TSDA_Analysis_RulesofThumb

Tom August & Mark Logie

19 September, 2018

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 understnding 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. We propose this includes:

- The 50th/90th percentile of the number of records of the focal species per year [median/P90]
- The 50th/70th/80th/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/P70/P80/P90]
- The proportion of years in which the focal species has been recorded [prop_of_years]
- 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 'Examining the Classification Variables'

Model quality metrics

These metrics measure the quality of the model produce and are what we will us to develop the rules of thumb. Currently we have two suggestions:

- 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':    13640 obs. of  12 variables:
##  $ speciesName      : Factor w/ 9266 levels "","" Lamprochromus bifasciatus",..: 5461 5463 5465 5467 5
##  $ Spnsite          : int  51 336 16 852 237 75 11 0 355 6 ...
##  $ Spnvisits        : int  94 785 35 1804 450 168 50 0 965 45 ...
##  $ SpvisitPerSite   : num  1.84 2.34 2.19 2.12 1.9 ...
##  $ SpvisitPerYear   : num  2 16.702 0.745 38.383 9.574 ...
##  $ Totalnvisits     : int  11260 11260 11260 11260 11260 11260 11260 11260 11260 11260 ...
##  $ Totalnsites      : int  2439 2439 2439 2439 2439 2439 2439 2439 2439 2439 ...
##  $ TotalvisitPerYear: num  240 240 240 240 240 ...
##  $ nyears           : int  47 47 47 47 47 47 47 47 47 47 ...
##  $ 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 ...
```

```
head(model_data)
```

```
##           speciesName Spnsite Spnvisits SpvisitPerSite SpvisitPerYear
## 1   FORMICA aquilonia      51        94       1.843137      2.0000000
## 2 FORMICA cunicularia     336       785       2.336310     16.7021277
## 3     FORMICA exsecta      16        35       2.187500      0.7446809
## 4       FORMICA fusca     852      1804       2.117371     38.3829787
## 5      FORMICA lemani     237       450       1.898734      9.5744681
## 6    FORMICA lugubris      75       168       2.240000      3.5744681
##   Totalnvisits Totalnsites TotalvisitPerYear nyears FirstYrConverged
## 1        11260        2439          239.5745     47            FALSE
## 2        11260        2439          239.5745     47             TRUE
## 3        11260        2439          239.5745     47             TRUE
## 4        11260        2439          239.5745     47             TRUE
## 5        11260        2439          239.5745     47             TRUE
## 6        11260        2439          239.5745     47             TRUE
##   LastYrConverged PropYrConverged
## 1           FALSE               0
## 2            TRUE               1
## 3            TRUE               1
## 4            TRUE               1
## 5            TRUE               1
## 6            TRUE               1
```

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 weather 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 hte 'final 10 year' 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')

  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 I have removed them from this. We can work with them seperatly by using the visit data files which I assume are on JASMIN where they will have been created.

```
rawDataMetrics <- function(sp, tFDall, basen, suffix = NULL, years = NULL){

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

  timetaken <- system.time({
    tFDall <- tFDall[,c('visit','site','L','year',as.character(sp))]
    tFD <- tFDall[tFDall[, as.character(sp)],]

    df <- data.frame(species = paste0(sp, suffix),
                     avVisitPerYear = 0,
                     Spnvisits = 0,
                     Spnsite = 0,
                     SpvisitPerSite = 0,
                     SpvisitPerYear = 0,
                     Totalnvisits = 0,
                     Totalnsites = 0,
                     TotalvisitPerYear = 0,
                     median = 0,
                     P70 = 0,
                     P80 = 0,
                     P90 = 0,
                     sdVisitsPerYear = 0,
                     coeffVar = 0,
                     prop_repeats_spc = 0,
                     prop_repeats_grp = 0,
                     visits_median = 0,
                     visits_P70 = 0,
                     visits_P80 = 0,
                     visits_P90 = 0,
                     prop_of_years = 0,
                     prop_abs_list = 0,
                     prop_abs = 1,
                     prop_list_one = 0,
```

```
                  Taxa = NA,
                  Taxa_Root = NA,
                  stringsAsFactors = FALSE)

# Extract taxonomic name from basename
tax_nom <- substr(basen,start = 12,stop = (regexpr('_17',basen)[1]-1))
df[,'Taxa_Root'] <- tax_nom

if(!is.null(years)){
  # There is a limit to the number of year's data we want
  # Set taxonomic name with this information included
  df[,'Taxa'] <- paste0('last',years,'yrs',tax_nom)
  #  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 {
  # We want all the data, so set taxa equal to base taxonomic name
  df[,'Taxa'] <- tax_nom
}
if(nrow(tFD)!=0){
  # Number of records of species
  df[, 'Spnvisits'] <- nrow(tFD)
  # Number of sites for species
  df[, 'Spnsite'] <- length(unique(tFD$site))
  df[, 'SpvisitPerSite'] <- df$Spnvisits/df$Spnsite
  df[, 'SpvisitPerSite'][is.na(df[, 'SpvisitPerSite'])] <- 0
  total_years <- length(min(tFD$year):max(tFD$year))
  df[, 'SpvisitPerYear'] <- df$Spnvisits/total_years
  df[, 'SpvisitPerYear'][is.na(df[, 'SpvisitPerYear'])] <- 0

  # proportion of years with data
  df[, 'prop_of_years'] <- length(unique(tFD$year))/total_years
  # average visits per year
  visits_year <-
    tapply(tFD$visit, tFD$year, FUN = function(x) length(unique(x)))
  df[, 'avVisitPerYear'] <- mean(visits_year)

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

```
  # sd visits per year
  df[, 'sdVisitsPerYear'] <- sd(visits_year)

  # coefficient of variation for the visits
  df[, 'coeffVar'] <- 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
  df[, '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]

  # What proportion of site year combinations are repeated for the species
  # of interest within the whole group
  df[, '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))
  df[, 'visits_median'] <- as.numeric(percentiles[1])
  df[, 'visits_P70'] <- as.numeric(percentiles[2])
  df[, 'visits_P80'] <- as.numeric(percentiles[3])
  df[, 'visits_P90'] <- as.numeric(percentiles[4])
  # lists of length 1
  df[, 'prop_list_one'] <- sum(tFD$L == 1) / nrow(tFD)
}

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

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

# Number of sites for taxonomic group
df[, 'Totalnsites'] <- length(unique(tFDall$site))
# Number of visits for taxonomic group
df[, 'Totalnvisits'] <- nrow(tFDall)

total_years <- length(min(tFDall$year):max(tFDall$year))
df[, 'TotalvisitPerYear'] <- df$Totalnvisits/total_years
```

```
    df[, 'TotalvisitPerYear'][is.na(df[, 'TotalvisitPerYear'])] <- 0

  })
  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!
data_files <- data_files[!grepl('Moths', data_files)]
data_files <- data_files[!grepl('VascPlants', data_files)]

for(data_file in data_files){
  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)

  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 = rawDataMetrics,
                        tFDall = 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 = rawDataMetrics,
                        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

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 threholds that I think 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')]

# These merges drop a LOT of species that dont match
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            Lamprochromus bifasciatus             TRUE            TRUE
## 2  Lamprochromus bifasciatus_last10yrs             TRUE            TRUE
## 3           Acanthosoma haemorrhoidale             TRUE            TRUE
## 4                Achalcus bimaculatus             TRUE            TRUE
## 5       Achalcus bimaculatus_last10yrs            FALSE            TRUE
## 6                Achalcus britannicus             TRUE           FALSE
##   PropYrConverged mean_growth_rate precision_growth_rate
## 1       1.0000000        2.9382204           0.024610675
## 2       1.0000000      -12.0366451           0.002793217
## 3       1.0000000        0.1742559           0.847053523
## 4       0.6382979        0.2458987           0.610197450
## 5       0.8000000        0.5830347           0.055452645
## 6       0.7234043        1.3467195           0.047072983
##   mean_year_precision avVisitPerYear Spnvisits Spnsite SpvisitPerSite
## 1           999.99334       3.571429        25      18       1.388889
## 2            83.24288       5.000000        20      14       1.428571
## 3           158.84711      26.534884      1141     760       1.501316
## 4            21.48738       1.000000         3       3       1.000000
## 5            21.36882       1.000000         1       1       1.000000
## 6         86108.00740       7.833333        47       5       9.400000
##   SpvisitPerYear Totalnvisits Totalnsites TotalvisitPerYear median  P70
## 1      0.8928571        13201        2135          280.8723    2.0  3.8
## 2      3.3333333         4765        1172          297.8125    5.0  7.2
## 3     24.8043478        10914        2441          232.2128   12.0 36.8
## 4      0.1153846        13201        2135          280.8723    1.0  1.0
## 5      1.0000000         3650         949          280.7692    1.0  1.0
## 6      3.9166667        13201        2135          280.8723    4.5  9.5
##    P80  P90 sdVisitsPerYear  coeffVar prop_repeats_spc prop_repeats_grp
## 1  6.2  7.8        3.154739 0.8833270       0.20000000        0.6000000
## 2  7.8  8.4        3.651484 0.7302967       0.18750000        0.5625000
## 3 52.8 69.0       29.763631 1.1216794       0.08442211        0.3939698
## 4  1.0  1.0        0.000000 0.0000000       0.00000000        1.0000000
## 5  1.0  1.0              NA        NA       0.00000000        1.0000000
## 6 13.0 18.0        8.681398 1.1082635       0.71428571        0.8571429
##   visits_median visits_P70 visits_P80 visits_P90 prop_of_years
## 1             2        3.0        3.4        6.0     0.2500000
## 2             2        3.0        3.0        6.0     0.6666667
```

```
## 3             1        2.0        3.0        5.0     0.9347826
## 4             2        9.6       13.4       17.2     0.1153846
## 5            21       21.0       21.0       21.0     1.0000000
## 6             9       21.8       24.2       27.8     0.5000000
##   prop_abs_list  prop_abs prop_list_one         Taxa  Taxa_Root
## 1     0.5651943 0.9981062    0.08000000          E&D        E&D
## 2     0.6307692 0.9958027    0.00000000 last10yrsE&D        E&D
## 3     0.2566254 0.8954554    0.52673094   ShieldBugs ShieldBugs
## 4     0.5658433 0.9997727    0.33333333          E&D        E&D
## 5     0.6440121 0.9997260    1.00000000 last10yrsE&D        E&D
## 6     0.5645431 0.9964397    0.06382979          E&D        E&D
```

```
str(trendsData)
```

```
## 'data.frame':    9323 obs. of  33 variables:
##  $ speciesName          : Factor w/ 9266 levels "","," Lamprochromus bifasciatus",..: 2 3 4 5 6 7 8 9 1
##  $ FirstYrConverged     : logi  TRUE TRUE TRUE TRUE FALSE TRUE ...
##  $ LastYrConverged      : logi  TRUE TRUE TRUE TRUE TRUE FALSE ...
##  $ PropYrConverged      : num  1 1 1 0.638 0.8 ...
##  $ mean_growth_rate     : num  2.938 -12.037 0.174 0.246 0.583 ...
##  $ precision_growth_rate: num  0.02461 0.00279 0.84705 0.6102 0.05545 ...
##  $ mean_year_precision  : num  1000 83.2 158.8 21.5 21.4 ...
##  $ avVisitPerYear       : num  3.57 5 26.53 1 1 ...
##  $ Spnvisits            : int  25 20 1141 3 1 47 11 128 71 62 ...
##  $ Spnsite              : int  18 14 760 3 1 5 4 66 34 45 ...
##  $ SpvisitPerSite       : num  1.39 1.43 1.5 1 1 ...
##  $ SpvisitPerYear       : num  0.893 3.333 24.804 0.115 1 ...
##  $ Totalnvisits         : int  13201 4765 10914 13201 3650 13201 7115 13201 3463 13201 ...
##  $ Totalnsites          : int  2135 1172 2441 2135 949 2135 1602 2135 888 2135 ...
##  $ TotalvisitPerYear    : num  281 298 232 281 281 ...
##  $ median               : num  2 5 12 1 1 4.5 2 2 2 2 ...
##  $ P70                  : num  3.8 7.2 36.8 1 1 9.5 3.3 3.6 5.6 3.4 ...
##  $ P80                  : num  6.2 7.8 52.8 1 1 13 4.2 5 14.6 4 ...
##  $ P90                  : num  7.8 8.4 69 1 1 18 5.1 10.8 18.2 5 ...
##  $ sdVisitsPerYear      : num  3.15 3.65 29.76 0 NA ...
##  $ coeffVar             : num  0.883 0.73 1.122 0 NA ...
##  $ prop_repeats_spc     : num  0.2 0.1875 0.0844 0 0 ...
##  $ prop_repeats_grp     : num  0.6 0.562 0.394 1 1 ...
##  $ visits_median        : num  2 2 1 2 21 9 6.5 2 2 2 ...
##  $ visits_P70           : num  3 3 2 9.6 21 21.8 8.1 2 2 3 ...
##  $ visits_P80           : num  3.4 3 3 13.4 21 ...
##  $ visits_P90           : num  6 6 5 17.2 21 ...
##  $ prop_of_years        : num  0.25 0.667 0.935 0.115 1 ...
##  $ prop_abs_list        : num  0.565 0.631 0.257 0.566 0.644 ...
##  $ prop_abs             : num  0.998 0.996 0.895 1 1 ...
##  $ prop_list_one        : num  0.08 0 0.527 0.333 1 ...
##  $ Taxa                 : Factor w/ 64 levels "Ants","AquaticBugs",..: 9 25 58 9 25 9 25 9 25 9 ...
##  $ Taxa_Root            : Factor w/ 32 levels "Ants","AquaticBugs",..: 9 9 26 9 9 9 9 9 9 9 ...
```

