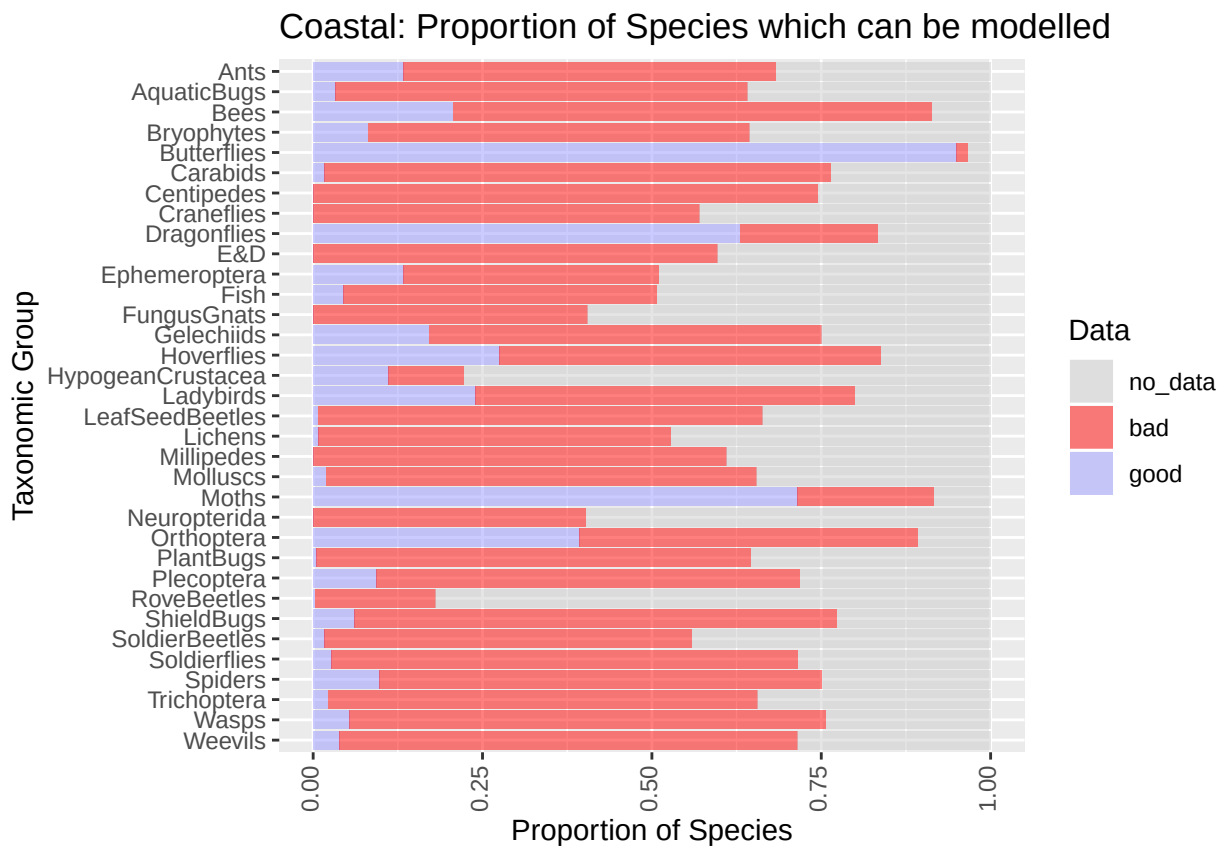


Figure 27: Proportion of species which can be modelled in coastal habitats

By region

The data can also be split by region. Splitting by region is simpler than for habitat as all squares in the UK exist within a region which can be obtained from the NUTS lookup table. The data is then subset by region and the metrics extracted for each of these regions.

```
regions <- read.csv(file.path('./Region/NUTS1_1km_dupsorted.csv'))
regions <- regions[,c(2,4)]

for(data_file in data_files){
  formatted_data <- dataPrep(data_file)
  spp <- unique(taxa_data$CONCEPT)
  spp <- spp[spp != '']

  # Create combined dataframe
  yps <- rowSums(acast(formatted_data$occDetdata,
                       site ~ year, length, value.var = 'L') > 0)
  sites_to_include <- names(yps[yps >= 2])
  formatted_data$occDetdata <-
    formatted_data$occDetdata[(formatted_data$occDetdata$site %in%
                                 sites_to_include),]
  species_obs <-
    formatted_data$occDetdata %>% left_join(formatted_data$spp_vis,by="visit")
  species_obs$site <- as.character(species_obs$site)

  raw_metrics <- data.frame()
  species_obs <- merge(x = species_obs, y = regions, by.x= 'site', by.y = 'km_sq')

  # For each species, find the visits with just that species recorded
  for(sp in as.character(spp)){
    if(sp %in% names(species_obs)){
      taxa_rec <-
        species_obs[,names(species_obs) %in% c('visit','site','nutsname',
                                               'L','year',sp)]
      nutsnames <- as.character(unique(species_obs$nutsname))
      for(region in nutsnames){
        ## Extract the metrics for each habitat
        speciesMetric <- dataMetrics(hab = region,
                                     sp = sp,
                                     basen = basename(data_file),
                                     species_obs = taxa_rec,
                                     region = TRUE)
        raw_metrics <- rbind(raw_metrics, speciesMetric)
      }
    }
  }
  group_name <- regmatches(basename(data_file),
                           regexpr('[A-z]+(?=_)',basename(data_file),perl=TRUE))
  write.csv(raw_metrics,
            file = file.path('./Region/RegMetrics',
                             paste0('regMetrics_',
                                    gsub('.rdata$', '.csv',basename(data_file)))),
            row.names = FALSE)
}
```

```
td <- data.frame()
file_list <- list.files(file.path('./Region/RegMetrics'),
                        full.names = TRUE)
for(i in file_list){
  td <- rbind(td,read.csv(i))
}
write.csv(x = td, file.path('./Region/RegMetrics/ALL_regMetrics.csv'))
```

Now include the butterfly and moth data

```
for(data_file in list.files(jasmin,full.names = TRUE)){
  cat('Starting', basename(data_file))
  load(data_file)

  spp <- unique(taxa_data$CONCEPT)
