```
##-----------------------------------------------------------------------------
# Script Name:        main.R
# Description:        r script to perform the exploratory data analysis,
#                     modelling for predicting inter-winter and intra-winter
#                     survival given capture history data
#
# Author:             Jacob Pryce
# Date:               2023-08-06
# Version:            1.0
# Dependencies: - data.table: Enhanced, efficient data manipulation
#               - stats4: Basic statistical functions, estimation
#               - dplyr: Grammar of data manipulation
#               - tidyr: Data cleaning, organisation
#               - stringr: Easy string manipulation
#               - ggplot2: Declarative graphics creation system
#               - jagsUI: Interface for JAGS Bayesian models
#               - xtable: Exporting tables to LaTeX
#
# Data Sources: - "CR.blackcap_FixRing.csv"
#               - "CR.chifchaf_FixRing.csv"
#               - "CR.robin_FixRing.csv"
#-----------------------------------------------------------------------------

library(data.table) # Enhanced, efficient data manipulation
library(stats4) # Basic statistical functions, estimation
library(dplyr) # Grammar of data manipulation
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:data.table':
##
##     between, first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

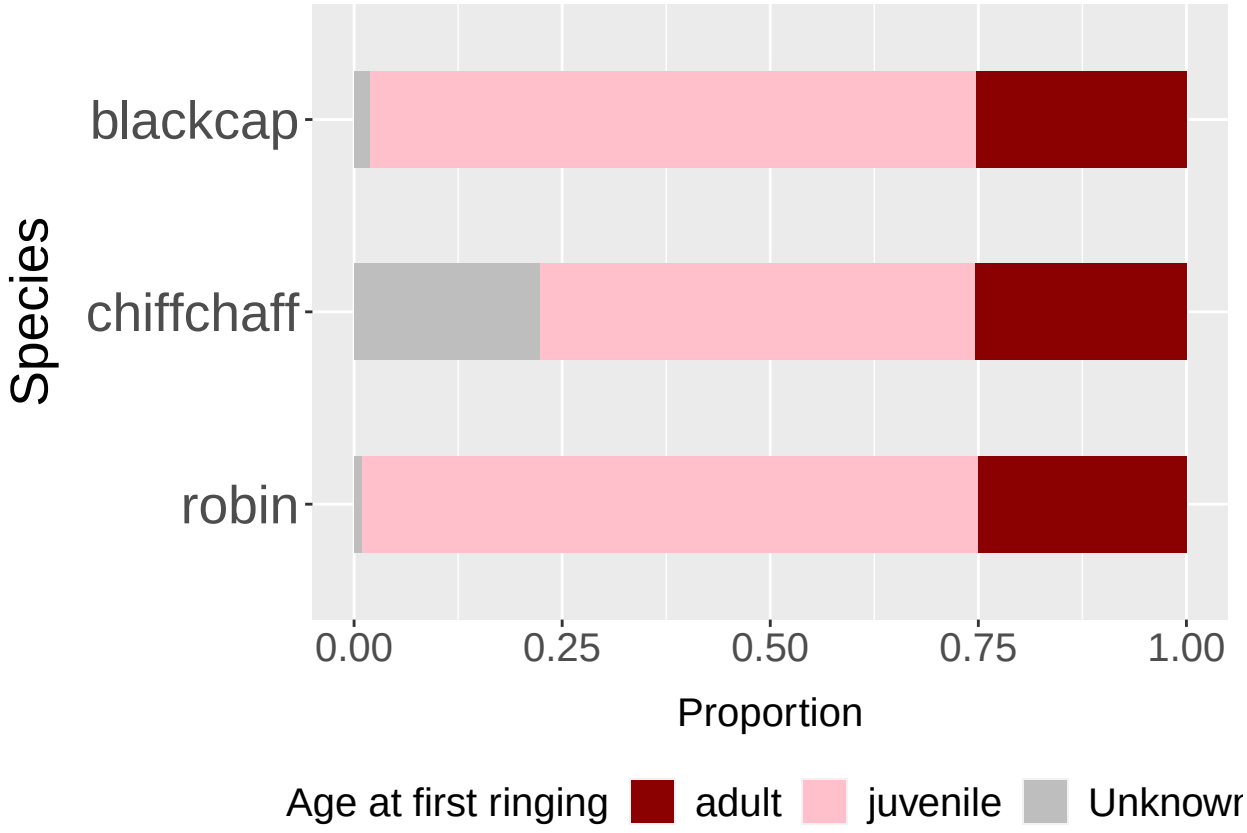
```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr) # Data cleaning, organisation
library(stringr) # Easy string manipulation
library(ggplot2) # Declarative graphics creation system
library(jagsUI)  # Interface for JAGS Bayesian models
library(xtable) # Exporting tables to LaTeX