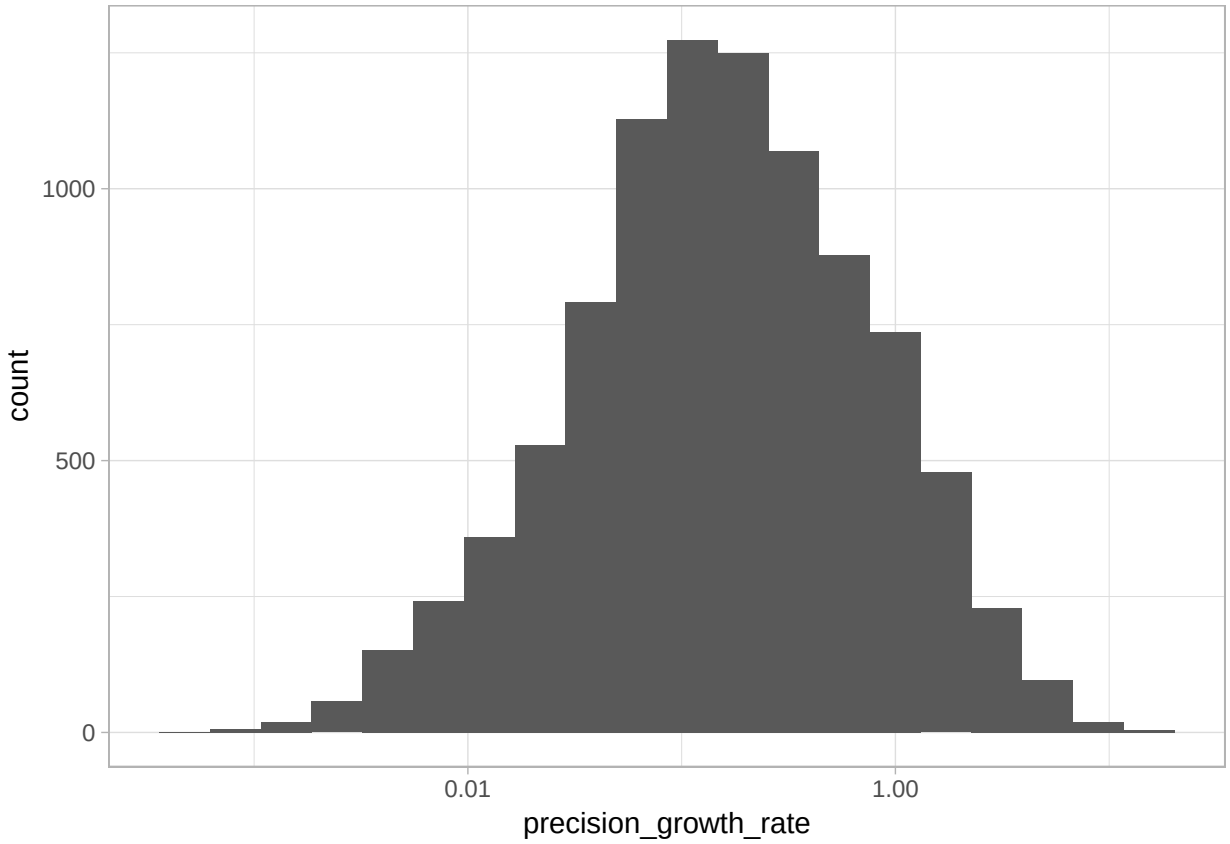
Next, define which of the species have 'good' models and which are 'bad'.

This was done by consultation with 3 experts on the models, testing them on 100 models qhich 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.

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

```
##           0%          25%          50%          75%         100%
##  0.000582423  0.054378193  0.153248562  0.472834262 18.948317680
```

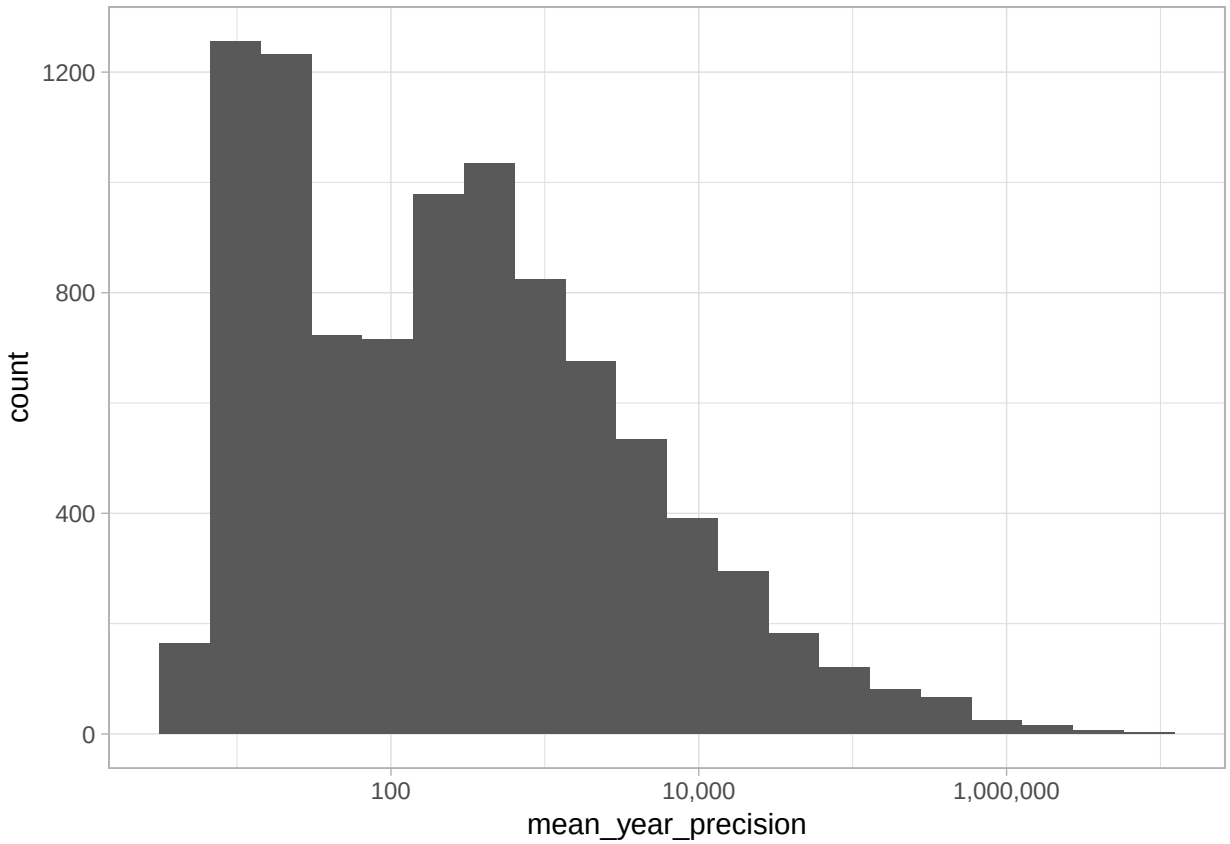
```r
ggplot(trendsData, aes(x = precision_growth_rate)) + geom_histogram(bins=20) +
  scale_x_log10(labels=scales::comma) + theme_light()
```




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

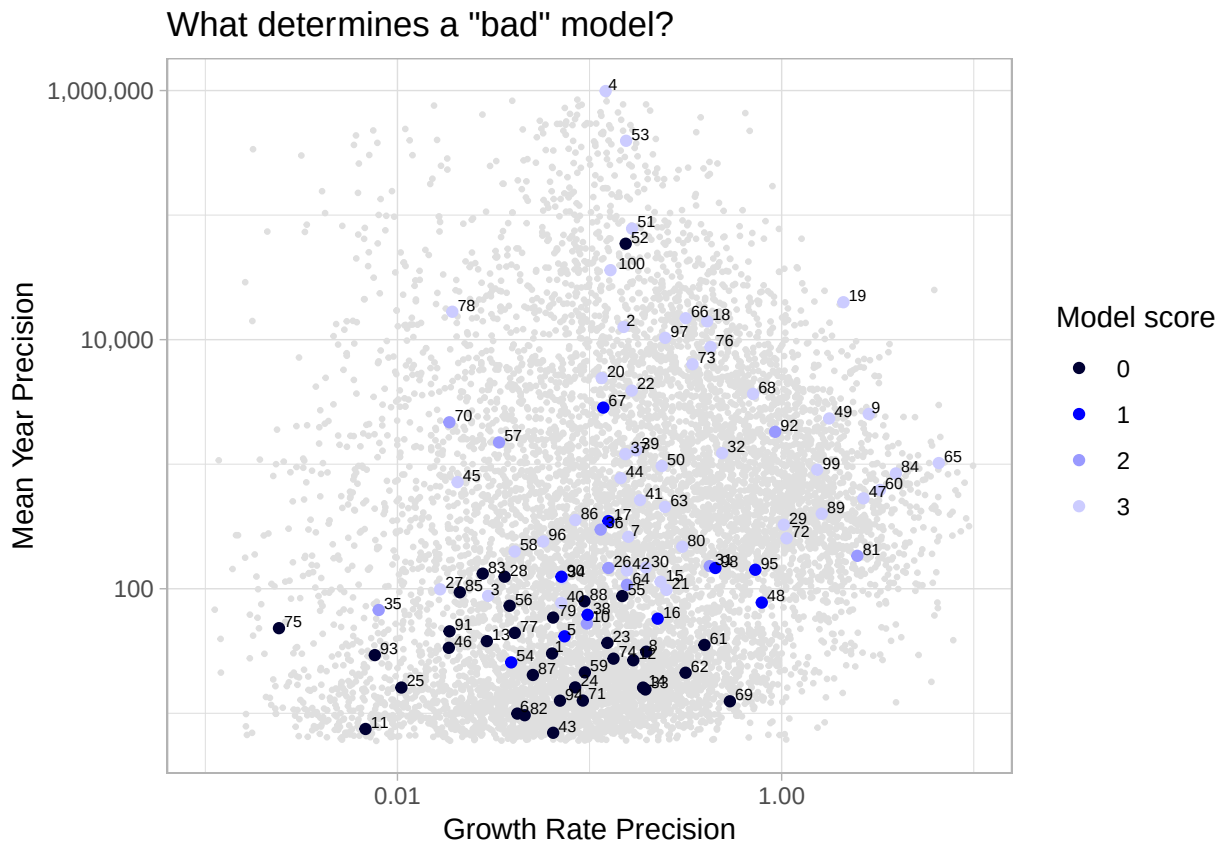
```
##           0%          25%          50%          75%         100%
## 4.721129e+00 2.451261e+01 2.197646e+02 1.443673e+03 8.739698e+06
```

```r
ggplot(trendsData, aes(x = mean_year_precision)) + geom_histogram(bins=20) +
  scale_x_log10(labels=scales::comma) + theme_light()
```