  # one group has a species called '', best to remove this (single) record
  spp <- spp[spp != '']

  cat('\n', length(spp), 'species\n')
  num_rec <- nrow(taxa_data)
  taxa_data$year <- lubridate::year(taxa_data$TO_STARTDATE)

  # Remove sites which have not seen a repeat in subsequent years.
  # This is a requirement here as this was a step taken by the model.
  yps <- rowSums(acast(taxa_data, SQ_1KM ~ year, length, value.var = 'year') > 0)
  sites_to_include <- names(yps[yps >= 2])
  taxa_data <- taxa_data[taxa_data$SQ_1KM %in% sites_to_include,]

  Taxa <- substr(basename(data_file),1,regexpr('_',basename(data_file))[1]-1)

  # Load up the regional lookup table
  regions <- read.csv(file.path('./Region/NUTS1_1km_dupsorted.csv'))
  regions <- regions[,c(2,4)]
  region_names <- unique(regions$nutsname)

  # Create a list of species observances by region
  species_obs <- list()
  for(region in region_names){
    cat('..',region,'...\n')
    species_obs[[match(region, region_names)]] <-
      taxa_data[taxa_data$SQ_1KM %in% regions$km_sq[regions$nutsname == region],]
  }
  names(species_obs) <- region_names

  # Now calculate metrics by species, by region
  reg_metrics <- NULL
  i <- 0
  for(sp in spp){
    if((i %% 5) == 0){cat('  Region: Species',i,'of',length(spp),'\n')}
    for(region in region_names){
      num_rec <- nrow(species_obs[[match(region, region_names)]])
      if(num_rec != 0){
        reg_metrics <- rbind(reg_metrics,
                             simpleDataMetrics(sp = sp,
```

```
                                                tFDall =
                                                  species_obs[[match(region,
                                                                     region_names)]],
                                                num_rec = num_rec,
                                                Taxa = Taxa,
                                                region = region))
      }
    }
    i <- i + 1
  }
  write.csv(reg_metrics, file = file.path('Region/RegMetrics',
                                          paste0('jasminRegMetrics_',
                                                 gsub('.rdata$', '',
                                                      basename(data_file)),
                                                 '.csv')),
            row.names = FALSE)
}

file_list <- list.files(file.path('./Region/RegMetrics'),
                        pattern = '^jasminRegMetrics.*csv$',
                        full.names = TRUE)
master <- NULL
for(i in c(file_list)){
  tmp <- read.csv(i)
  master <- rbind(master,tmp)
}
regRM <- read.csv('./Region/RegMetrics/ALL_regMetrics.csv')
# Remove all metrics not required
regRM <- regRM[,colnames(regRM) %in% colnames(master)]
td <- rbind(regRM,master)

write.csv(x = td, file.path('./Region/RegMetrics/ALL_combinedRegMetrics.csv'))
```