# Check and create directory for images
if (!dir.exists("report/images")) {
  dir.create("report/images", recursive = TRUE)
}

# Check and create directory for model outputs
if (!dir.exists("report/model_outputs")) {
  dir.create("report/model_outputs", recursive = TRUE)
```

```
}


# Define file paths
file_paths <- c("CR.blackcap_FixRing.csv",
                "CR.chifchaf_FixRing.csv",
                "CR.robin_FixRing.csv")

# Check if files exist and read them in if they do
if(all(file.exists(file_paths))){
  blackcap_data_raw <- tryCatch({
    fread("CR.blackcap_FixRing.csv", header = TRUE)
  }, error = function(e) {
    message("Error reading CR.blackcap_FixRing.csv: ", e)
    return(NULL)
  })

  chiffchaff_data_raw <- tryCatch({
    fread("CR.chifchaf_FixRing.csv", header = TRUE)
  }, error = function(e) {
    message("Error reading CR.chifchaf_FixRing.csv: ", e)
    return(NULL)
  })

  robin_data_raw <- tryCatch({
    fread("CR.robin_FixRing.csv", header = TRUE)
  }, error = function(e) {
    message("Error reading CR.robin_FixRing.csv: ", e)
    return(NULL)
  })
} else {
  stop("One or more files do not exist. Please check the file paths.")
}


##############################
# DATA PREPARATION
##############################

clean_dataset <- function(df, species) {
  #' Clean a dataset by adding an ID column, removing unwanted columns, and
  #' filtering rows
  #'
  #' @param df Dataframe containing the raw data.
  #' @param species Character string indicating the species name.
  #' @return A cleaned dataframe with a unique ID for each row, unnecessary
  #'  columns removed, and rows with no capture information filtered out.
  #'
  #'
  if (!is.data.frame(df)) stop("df is not a dataframe.")
  if (!is.character(species) | length(species) != 1) {
    stop("species must be asingle character string.")
  }
```

```r
  # Add row number and species name columns
  df <- df %>% mutate(rn = row_number(), species = species) %>%
    # Create a unique ID for each row using species name and row number, then
    # remove the temporary rn column
    mutate(id = paste0(species, "_", rn)) %>% select(-rn)

  # Remove the first four months (as they represent only a partial winter)
  df <- df[, 5:(ncol(df))]

  # Filter out rows with no capture information, by ensuring the sum of values
  #  (excluding certain columns) is greater than 0
  df <- df %>% filter(rowSums(select(., -c(id, age_at_ringing, species))) > 0)

  # Return the cleaned dataset
  df
}

blackap_data_cleaned <- clean_dataset(blackcap_data_raw, 'blackcap')
chiffchaff_data_cleaned <- clean_dataset(chiffchaff_data_raw, 'chiffchaff')
robin_data_cleaned <- clean_dataset(robin_data_raw, 'robin')

birds <- rbind(blackap_data_cleaned, chiffchaff_data_cleaned,
               robin_data_cleaned)

################################
# EDA
################################
size <- 20
# custom theme for plotting purposes
custom_theme <- theme(
  axis.text.x = element_text(size = size),
  axis.text.y = element_text(size = size),
  axis.title.x = element_text(size = size, margin = margin(t = 10)),
  axis.title.y = element_text(size = size, margin = margin(r = 10)),
  plot.title = element_text(size = size, hjust = 0.5)
)

###### Age at first ringing proportions by species #####
# Order the levels of the 'species' variable
birds$species <- factor(birds$species, levels = c("robin", "chiffchaff",
                                                  "blackcap"))