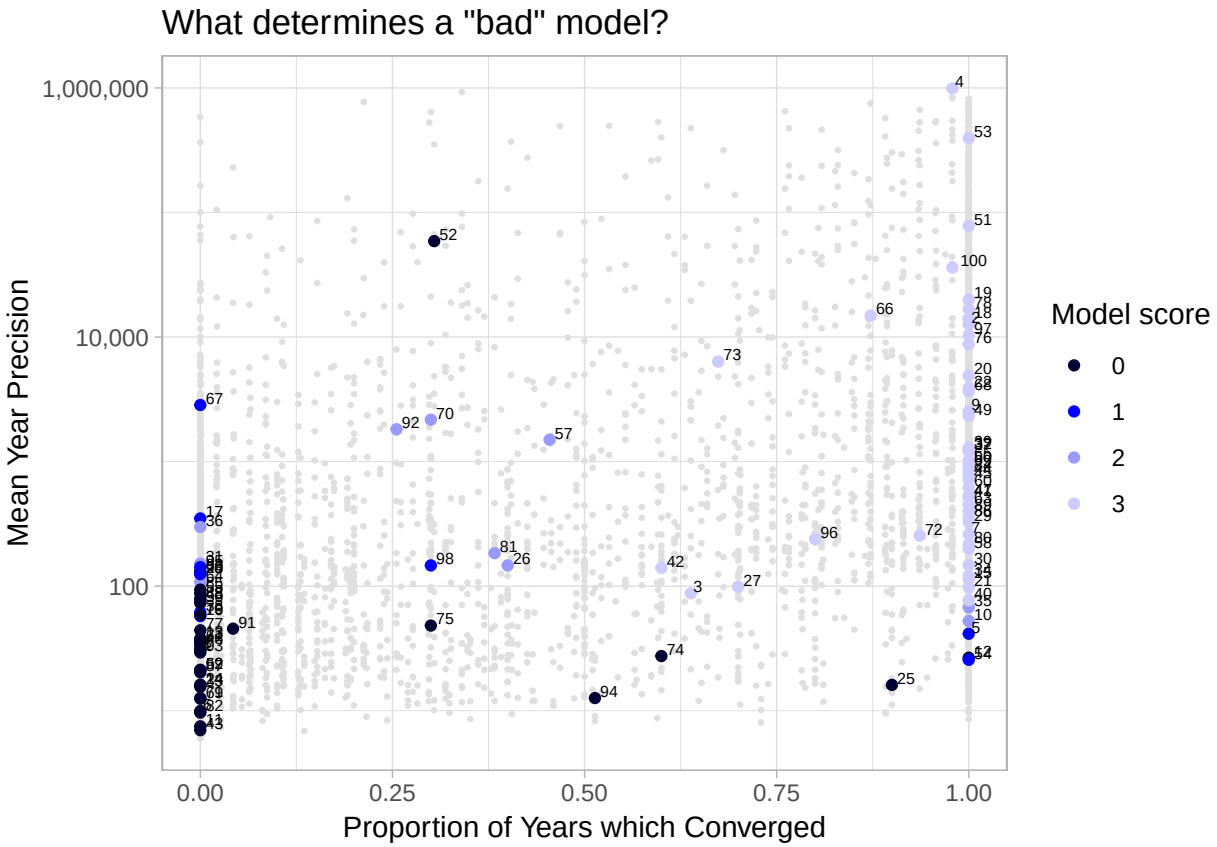

```
# Lets look at the results of the consultation with model experts
fsdf <- read.csv(file = 'Results/Consultation/fsdf_full.csv')

# Plot 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
ggplot() +
  geom_point(data=trendsData[trendsData$mean_year_precision>6&
                               trendsData$mean_year_precision<1000000&
                               trendsData$precision_growth_rate>0.001&
                               trendsData$precision_growth_rate<10,],
             aes(x=precision_growth_rate,
                 y=mean_year_precision),
             color='#DFDFDF',
             size=.5) +
  geom_point(data=fsdf,
             aes(x=precision_growth_rate,
                 y=mean_year_precision,
                 color=as.factor(score))) +
  geom_text(data=fsdf,
            aes(x=precision_growth_rate,
                y=mean_year_precision,
                label=paste0(fsdf$ID)),
            hjust=-0.3, vjust=-0.1, size=2) +
  scale_x_log10(limits=c(0.001,10),labels=scales::comma) +
  scale_y_log10(limits=c(6,1000000),labels = scales::comma) +
```

```
  theme_light() +
  scale_color_manual(values=c('#000033','#0000ff',"#9999ff", "#ccccff"),
                     name='Model score') +
  labs(title='What determines a "bad" model?') +
  xlab("Growth Rate Precision") + ylab("Mean Year Precision")
```




```
ggplot() +
  geom_point(data=trendsData[trendsData$mean_year_precision>6&
                             trendsData$mean_year_precision<1000000,],
             aes(x=PropYrConverged,
                 y=mean_year_precision),
             color='#DFDFDF',
             size=.5) +
  geom_point(data=fsdf,
             aes(x=PropYrConverged,
                 y=mean_year_precision,
                 color=as.factor(score))) +
  geom_text(data=fsdf,
            aes(x=PropYrConverged,
                y=mean_year_precision,
                label=paste0(fsdf$ID)),
            hjust=-0.3, vjust=-0.1, size=2) +
  scale_x_continuous(limits=c(0,1),labels=scales::comma) +
  scale_y_log10(limits=c(6,1000000),labels = scales::comma) +
  theme_light() +
  scale_color_manual(values=c('#000033','#0000ff','#9999ff', "#ccccff"),
```

```
                    name='Model score') +
  labs(title='What determines a "bad" model?') +
  xlab("Proportion of Years which Converged") + ylab("Mean Year Precision")
```



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.

The score shown on the graphs above are the number of experts who viewed the model as a good model. Therefore, a score of 3 means the model was deemed unanimously acceptable, a score of 0 means the model was deemed unanimously unacceptable.

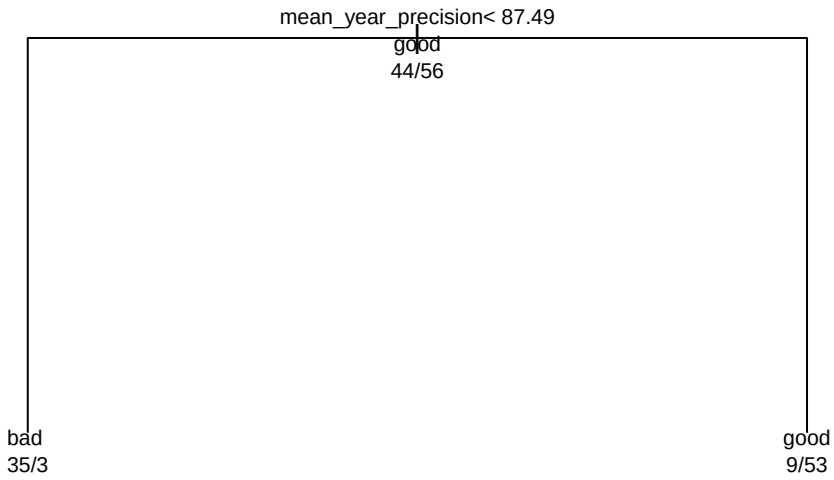
Plotting these scores against proportion of years which converged, mean year precision and growth rate precision show that these parameters are correlated with quality, but it is not obvious how to perform this split.

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 convergence rates, and the other not, for comparison.

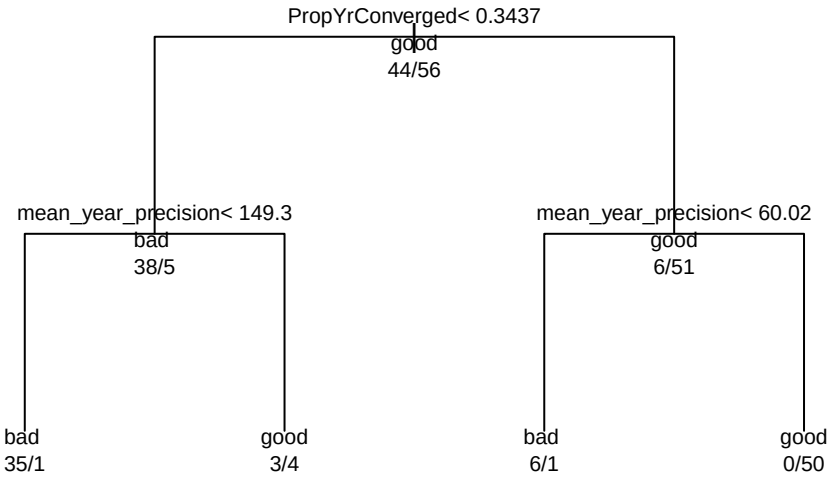
```
fsdf$good <- rep('bad',nrow(fsdf))
fsdf$good[fsdf$score>=2] <- 'good'

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

```
plot(fit_fsdf, uniform=TRUE,
     main = "Which models are good or bad (precision only)?",
     margin = .1)
text(fit_fsdf, use.n = TRUE,
     all = TRUE,
     cex = .7)
```




```
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)
```

Which models are good or bad (precision and convergence)?



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

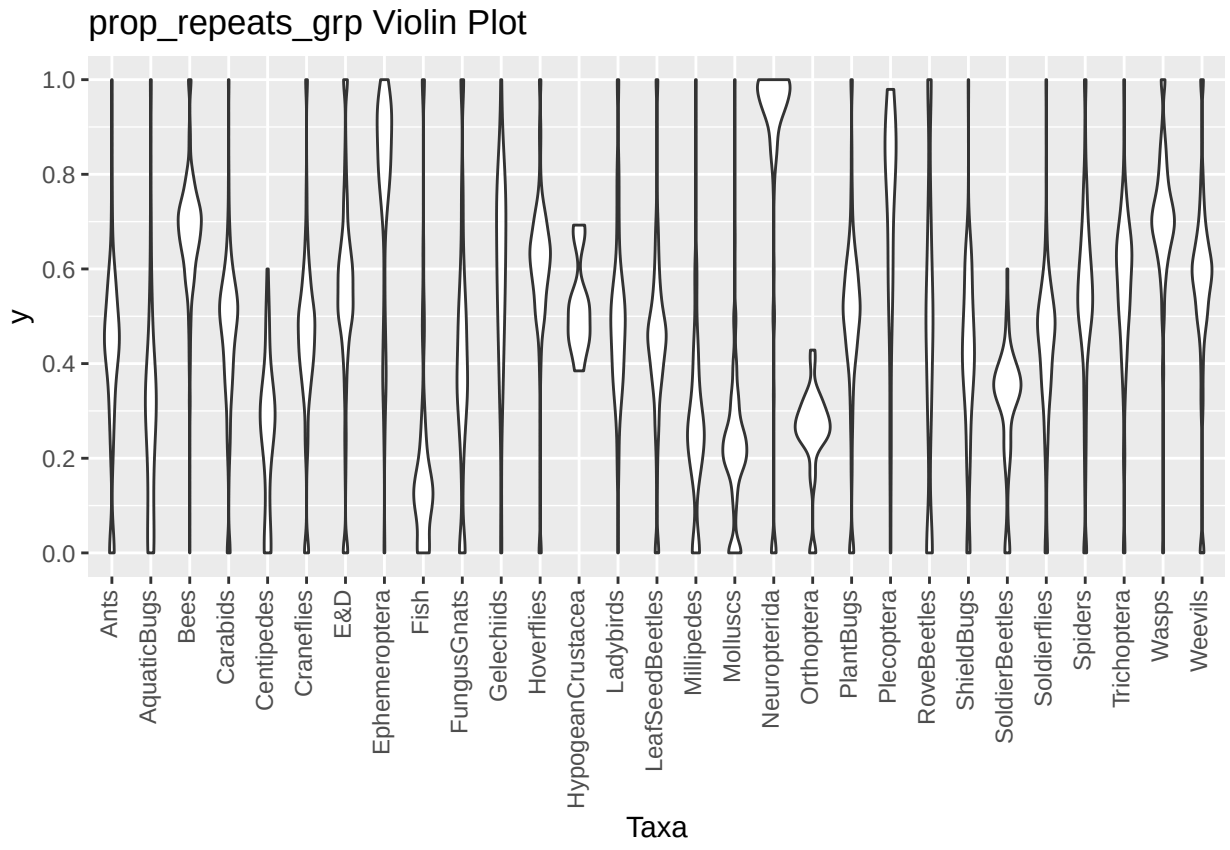
Before running the classifiers, it is important to determine whether or not any of the variables are highly taxa dependent. If they are, they are much less generalisable across data sets.

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

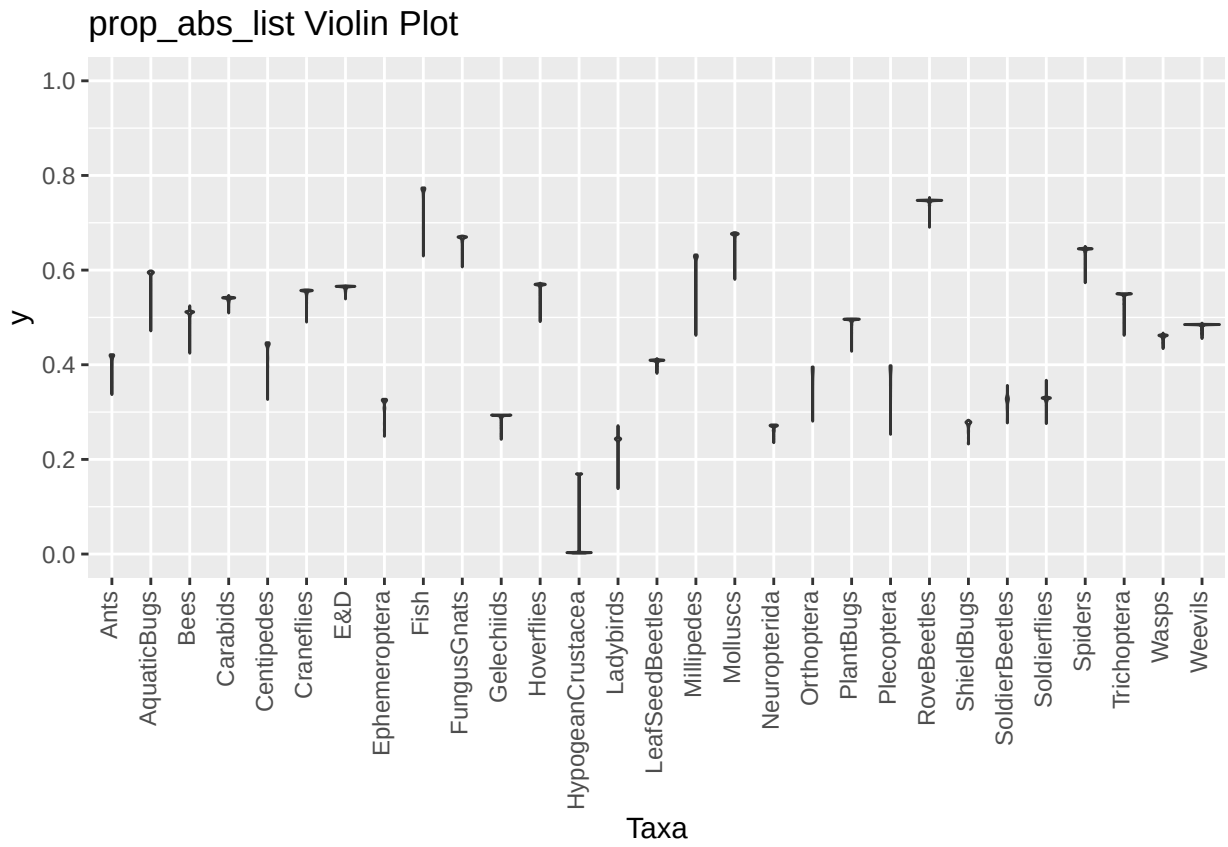
- First is the violin plot for prop_repeats_grp: the proportion of site:year combinations for the species of interest which have more than one visit.
- Second is the violin plot for prop_abs_list: the proportion of data for the taxonomic group *not including the species of interest* which have a list 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 this variable is effectively a taxonomic division. For this reason, prop_abs_list will not be used for producing decision trees.