Table 5: List of regions

region	code
Northern Ireland	UKN
Scotland	UKM
North East (England)	UKC
North West (England)	UKD
Yorkshire and The Humber	UKE
Wales	UKL
West Midlands (England)	UKG
East Midlands (England)	UKF
South West (England)	UKK
South East (England)	UKJ
East of England	UKH
London	UKI

Now the data has been extracted, we can read in the region data file and query the data by region. The available regions are shown in Table 5.

```
# Read in the regional raw metrics
RM_reg <-
  read.csv(file =  file.path('Region/RegMetrics/ALL_combinedRegMetrics.csv'))
# Load the list of regions
region_lookup <- read.csv(file = file.path('Region/NUTS_lookup.csv'))
# Merge this lookup table with the raw data to allow it be queried by code
RM_reg <- merge(RM_reg,region_lookup)
# Calculate which datasets are likely to produce good or bad models
RM_reg <- calc_bad(RM_reg)
```

```
region_lookup %>% kable(caption = 'List of regions')
```

```
# Plot up a sample graph for Scotland
num_spec(RM_reg,region='Scotland',proportional=T) %>% stack_taxa(prefix='Scotland')
```

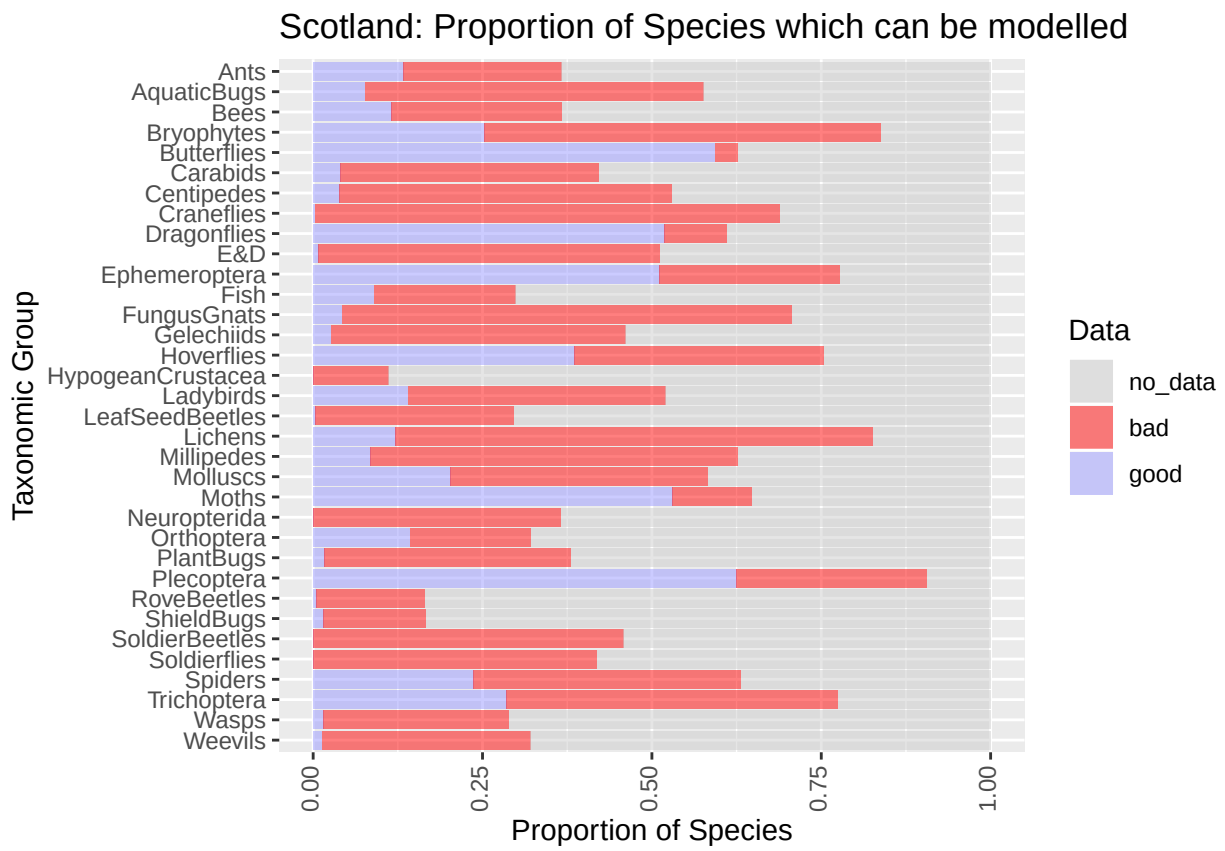


Figure 28: Proportion of species which can be modelled in Scotland

```
# And for Wales
num_spec(RM_reg,region='UKL',proportional=TRUE) %>% stack_taxa(prefix='Wales')
```