ggplot(birds, aes(fill=age_at_ringing, x=species)) +
  geom_bar(position="fill", width=0.5) +
  coord_flip() +
  scale_fill_manual(values = c("Unknown" = "grey", "adult" = "dark red",
                               "juvenile" = "pink")) +
  labs(x = "Species", y = "Proportion", fill = "Age at first ringing") +
  custom_theme +
  theme(legend.text = element_text(size = size / (4/3)),
        legend.title = element_text(size = size / (4/3)),
        legend.position = "bottom",
        axis.text.x = element_text(size = size / (4/3)),
```

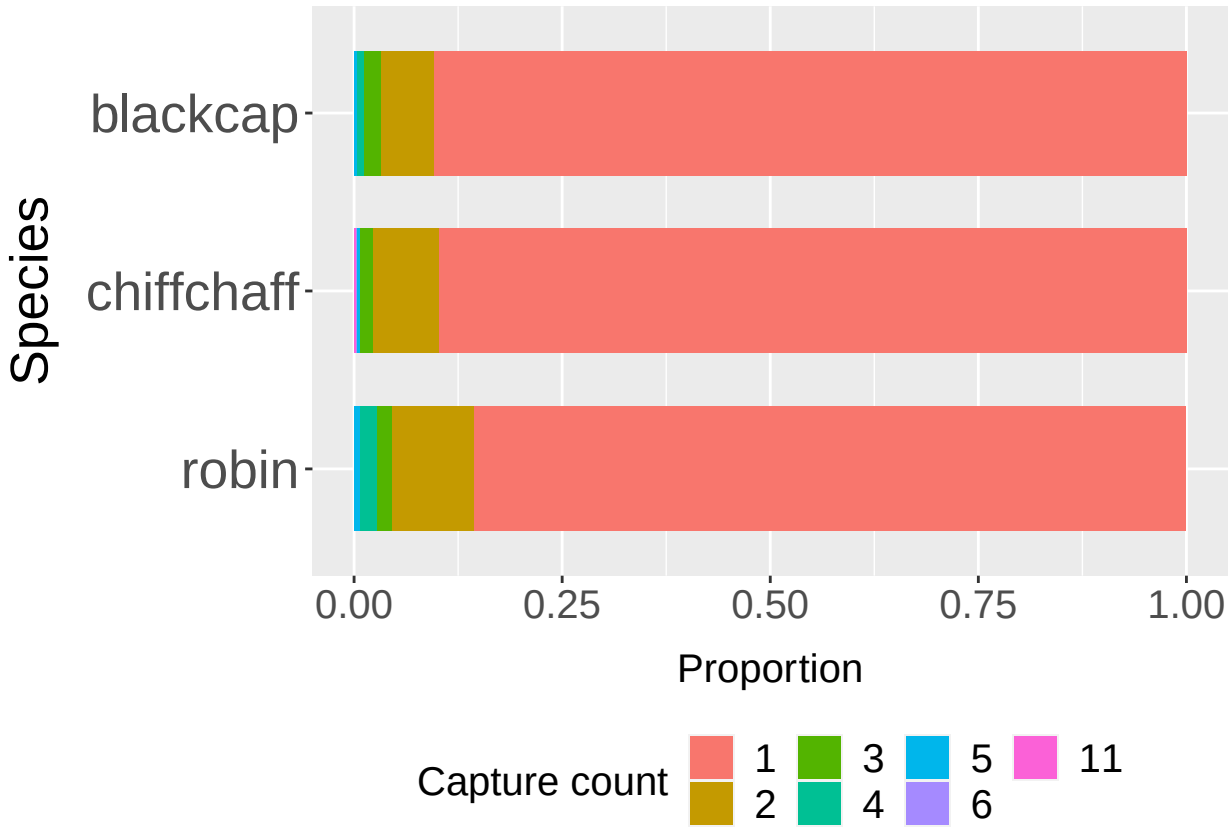
```
        axis.title.x = element_text(size = size / (4/3)))
```




```
ggsave("report/images/age_props.png")
```

```
## Saving 6.5 x 4.5 in image
```

```
#### Capture count proportions by species ####
birds$row_sum <- rowSums(select_if(birds, is.numeric))
# Convert the row_sum to a factor
birds$row_sum <- as.factor(birds$row_sum)
# Plot the data
ggplot(birds, aes(fill=row_sum, x=species)) +
  geom_bar(position="fill", width = 0.7) +
  scale_fill_discrete(name = "Capture count") +
  coord_flip() +
  labs(x = "Species", y = "Proportion") +
  custom_theme +
  theme(legend.text = element_text(size = size/ (4/3)),
        legend.title = element_text(size = size / (4/3)),
        legend.position = "bottom",
        axis.text.x = element_text(size = size / (4/3)),
        axis.title.x = element_text(size = size / (4/3)))
```