```
violin <- function(y,title=NULL){
ggplot(td_taxa, aes(x=Taxa,y=y)) +
  geom_violin(trim=TRUE) +
  scale_y_continuous(limits = c(0,1),breaks=c(seq(0,1,.2))) +
  theme(axis.text.x = element_text(angle=90, hjust=1, vjust = 0.3)) +
  ggtitle(title)
}
# 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),]
violin(td_taxa$prop_repeats_grp,title='prop_repeats_grp Violin Plot')
```




```
violin(td_taxa$prop_abs_list,title='prop_abs_list Violin Plot')
```

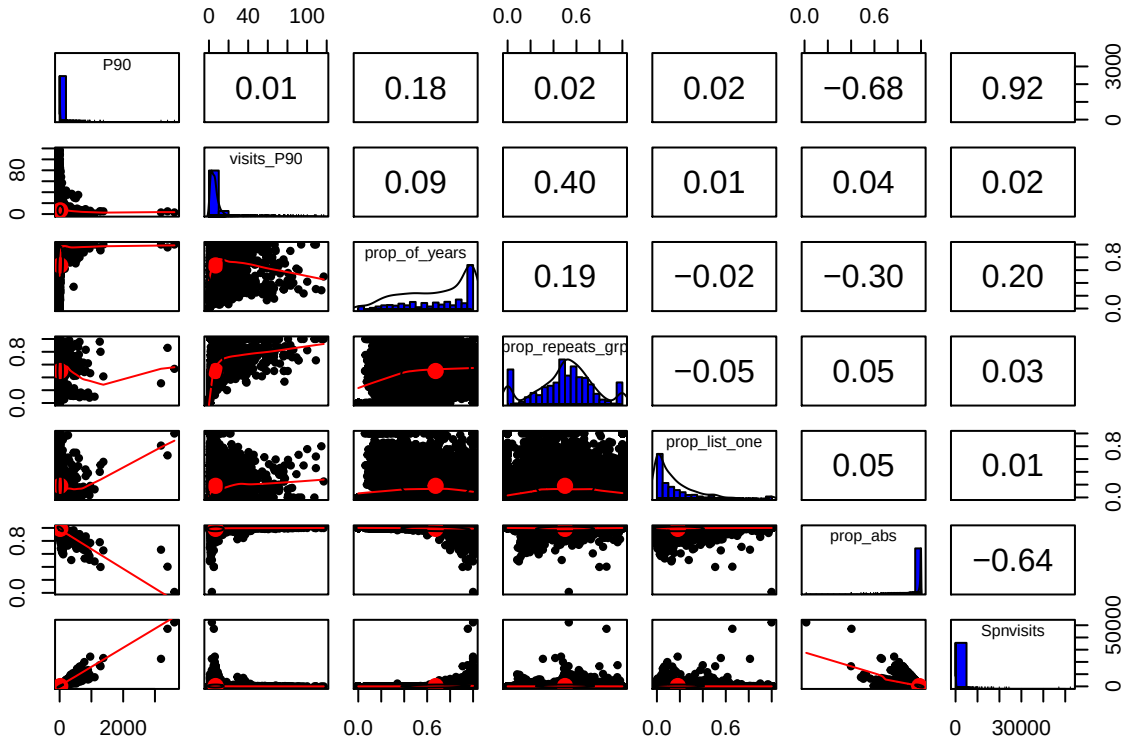


Correlation between variables

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('P90','visits_P90','prop_of_years',
                        'prop_repeats_grp','prop_list_one',
                        'prop_abs','Spnvisits')]
pairs.panels(td_sub, hist.col = 'blue',smooth = TRUE)
```



Input data caveat

In order to be included in the dataset, one key 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.

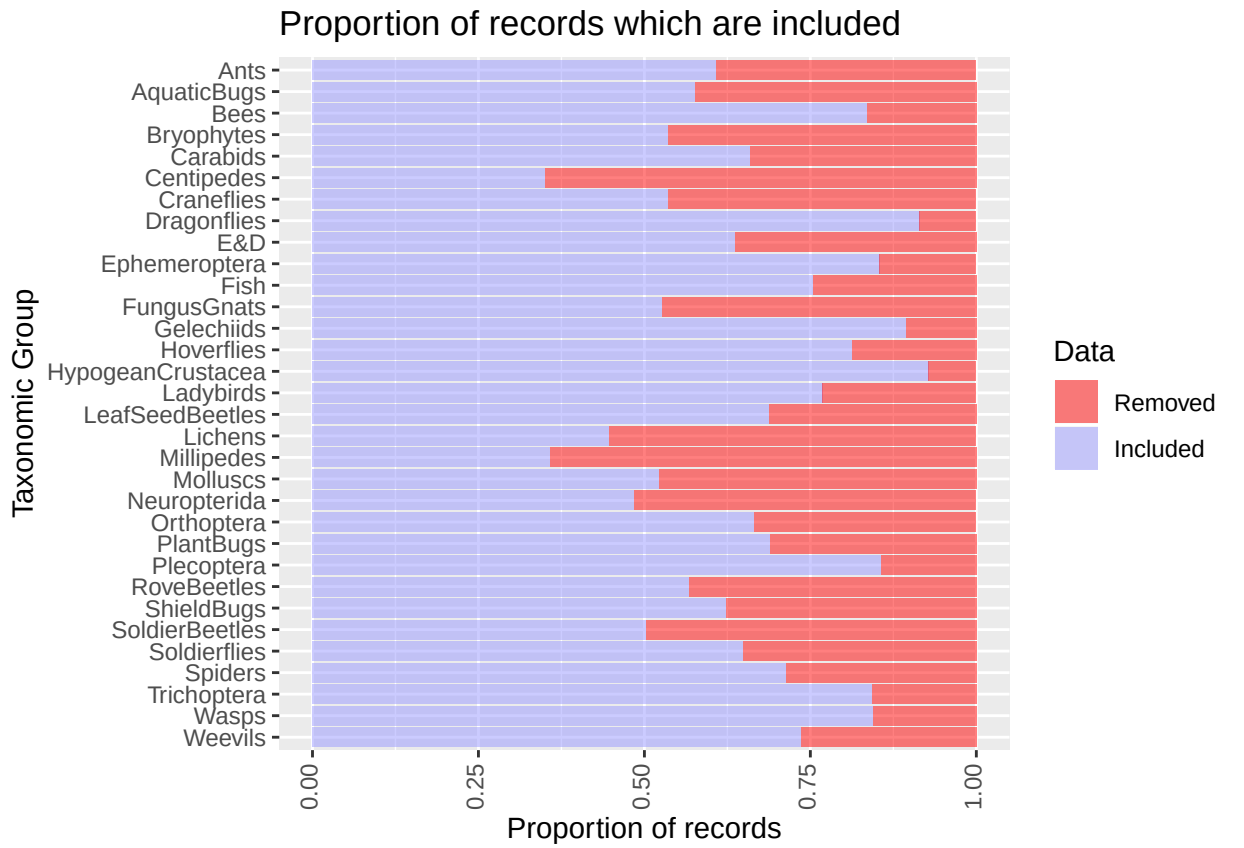
This step was carried out as it was carried out on the data prior to running the models. Failing to do this step would make the results invalid.

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'))

# Plot the results
stack_records <- function(df,colours,ylabel,title){
  ggplot(data=df,
         aes(fill=variable,y=value,x=taxa)) +
    geom_bar(stat='identity',alpha = .5) +
    theme(axis.text.x = element_text(angle=90, hjust=1, vjust = 0.3)) +
    labs(fill = 'Data',x='Taxonomic Group',y=ylabel) +
    scale_fill_manual(values = colours) +
    ggtitle(title) +
    scale_x_discrete(limits = rev(levels(df$taxa))) +
    coord_flip()
}

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



Classification

Equally weighted decision tree

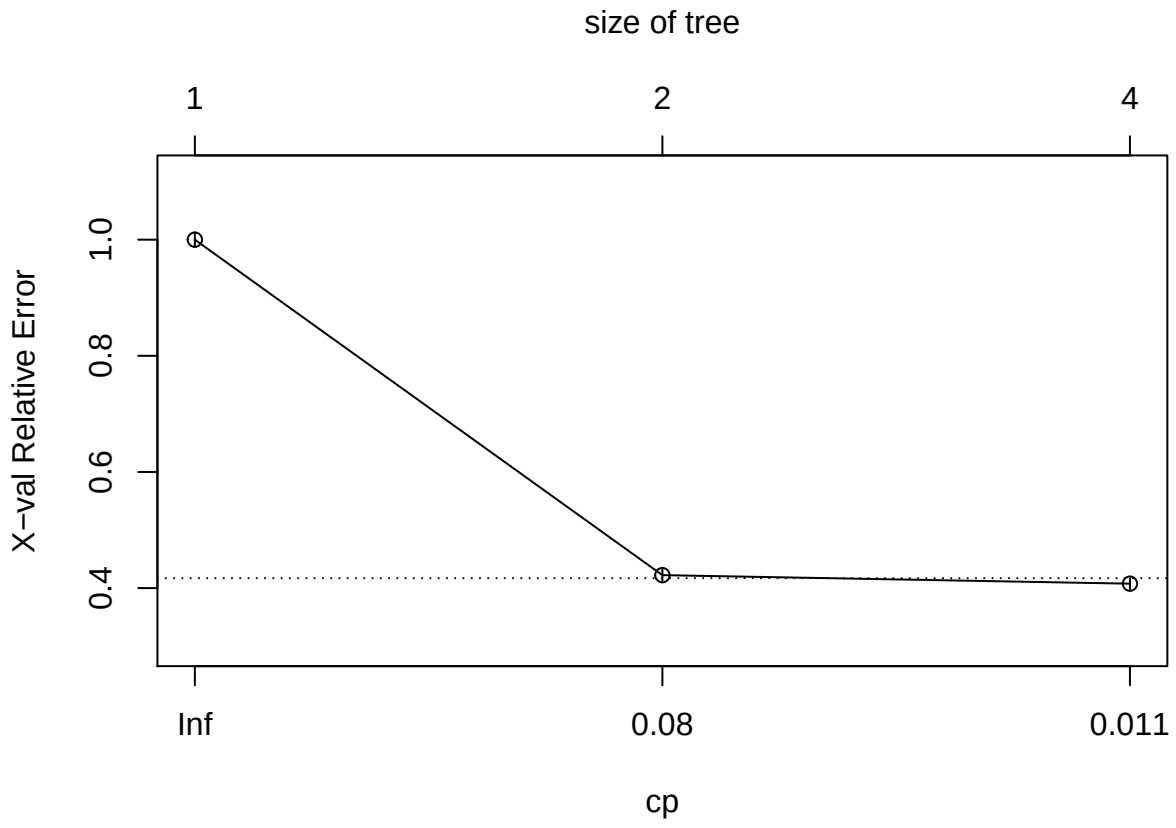
Using the split in the models between 'good' and 'bad', you can create a classification tree that attempts to use the variables we have extracted to partition the data to the best of its ability using a decision tree.