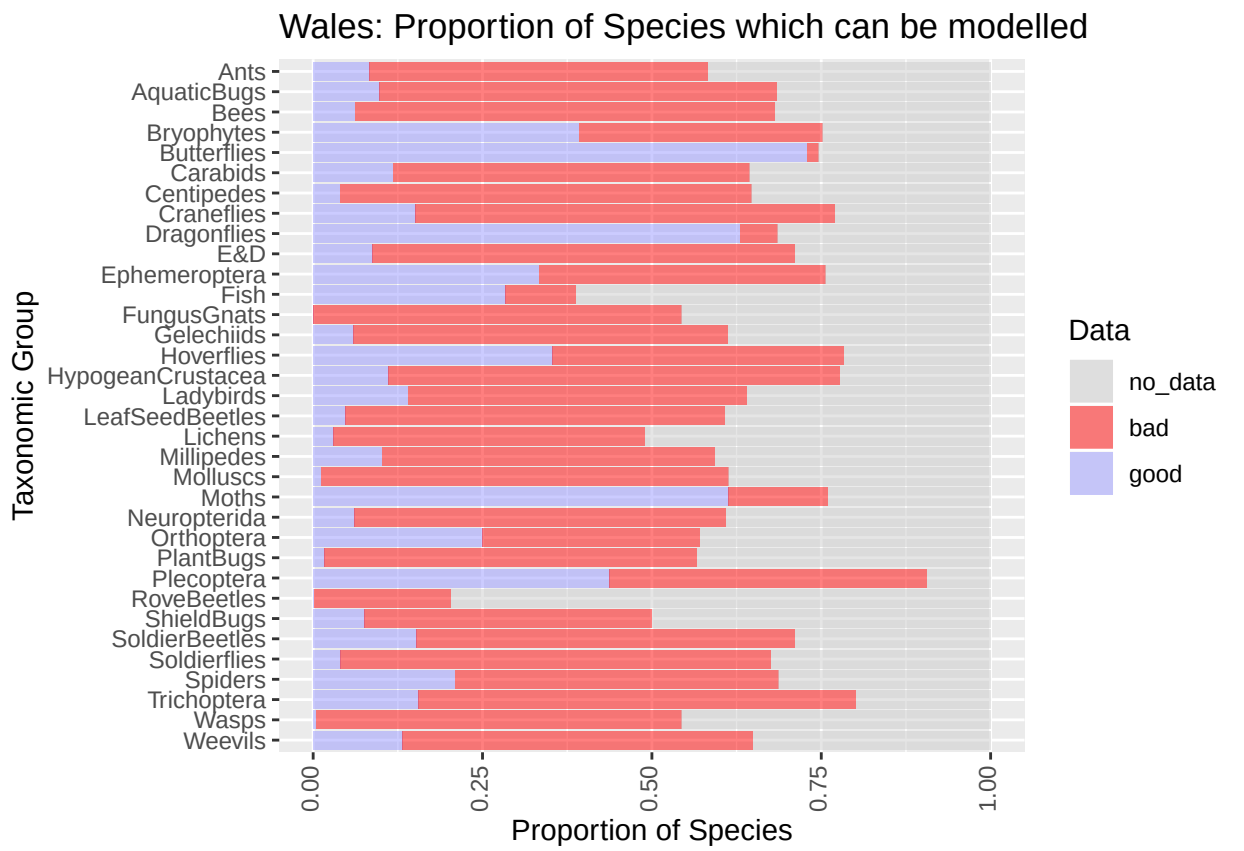


Figure 29: Proportion of species which can be modelled in Wales

Conclusions

Previously it was difficult to predict from a dataset whether it was possible to produce an occupancy model with a useful level of precision. This work has demonstrated that the criteria for making such an assessment are relatively simple and easy to apply to new datasets.

With this tool, it can be estimated how many species we can model, not just in the UK as a whole, but for any region or habitat of interest within the UK. The aspirational criteria also represent a target to aim for, to enable the modelling of any species of interest.