```
ggsave("report/images/capture_count_proportions.png")
```

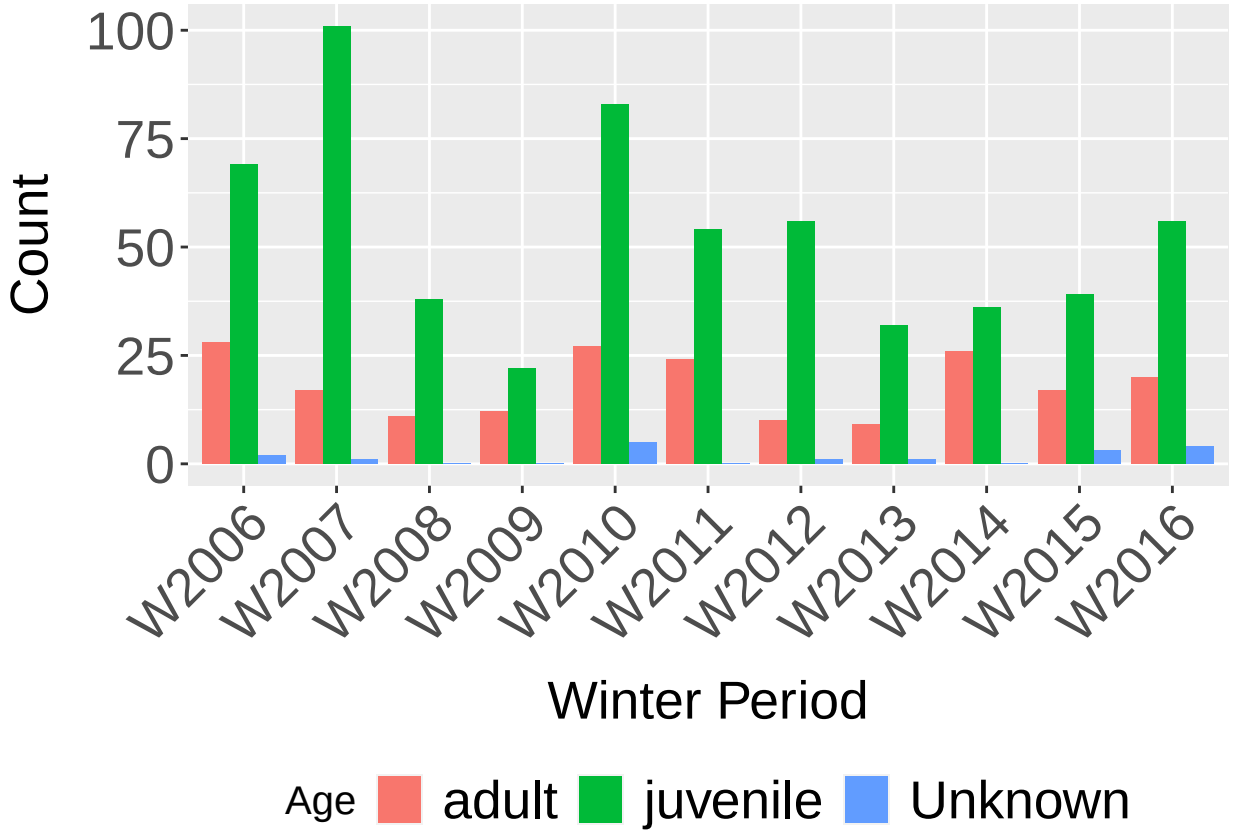
```
## Saving 6.5 x 4.5 in image
```

```
birds_long <- birds %>%
  # Pivot the dataset from wide to long, taking columns that start with "20"
  # as date information and storing the corresponding values in "count"
  pivot_longer(cols = starts_with("20"), names_to = "date",
               values_to = "count") %>%
  # Extract year and month from the "date" column as separate numerical variables
  mutate(
    year = as.numeric(substr(date, 1, 4)),
    month = as.numeric(substr(date, 5, 6)),
    # Determine the winter period based on the month, considering months from
    #  October as belonging to the next winter period
    winter_period = ifelse(month >= 10, year, year - 1)
  ) %>%
  # Transform month number into winter month names, starting from October as
  # month 1, November as month 2, etc.
  mutate(
    month_name = factor(ifelse(month >= 10, month - 9, month + 3),
                        levels = c(1:7),
                        labels = c("Oct", "Nov", "Dec",
                                   "Jan", "Feb", "Mar", "Apr"))
  )