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

# Perform the fit.  The loss parameter in the rpart function determines the
# weighting:
#   By increasing 'goodweight', this increases the importance of correctly
#   identifying good models and not assigning bad models to the good category.
#   However, this is at the expense of assigning many good models to the bad
#   category.
#   By increasing 'badweight', the inverse is true
badweight <- 1
goodweight <- 1

fit <- rpart(bad ~ median + P90 +
               visits_median + visits_P70 + visits_P80 + visits_P90 +
               prop_of_years + prop_repeats_grp + prop_list_one +
```

```
              prop_abs,
       method = "class",
       data = trendsData,
       parms = list(loss = matrix(c(0,goodweight,badweight,0),ncol = 2)))

printcp(fit) # display the results
```

```
##
## Classification tree:
## rpart(formula = bad ~ median + P90 + visits_median + visits_P70 +
##     visits_P80 + visits_P90 + prop_of_years + prop_repeats_grp +
##     prop_list_one + prop_abs, data = trendsData, method = "class",
##     parms = list(loss = matrix(c(0, goodweight, badweight, 0),
##         ncol = 2)))
##
## Variables actually used in tree construction:
## [1] P90      prop_abs
##
## Root node error: 3805/9323 = 0.40813
##
## n= 9323
##
##         CP nsplit rel error  xerror      xstd
## 1 0.577924      0   1.00000 1.00000 0.0124720
## 2 0.011038      1   0.42208 0.42234 0.0095845
## 3 0.010000      3   0.40000 0.40762 0.0094502
```

```
plotcp(fit) # visualize cross-validation results
```



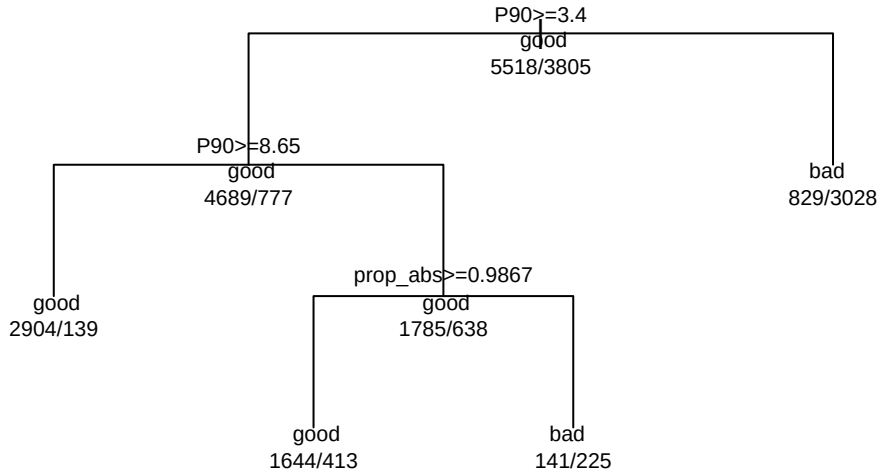

```
summary(fit) # detailed summary of splits
```

```
## Call:
## rpart(formula = bad ~ median + P90 + visits_median + visits_P70 +
##     visits_P80 + visits_P90 + prop_of_years + prop_repeats_grp +
##     prop_list_one + prop_abs, data = trendsData, method = "class",
##     parms = list(loss = matrix(c(0, goodweight, badweight, 0),
##         ncol = 2)))
##   n= 9323
##
##           CP nsplit rel error    xerror        xstd
## 1 0.57792378      0 1.0000000 1.0000000 0.012471988
## 2 0.01103811      1 0.4220762 0.4223390 0.009584546
## 3 0.01000000      3 0.4000000 0.4076216 0.009450174
##
## Variable importance
##           P90        median      prop_abs prop_of_years    visits_P90
##            29            21            18            14             9
## prop_list_one
##             9
##
## Node number 1: 9323 observations,    complexity param=0.5779238
##   predicted class= good  expected loss=0.4081304  P(node) =1
##     class counts:  5518  3805
##    probabilities: 0.592 0.408
##   left son=2 (5466 obs) right son=3 (3857 obs)
```

```
##   Primary splits:
##       P90           < 3.4          to the right, improve=1869.3910, (0 missing)
##       median        < 1.75         to the right, improve=1511.1460, (0 missing)
##       prop_abs      < 0.9985047    to the left,  improve=1077.0120, (0 missing)
##       prop_list_one < 0.0003182686 to the right, improve= 858.3649, (0 missing)
##       prop_of_years < 0.5053191    to the right, improve= 724.7123, (0 missing)
##   Surrogate splits:
##       median        < 1.75         to the right, agree=0.887, adj=0.728, (0 split)
##       prop_abs      < 0.9980539    to the left,  agree=0.834, adj=0.599, (0 split)
##       prop_of_years < 0.5108747    to the right, agree=0.781, adj=0.472, (0 split)
##       visits_P90    < 2.9          to the right, agree=0.729, adj=0.345, (0 split)
##       prop_list_one < 0.0003182686 to the right, agree=0.723, adj=0.331, (0 split)
##
## Node number 2: 5466 observations,    complexity param=0.01103811
##   predicted class= good  expected loss=0.1421515  P(node) =0.586292
##     class counts:  4689   777
##    probabilities: 0.858 0.142
##   left son=4 (3043 obs) right son=5 (2423 obs)
##   Primary splits:
##       P90           < 8.65         to the right, improve=127.77870, (0 missing)
##       prop_list_one < 0.0003182686 to the right, improve=100.01500, (0 missing)
##       median        < 2.75         to the right, improve= 97.94465, (0 missing)
##       prop_of_years < 0.7527778    to the right, improve= 64.01049, (0 missing)
##       prop_abs      < 0.9993402    to the left,  improve= 21.58277, (0 missing)
##   Surrogate splits:
##       median        < 3.75         to the right, agree=0.887, adj=0.746, (0 split)
##       prop_of_years < 0.8352713    to the right, agree=0.798, adj=0.544, (0 split)
##       prop_abs      < 0.9921345    to the left,  agree=0.758, adj=0.454, (0 split)
##       prop_list_one < 0.0003182686 to the right, agree=0.611, adj=0.123, (0 split)
##       visits_P90    < 4.8          to the right, agree=0.578, adj=0.048, (0 split)
##
## Node number 3: 3857 observations
##   predicted class=bad    expected loss=0.2149339  P(node) =0.413708
##     class counts:   829  3028
##    probabilities: 0.215 0.785
##
## Node number 4: 3043 observations
##   predicted class= good  expected loss=0.04567861  P(node) =0.3263971
##     class counts:  2904   139
##    probabilities: 0.954 0.046
##
## Node number 5: 2423 observations,    complexity param=0.01103811
##   predicted class= good  expected loss=0.2633099  P(node) =0.2598949
##     class counts:  1785   638
##    probabilities: 0.737 0.263
##   left son=10 (2057 obs) right son=11 (366 obs)
##   Primary splits:
##       prop_abs      < 0.9866517    to the right, improve=106.498300, (0 missing)
##       prop_list_one < 0.02390476   to the right, improve= 55.406150, (0 missing)
##       P90           < 4.85         to the right, improve= 23.653900, (0 missing)
##       prop_of_years < 0.252907     to the right, improve=  8.075591, (0 missing)
##       median        < 2.75         to the right, improve=  6.731192, (0 missing)
##
## Node number 10: 2057 observations
```

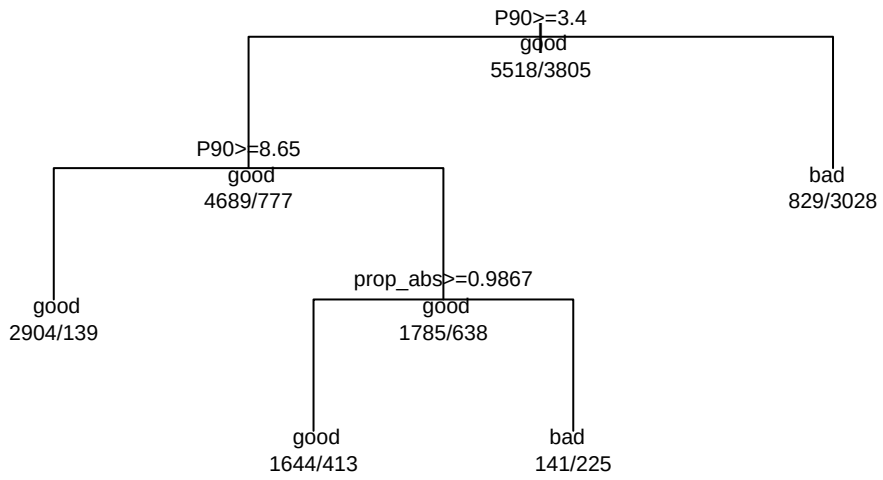
```
##   predicted class= good  expected loss=0.2007778  P(node) =0.2206371
##     class counts:  1644   413
##    probabilities: 0.799 0.201
##
## Node number 11: 366 observations
##   predicted class=bad    expected loss=0.3852459  P(node) =0.03925775
##     class counts:   141   225
##    probabilities: 0.385 0.615
```

```
# plot tree
plot(fit, uniform=TRUE,
     main = "'Rules of thumb' for species occupancy modelling",
     margin = .1)
text(fit, use.n = TRUE,
     all = TRUE,
     cex = .7)
```




```
# prune the tree
pfit <- prune(fit, cp = fit$cptable[which.min(fit$cptable[,"xerror"]),"CP"])

# plot the pruned tree
plot(pfit, uniform=TRUE,
    main="Pruned 'Rules of thumb' for species occupancy modelling",
    margin = .1)
text(pfit, use.n = TRUE, all = TRUE, cex = .7)
```

Pruned 'Rules of thumb' for species occupancy modelling



The decision tree with equal weighting to good and bad results shows a recall (chance of correct prediction) of 82% for good models and 85% for bad models, when the good/bad classifications are equally weighted. The precision (chance of classification being correct) for good models is 89% and 77% for bad models.

To explain this decision tree in words:

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 caveat):

- P90>=3.4: The 90th percentile of number of visits per year *in which there was at least one visit* must be greater than 3.4 or the model will definitely be classified as bad
  - For example, a species with 11 records, for which the second largest record was 4, would meet this target as the P90 would be 4.
  - A species with 2 years with records, with 1 record in one year and 4 in the other would have a P90 of 3.7, so would meet this target.
  - A species with 1 year with 4 records would have a P90 of 4, so would also meet this requirement.

If the data meets this criterion, it still must meet one of these two criteria to be classified as good:

- P90>=8.65 OR
- prop_abs>=0.9867. Prop_abs is the proportion of records for the taxonomic group which do not include the species of interest.
  - For example, if there were 2000 records within a group, and 40 records of the species of interest, prop_abs 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).

Examining the equally weighted classification

Lets assess the quality of this decision tree

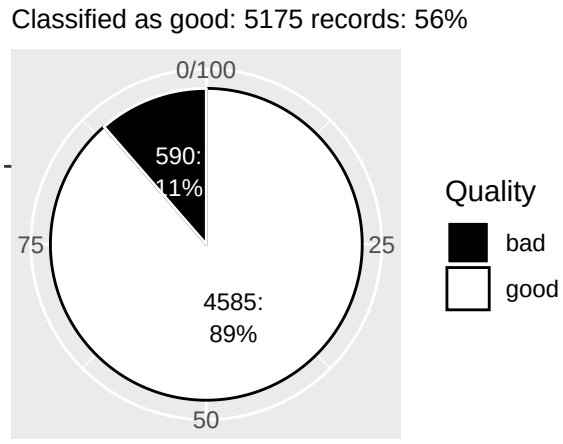
```
td <- trendsData

td$Node1 <- td$P90>=3.4
td$Node2 <- td$P90>=8.65
td$Node3 <- td$prop_abs>=0.9867

td$data_good <- rep('bad',length(td$bad))
td$data_good[(td$Node1)&(td$Node2|td$Node3)] <- 'good'

td$model_good <- rep('good',length(td$bad))
td$model_good[td$bad=='bad'] <- 'bad'

plot_pie <- function(subdata,numrow,title){
  numrecords <- nrow(subdata)
  numgood <- length(subdata$model_good[subdata$model_good=='good'])
  numbad <- numrecords - numgood
  tmpdf <- data.frame(
    Quality = c('good','bad'),
    value = c(100*numgood/numrecords,100*numbad/numrecords),
    num = c(numgood,numbad)
  )
  ggplot(tmpdf, aes(x='',y=value, fill=Quality, color=Quality)) +
    geom_bar(width=1,stat='identity') +
    coord_polar('y',start=0) +
    labs(title=paste0(title,': ',numrecords," records: ",
                      round(100*numrecords/numrow,0),"%")) +
    ylab(NULL) + xlab(NULL) +
    geom_text(aes(y = value/2 + c(0, cumsum(value)[-length(value)]),
                  label = paste0(num,':\n',round(value,0),'%'),
                  color=Quality),
              size=3,show.legend=FALSE) +
    scale_fill_manual(values=c('black','white')) +
    scale_color_manual(values=c('white','black')) +
    theme(plot.title = element_text(size=10))
}
```

Below are two pie charts showing how many of our 9323 species datasets have been classified as likely to produce 'good' or 'bad' models by this decision tree, and what percentage of those models were actually good (above the average precision threshold) or bad (below the threshold).

Classifying the data as likely to produce a good model:

P90>=3.4 & (P90>=8.65 OR prop_abs>=0.9867)

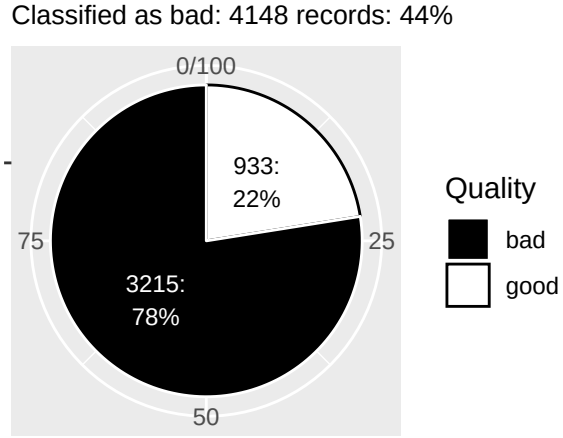
```
plot_pie(td[(td$data_good)=='good',],
         nrow(td),
         "Classified as good")
```



Classifying the data as likely to produce a 'bad' model:

P90<3.4 OR (P90>=3.4 & P90<8.65 & prop_abs<0.9867)