###### Capture count by winter period ####
```

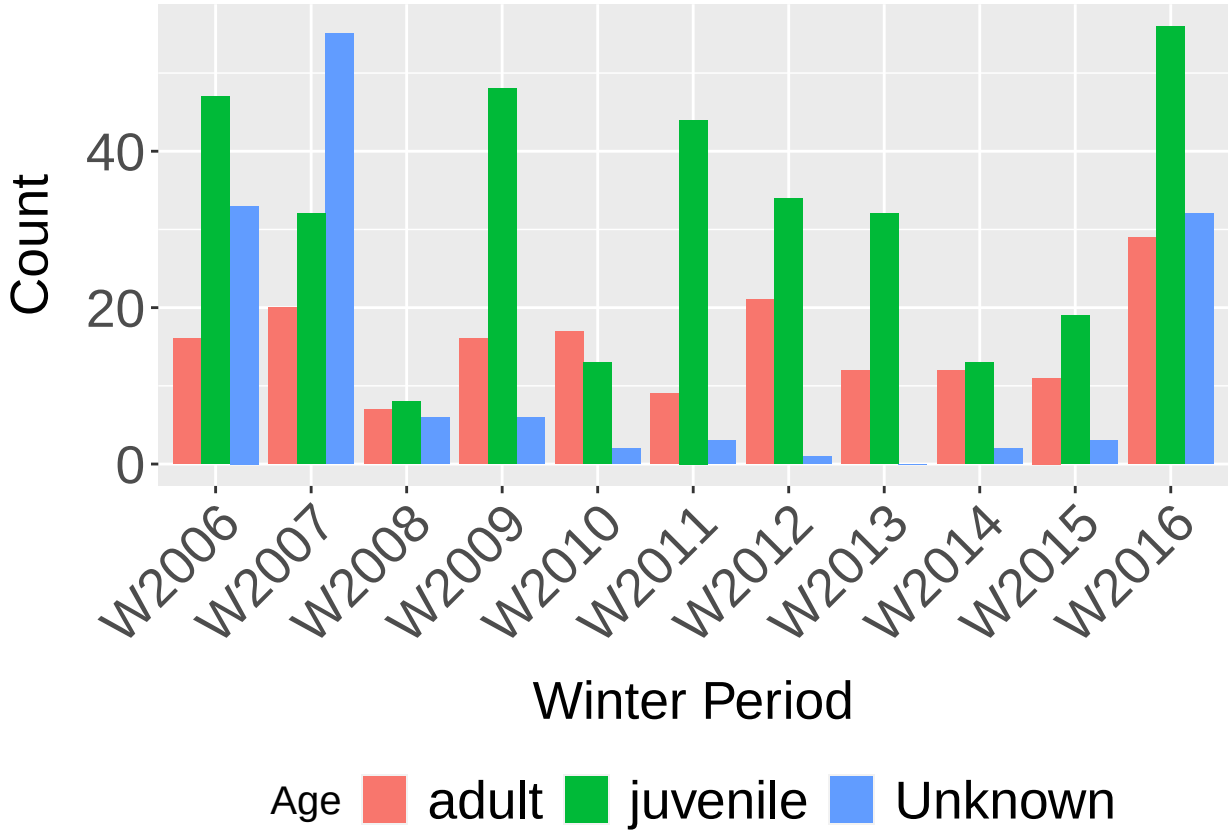
```r
for (species_idx in c("blackcap", "chiffchaff", "robin")) {
  winter_sums <- birds_long %>%
    filter(species == species_idx) %>%
    group_by(winter_period, age_at_ringing) %>%
    summarise(count = sum(count, na.rm = TRUE))

  p_winter <- ggplot(winter_sums, aes(x = factor(winter_period), y = count,
                                      fill = age_at_ringing)) +
    geom_bar(stat = "identity", position = "dodge") +
    scale_x_discrete(name = "Winter Period",
                     labels = function(x) paste0("W", as.numeric(x) - 1)) +
    labs(y = "Count", fill = "Age") +
    custom_theme +
    theme(axis.text.x = element_text(angle = 45, hjust = 1),
          legend.position = "bottom", legend.text = element_text(size = size),
          legend.title = element_text(size = size / (4/3)))

  print(p_winter)
  ggsave(paste0("report/images/", species_idx, "_capture_counts_per_year.png"),
         p_winter, width = 10, height = 8)
}
```

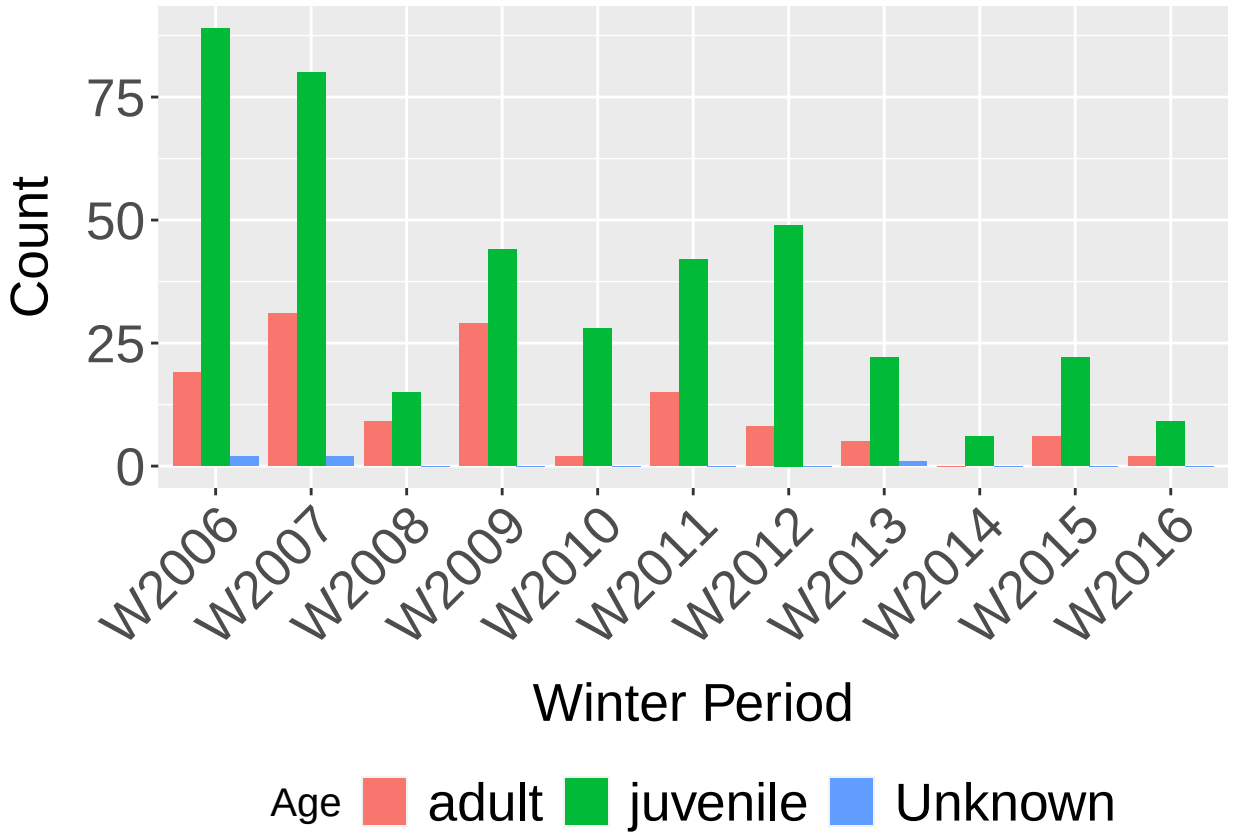
```
## `summarise()` has grouped output by 'winter_period'. You can override using the
## `.groups` argument.
```