```
plot_pie(td[(td$data_good)=='bad',],
         nrow(td),
         "Classified as bad")
```



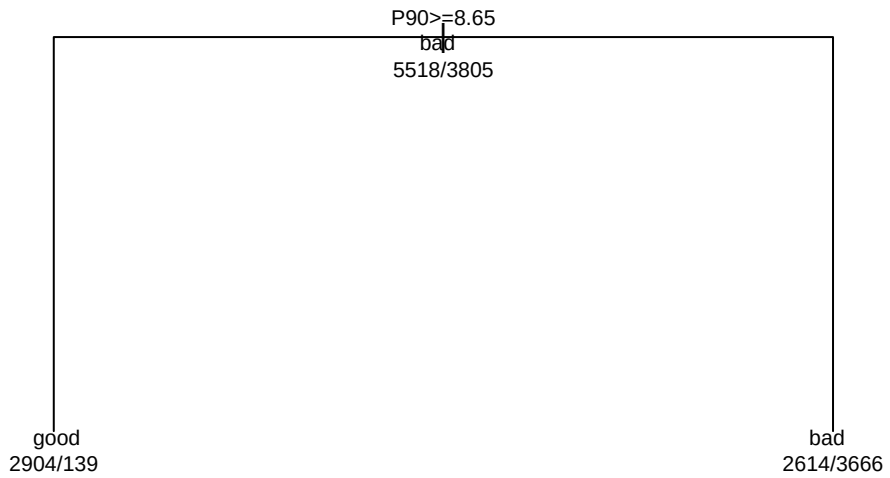
Non-equally weighted classifications

The decision tree is computed again twice more below. In the models below, the penalty of misclassifying good models as bad, or misclassifying bad models as good, is varied. With a 10:1 good:bad ratio, the decision tree can be considered an aspirational target. A 1:10 good:bad ration would be a bare minimum 'is it worth having a go at running a model' threshold.

10:1 Good:bad tree - Aspirational Target

```
# produce a decision tree for goodweight = 10
badweight <- 1
goodweight <- 10
fit_good <- rpart(bad ~ median + P90 +
                    visits_median + visits_P70 + visits_P80 + visits_P90 +
                    prop_of_years + prop_repeats_grp + prop_list_one +
```

```
                    prop_abs,
              method = "class",
              data = trendsData,
              parms = list(loss = matrix(c(0,goodweight,badweight,0),ncol = 2)))

# prune the tree
pfit_good <- prune(fit_good,
                   cp = fit_good$cptable[which.min(fit_good$cptable[,"xerror"]),
                                         "CP"])

# plot the pruned tree
plot(pfit_good, uniform=TRUE,
    main="'Rules of thumb' for species occupancy modelling, 10:1 good:bad",
    margin = .1)
text(pfit_good, use.n = TRUE, all = TRUE, cex = .7)
```



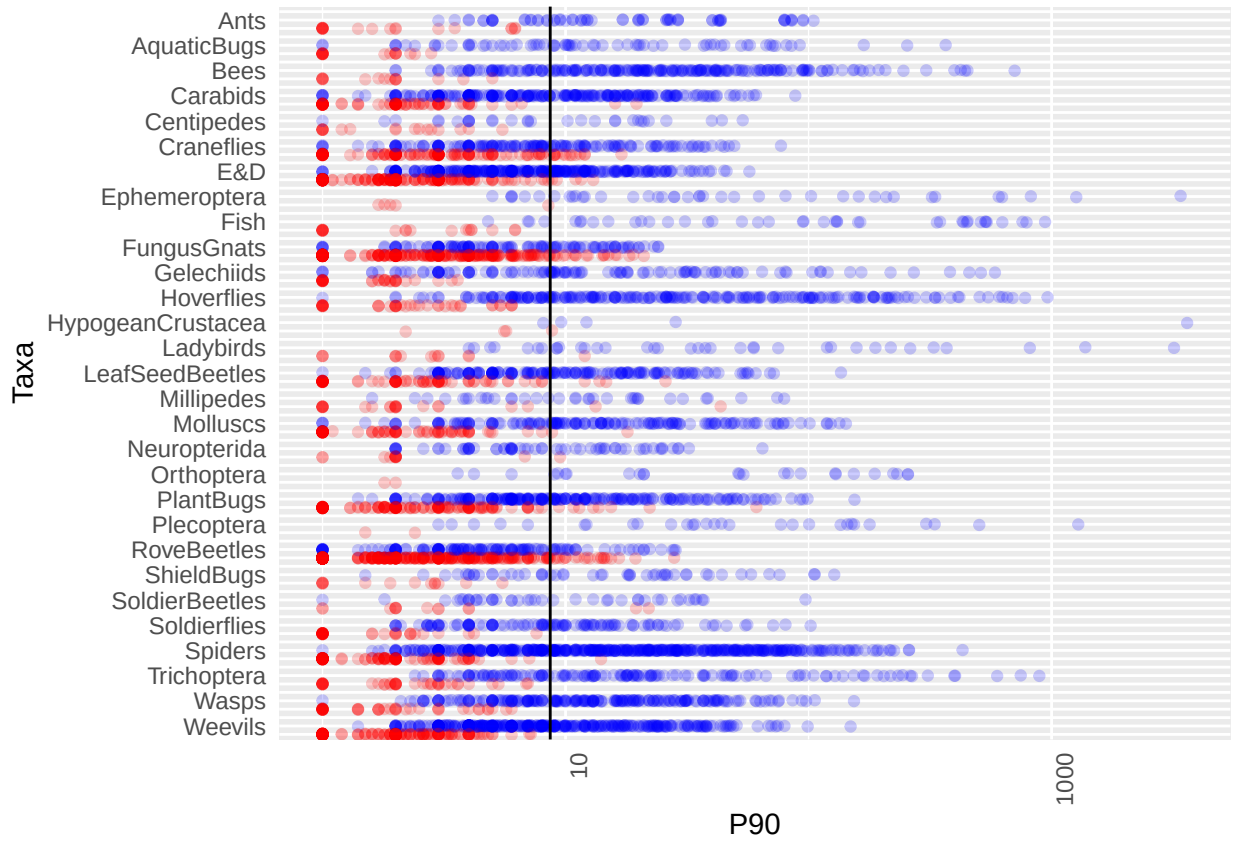
In the case of the 10:1 good:bad tree, the tree is very simple. If P90>=8.65, the data are likely to produce a good model.

The precision of this decision is 95%, while the precision of the bad classification is 58%. The recall for good models is 53% and 96% for bad models. This means only 53% of data that might produce a good model are deemed good enough. However, if data meet this threshold, they are very likely going to produce a good model.

To show how this split applies to all the data, a plot is shown below. On the x-axis are all the taxonomic groups, while the y-axis shows the P90 with the 8.65 cutoff shown as a horizontal bar. Red circles represent models which have been classified as bad, while blue are good.

As can be seen, this split does a good job of removing most bad models, with the exception of some models from taxonomic groups which in general do not have many records per species, such as FungusGnats and RoveBeetles.

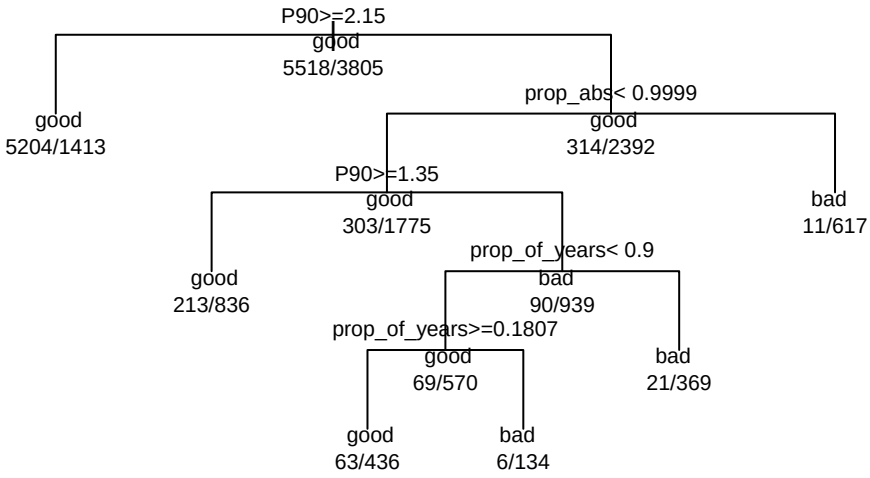
```
td_taxa <- trendsData[as.character(trendsData$Taxa)==
                      as.character(trendsData$Taxa_Root),]
td_taxa <- td_taxa[td_taxa$P90!=0,]

## A little bit of data manipulation is required to make a nice graph
td_taxa$colour <- rep('blue',length(td_taxa$bad))
td_taxa$colour[td_taxa$bad=='bad'] <- 'red'
td_taxa$cat <- paste0(td_taxa$Taxa_Root,' - ',td_taxa$bad)
td_taxa$alpha <- 0.2
taxa_list <- as.character(unique(td_taxa$Taxa))
taxa_list <- taxa_list[!is.na(taxa_list)]
tmp_bind <- td_taxa[1:length(taxa_list),]
tmp_bind$P90 <- 1
tmp_bind$cat <- paste0(sort(taxa_list),' ')
tmp_bind$alpha <- 0
td_taxa <- rbind(td_taxa,tmp_bind)
names <- sort(unique(as.character(td_taxa$Taxa_Root)))
names_long <- c()
for(i in 1:length(names)){
  names_long[i*3-2] <- ''
  names_long[i*3-1] <- names[i]
  names_long[i*3] <- ''
}

## Data wrangling complete, plot it up
ggplot(td_taxa, aes(Taxa, P90), main = 'Title') +
  geom_jitter(aes(reorder(cat,desc(cat)), P90), data = td_taxa,
              colour = td_taxa$colour, alpha = td_taxa$alpha,
              position = position_jitter(width = 0.08)) +
  scale_y_log10() +
  geom_hline(yintercept=8.65) +
  theme(axis.text.x = element_text(angle=90, hjust=1, vjust = 1.25),
        axis.ticks.x = element_blank(), axis.ticks.y = element_blank()) +
  scale_x_discrete(labels=rev(names_long)) +
  coord_flip()
```



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

```
# produce a decision tree for badweight = 10
goodweight <- 1
badweight <- 10
fit_bad <- rpart(bad ~ median + P90 +
                    visits_median + visits_P70 + visits_P80 + visits_P90 +
                    prop_of_years + prop_repeats_grp + prop_list_one +
                    prop_abs,
             method = "class",
             data = trendsData,
             parms = list(loss = matrix(c(0,goodweight,badweight,0),ncol = 2)))

# prune the tree
pfit_bad <- prune(fit_bad,
                  cp = fit_bad$cptable[which.min(fit_bad$cptable[,"xerror"]),
                                       "CP"])