```
## `summarise()` has grouped output by 'winter_period'. You can override using the
## `.groups` argument.
```



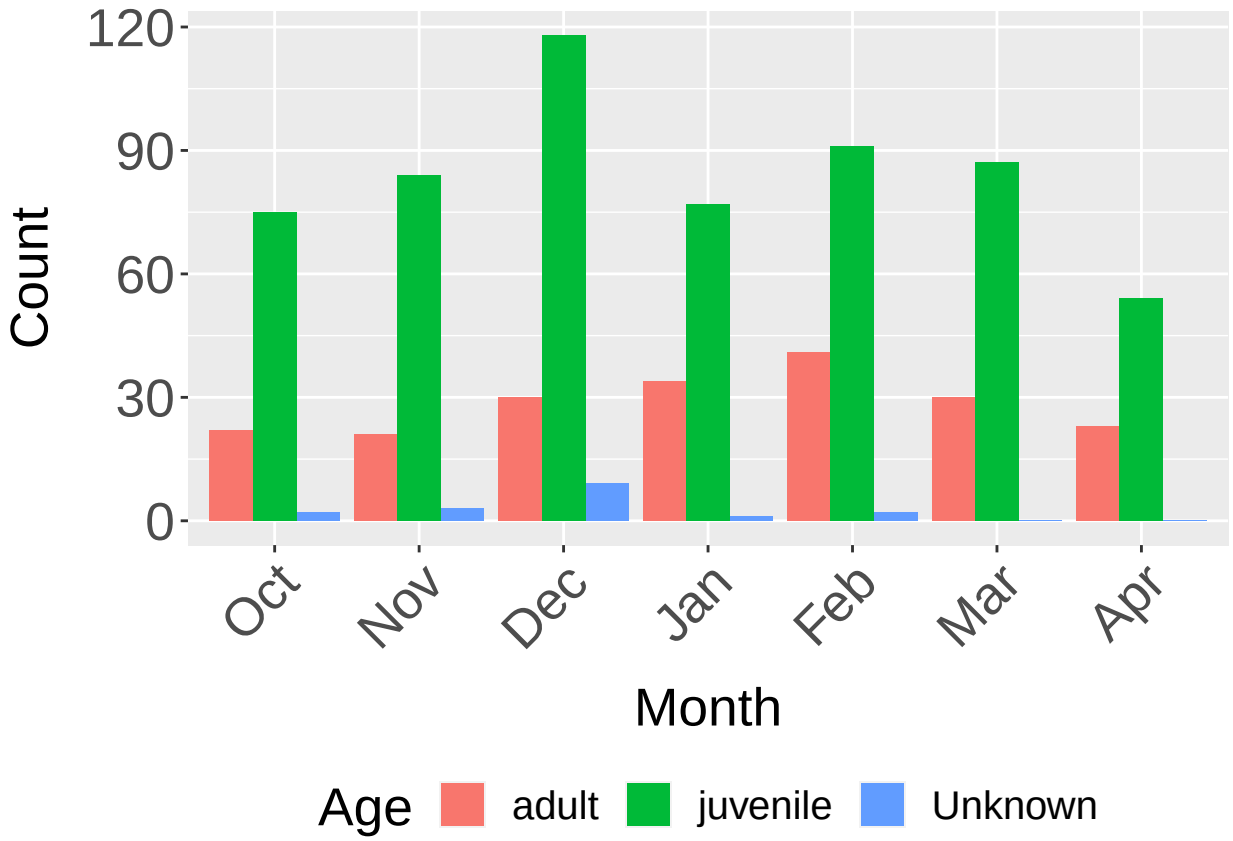
```
## `summarise()` has grouped output by 'winter_period'. You can override using the
## `.groups` argument.
```



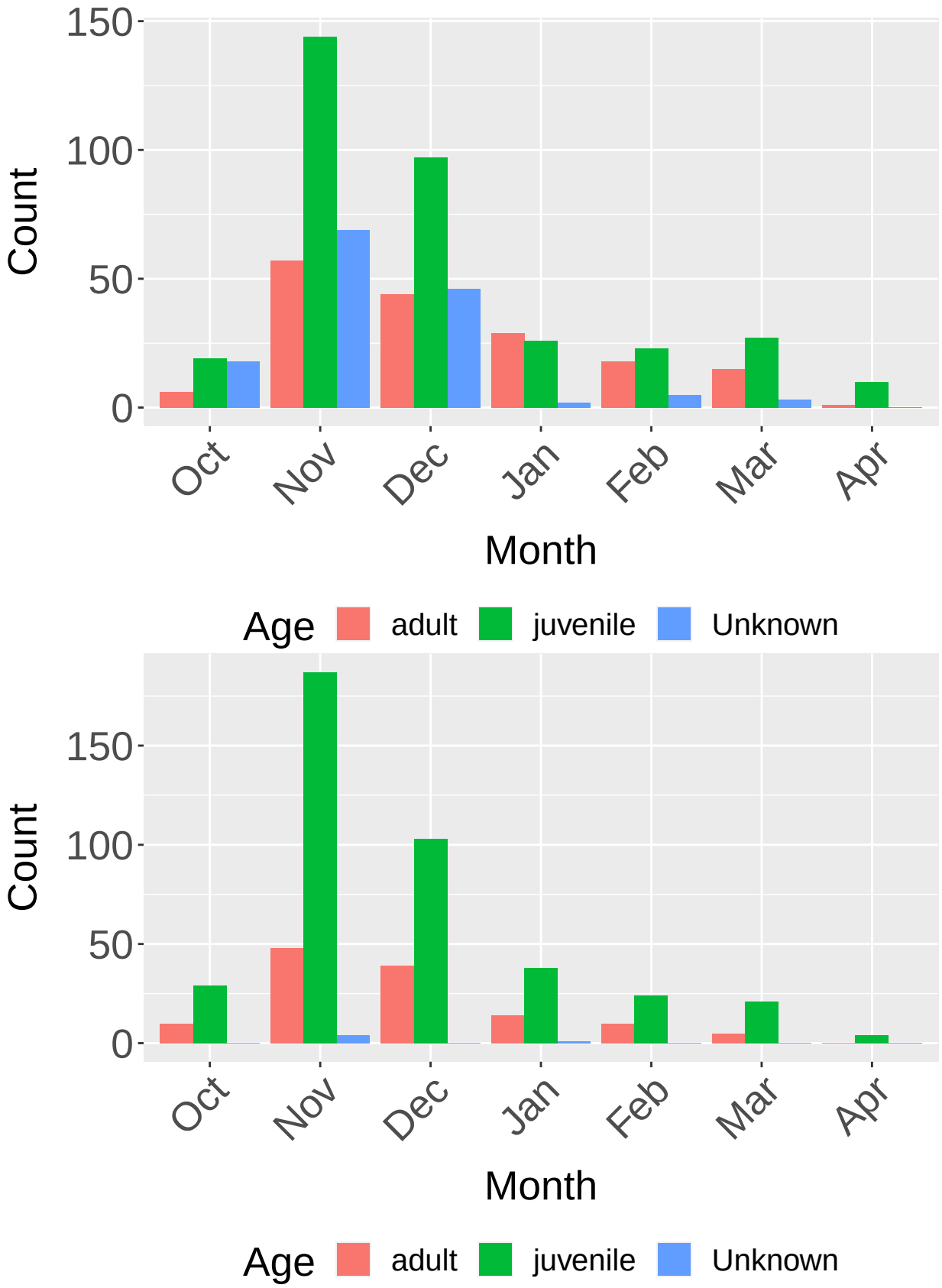