# plot the pruned tree
plot(pfit_bad, uniform=TRUE,
    main="'Rules of thumb' for species occupancy modelling, 1:10 good:bad",
    margin = .1)
text(pfit_bad, use.n = TRUE, all = TRUE, cex = .7)
```

'Rules of thumb' for species occupancy modelling, 1:10 good:bad



The 1:10 good:bad tree is more complex, and requires some disentangling.

If P90>=2.15 the model will be classified a good. This is a sensible low threshold and what is expected. However, beyond this point the decision tree looks more surprising.

The decision tree has been run without number of records as a variable. This is because it is desirable to have these decision trees time independent, and number of records was deemed to be related to the number of years over which data has been collected, rather than quality of the data.

However, a number of the variables in the decision tree can act as proxies for number of records. Prop_abs for instance will be very high if number of records is very low. Prop_abs < 0.9999 is therefore a proxy for number of records $>$ some very low number.

Similarly, prop_of_years is acting as a number of records proxy. To be classified as good, prop_of_years must be < 0.9 and $>= 0.1807$. This seems initially surprising, but makes sense on reflection. If prop_of_years is < 0.9, there must be more than one record, as records must be spread across more than one year, otherwise prop_of_years would be 1.

However, if prop_of_years < 0.1807, the records will be too sparsely spread out to be useful.

The decision tree produces a precision and recall for bad models of 97% and 29% respectively, and 67% and 99% for good models. If Spnvisits (number of records) is allowed back into the decision tree, the only question that needs to be asked is: are there at least 2 records? This results in a precision and recall for bad models of 97% and 21%, and 67% and 99.5% for good models, a similar classification rate for a much simpler decision tree.

How many species can we model in each habitat/region?

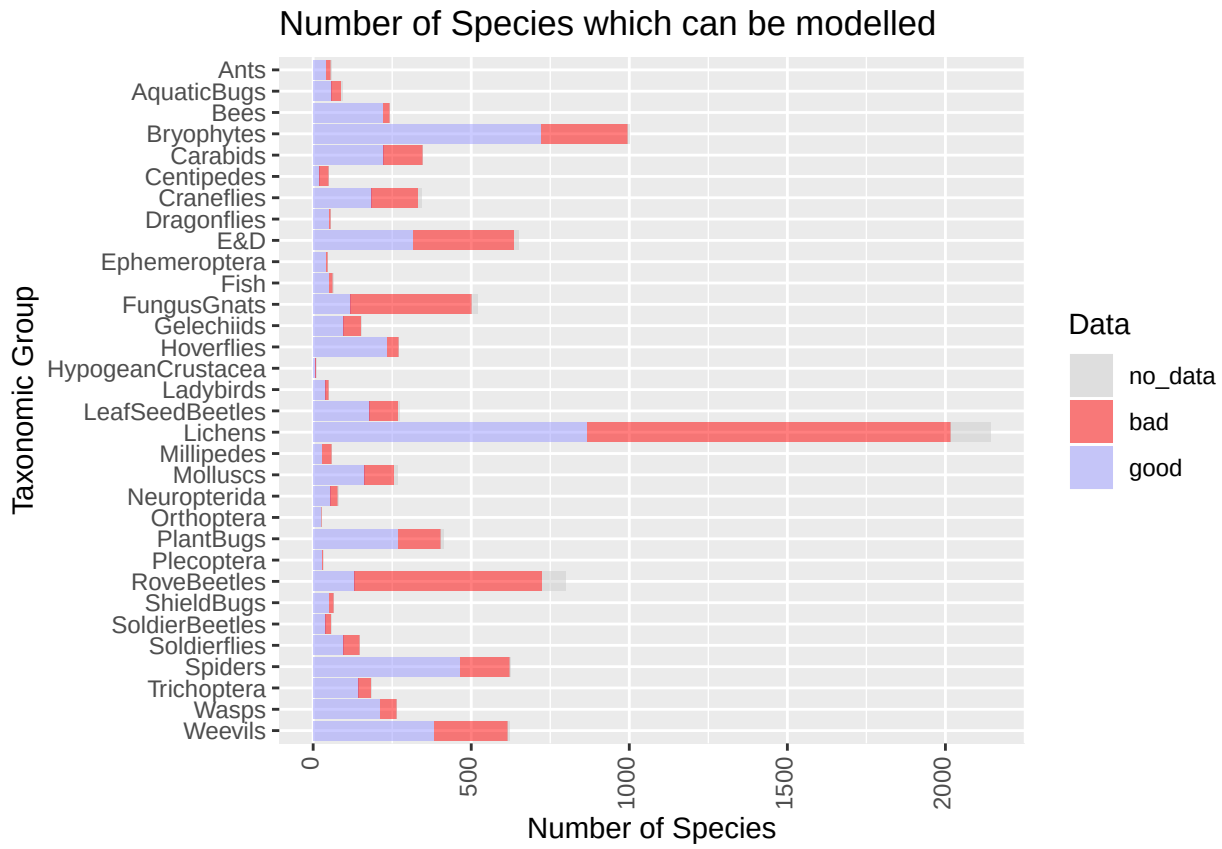
Based on the work above, we can calculate how many speices we can model for taxa across multiple habitat and regional areas.

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.

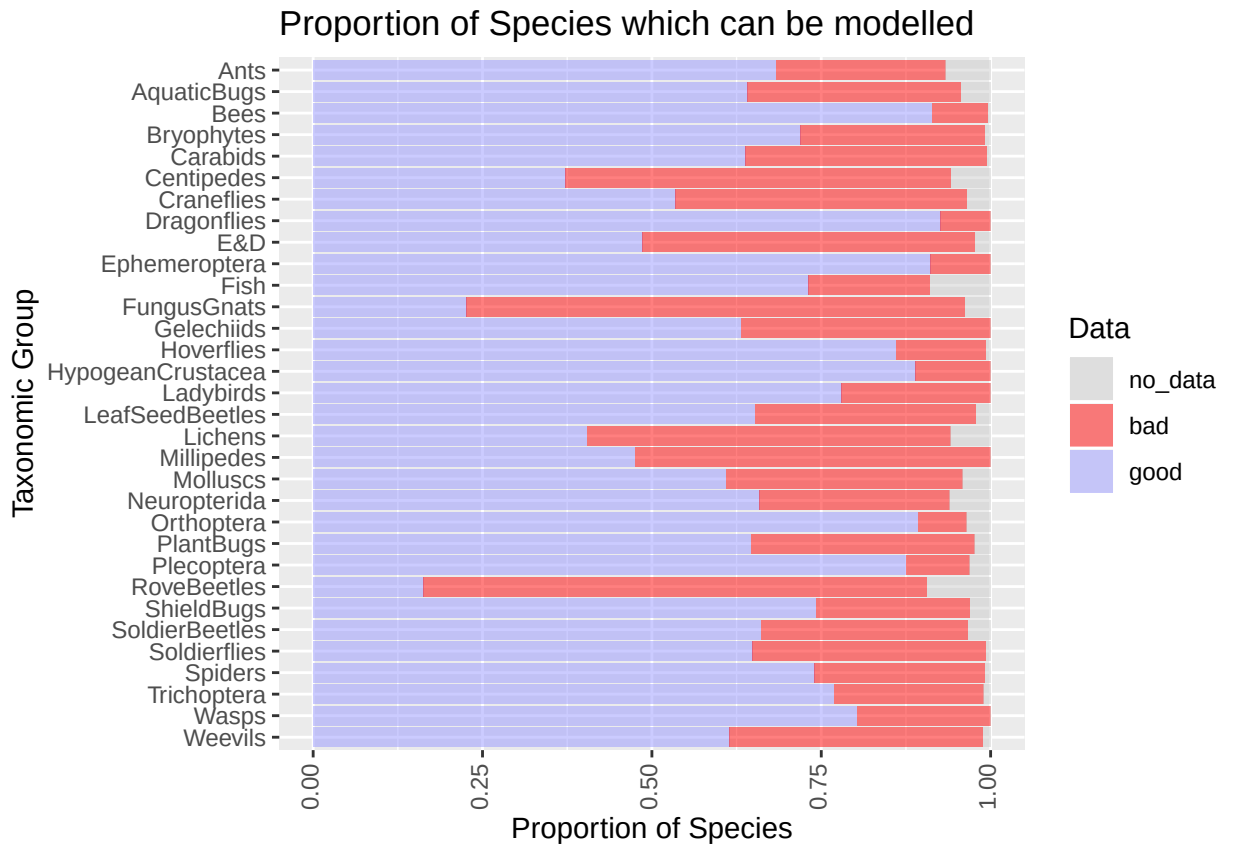
```
# Extract raw data metrics, as we want to look at all species, not just those that we had matching post
RM <- read.csv('Results/metrics/ALL_rawMetrics.csv')
# Remove all last 10 yr data
RM$Taxa <- as.character(RM$Taxa)
RM <- RM[RM$Taxa==as.character(RM$Taxa_Root),]

# Calculate which datasets are likely to produce good or bad models
calc_bad <- function(df){
  df$Node1 <- df$P90>=3.4
  df$Node2 <- df$P90>=8.65
  df$Node3 <- df$prop_abs>=0.9867
  df$data_good <- df$data_aspire <- rep('bad',length(df$species))
  df$data_good[(df$Node1)&(df$Node2|df$Node3)] <- 'good'
  df$data_aspire[df$Node2] <- 'good'
  df
}
RM <- calc_bad(RM)

# Function to extract data from data frame
#  df = a trendsData data frame
#  prop = is the graph to show proportion of species (vs absolute numbers)
#  hab = habitat of interest
#  reg = region of interest, either full name or NUTS code
#  zeroes = are species with no records (Spnvisits==0) to be included?
#  aspire = is the cutoff the aspirational target (10:1 good:bad)?
#  melt = is the long table for plotting required?
num_spec <- function(df,hab=NULL,reg=NULL,prop=FALSE,melt=TRUE,
                     zeroes=TRUE,aspire=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(hab)){
    df <- df[as.character(df$habitat)==hab,]
  }
  if(!is.null(reg)){
    if(grepl('UK[A-Z]',reg)){
      df <- df[as.character(df$code)==reg,]
    } else {
      df <- df[as.character(df$region)==reg,]
```

```
    }
  }
  taxa_count <- NULL
  for(taxa in sort(unique(taxa_num$taxa))){
    if(aspire){
      data_good <- length(df$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(prop){
    taxa_melt <- taxa_prop_melt
  }
  if(melt){
    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'
```

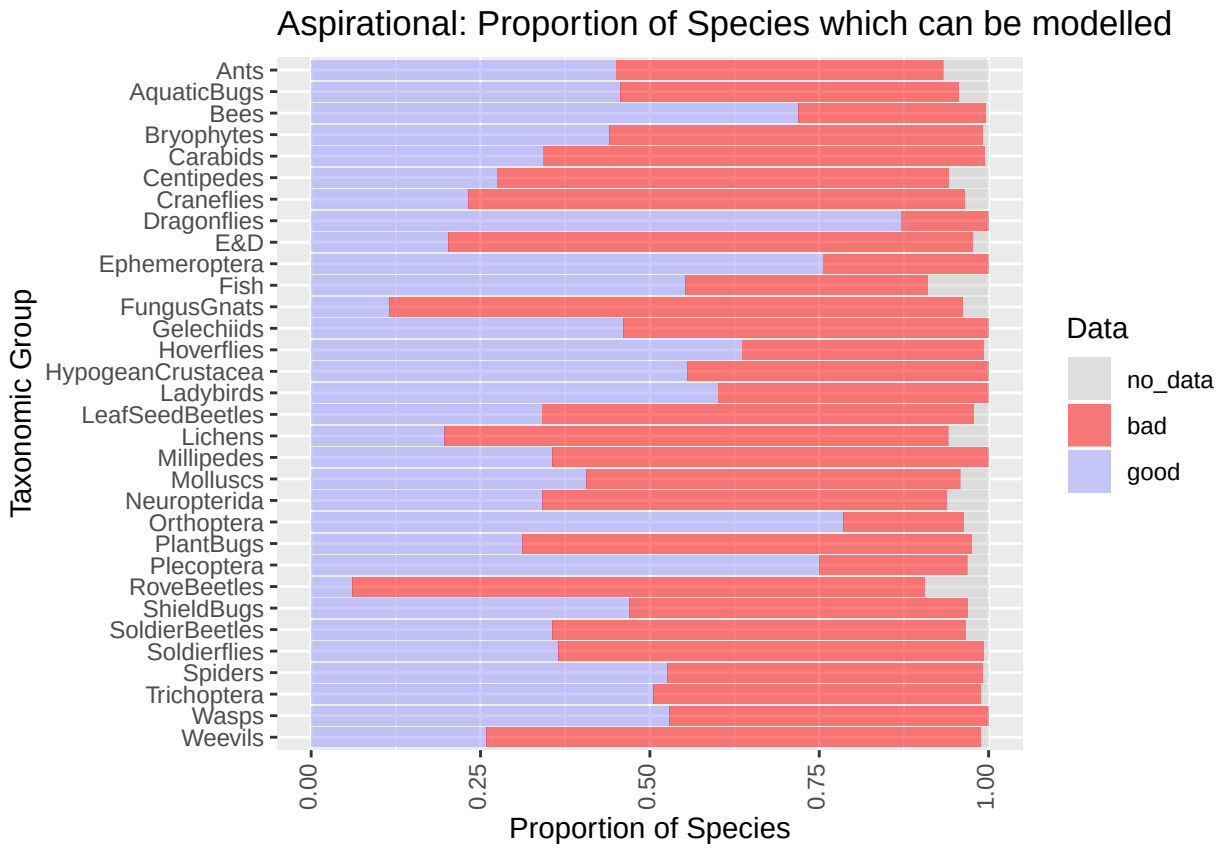
```
  } else {
    ylabel <- 'Number of Species'
  }
  if(!is.null(prefix)){
    title <- paste0(prefix,': ',ylabel,' which can be modelled')
  } else {
    title <- paste(ylabel,'which can be modelled')
  }
  stack_records(df,colours,ylabel,title)
}

# Plot the graphs
num_spec(RM) %>% stack_taxa()
```




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




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



By habitat

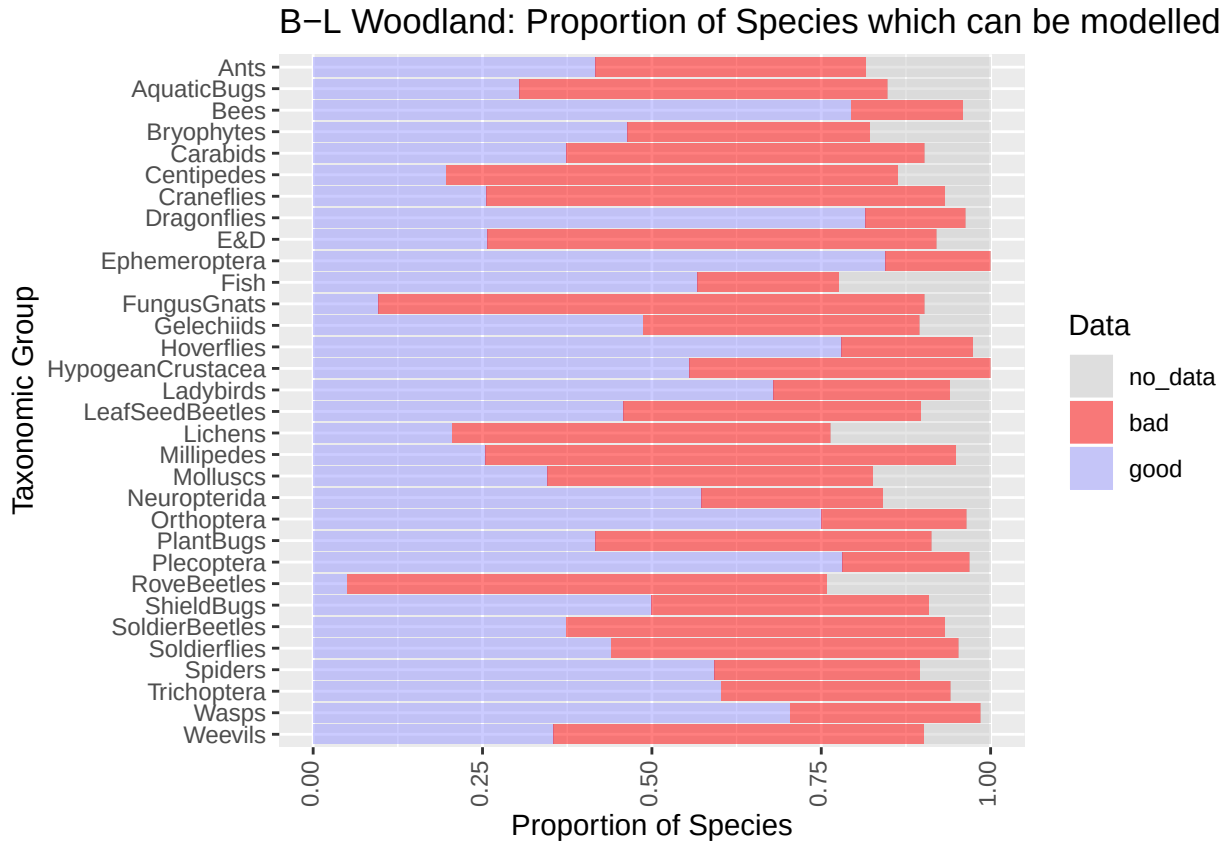
The raw metrics for each habitat have been calculated from the raw data. Using these data, the proportion of species which can be modelled for each habitat can be assessed.

```
# Read in the habitat raw metrics
RM_hab <-
  read.csv(file = file.path('Habitat/HabMetrics/ALL_habMetrics.csv'))
# Display the list of possible habitats
hab_list <- read.csv('Habitat/Habitat_List.csv')
column_spec(kable(hab_list, format = 'latex',
                  caption = 'List of available habitats'),
            2, width = "8cm")
```

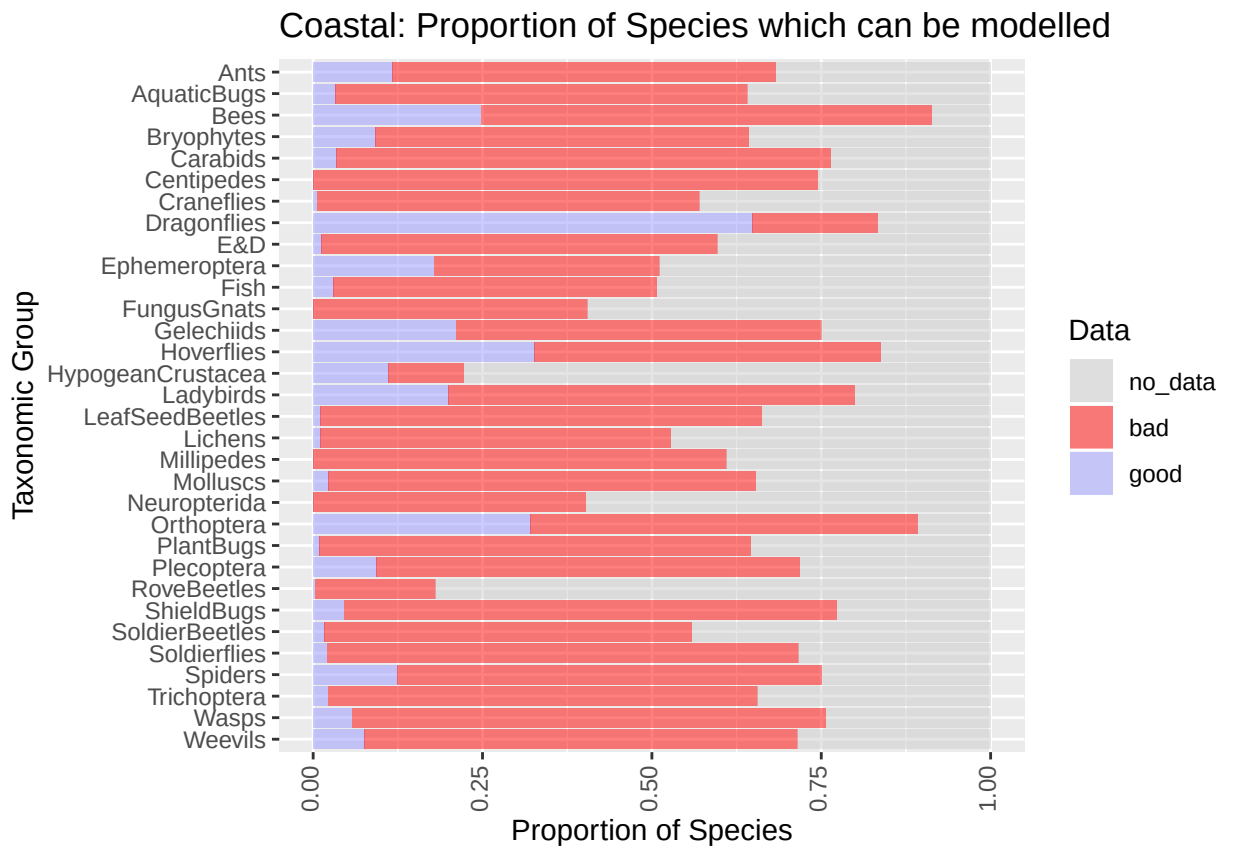
```
# Calculate which datasets are likely to produce good or bad models
RM_hab <- calc_bad(RM_hab)
RM_hab$habitat <- as.character(RM_hab$habitat)

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

Table 1: 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




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



By region

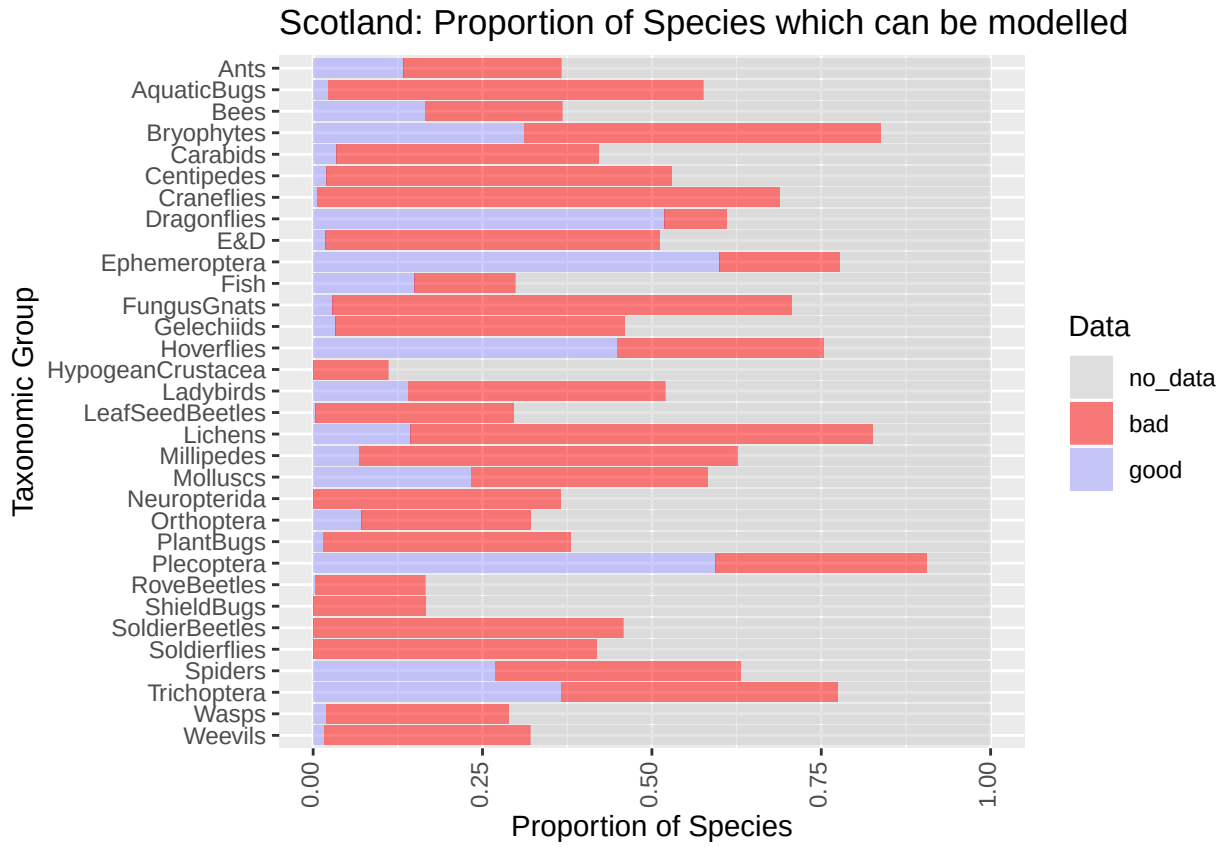
The data can also be split by region. For this we read in the region data file and query the data by region.

```
# Read in the regional raw metrics
RM_reg <-
  read.csv(file =  file.path('Region/RegMetrics/ALL_regMetrics.csv'))
# Display the list of regions
region_lookup <- read.csv(file = file.path('Region/NUTS_lookup.csv'))
kable(region_lookup, caption = 'List of regions')
```

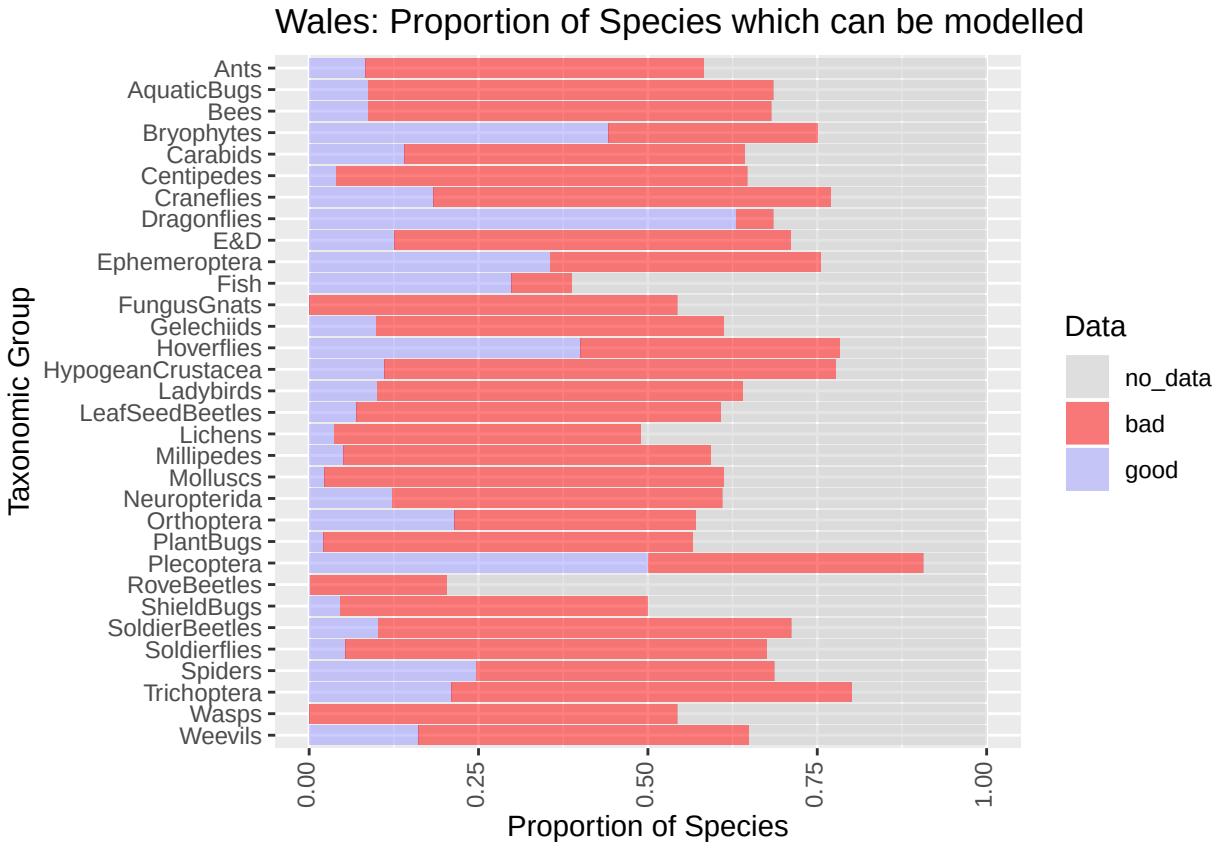
```
# Merge this lookup table with the raw data to allow it be queried by code
RM_reg <- merge(RM_reg,region_lookup)

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

# Plot up a sample graph for Scotland
num_spec(RM_reg,reg='Scotland',prop=TRUE) %>% stack_taxa(prefix='Scotland')
```

Table 2: 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




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


Wales: Proportion of Species which can be modelled
Taxonomic Group
Ants
AquaticBugs
Bees
Bryophytes
Carabids
Centipedes
Craneflies
Dragonflies
E&D
Ephemeroptera
Fish
FungusGnats
Gelechiids
Hoverflies
HypogeanCrustacea
Ladybirds
LeafSeedBeetles
Lichens
Millipedes
Molluscs
Neuropterida
Orthoptera
PlantBugs
Plecoptera
RoveBeetles
ShieldBugs
SoldierBeetles
Soldierflies
Spiders
Trichoptera
Wasps
Weevils
0.00
0.25
0.50
0.75
1.00
Proportion of Species
Data
no_data
bad
good