```
##### Capture count across months by age at ringing and for each species ####
for (species_idx in c("blackcap", "chiffchaff", "robin")) {
  monthly_sums <- birds_long %>%
    filter(species == species_idx) %>%
    group_by(month_name, age_at_ringing) %>%
    summarise(count = sum(count, na.rm = TRUE))

  p <- ggplot(monthly_sums, aes(x = month_name, y = count,
                                fill = age_at_ringing)) +
    geom_bar(stat = "identity", position = "dodge") +
    labs(x = "Month", y = "Count", fill = "Age") +
    scale_x_discrete(limit = c("Oct", "Nov", "Dec", "Jan",
                               "Feb", "Mar", "Apr")) +
    custom_theme +
    theme(legend.text = element_text(size = size / (4/3)),
          legend.title = element_text(size = size), axis.text.x =
            element_text(angle = 45, hjust = 1, size = size),
          legend.position = "bottom")

  print(p)
  ggsave(paste0("report/images/", species_idx, "_capture_count_per_month.png"),
         plot = p, width = 10, height = 8)
}
```

```
## `summarise()` has grouped output by 'month_name'. You can override using the
## `.groups` argument.
## `summarise()` has grouped output by 'month_name'. You can override using the
## `.groups` argument.
```




```
## `summarise()` has grouped output by 'month_name'. You can override using the
## `.groups` argument.
```


150
100
50
0
Count
Oct
Nov
Dec
Jan
Feb
Mar
Apr
Month
Age
adult
juvenile
Unknown

```r
##############################
# MODELLING SETUP
##############################

create_inter_winter_dataset <- function(df) {
  #' Create Inter-Winter Dataset
  #'
  #' This function takes a dataset containing bird observation data and
  #' collapses it by creating a new column 'winter_presence', which calculates
  #' if a bird was captured at all during a particular winter season.
  #'
  #' The dataset is restructured by identifying the years and months of
  #' observation and grouping by winter season (October to April)
  #' For each winter season, a presence value is computed for each bird,
  #' indicating whether the bird was captured at least once during that winter.
  #'
  #' @param df A data frame containing bird observation data. It must include
  #' columns for 'id', 'age_at_ringing', 'species', and 'row_sum', plus
  #' additional columns  representing the months of observation.
  #' @return A data frame grouped by bird 'id', 'species', and 'age_at_ringing',
  #'  with columns for each winter season, representing the presence (1) or
  #'  absence (0)  of the bird during that winter. The resulting data frame is
  #'  sorted by species and numeric ID.

  # Check if df is a data.frame and has necessary columns
  if (!is.data.frame(df)) stop("df must be a dataframe.")
  if (!all(c('id', 'age_at_ringing', 'species', 'row_sum') %in% colnames(df)))
    stop("df must have 'id', 'age_at_ringing', 'species', and 'row_sum' columns.")

  df %>%
    pivot_longer(-c(id, age_at_ringing, species, row_sum),
                 names_to = "month", values_to = "value") %>% # Pivot to longer format
    mutate(year = as.integer(str_sub(month, start = 1, end = 4)),
           month_num = as.integer(str_sub(month, start = 5, end = 6))) %>% # Extract
    # year & month
    mutate(winter = ifelse(month_num >= 10, year, year - 1)) %>% # Calculate
    # winter period
    group_by(id, winter, age_at_ringing, species) %>%
    summarise(winter_presence = ifelse(any(value == 1), 1, 0)) %>% # Calculate
    # if a bird appeared in that winter
    pivot_wider(names_from = winter, values_from = winter_presence) %>%
    mutate(numeric_id = as.numeric(str_extract(id, "\\d+"))) %>% # Extract
    # numeric ID
    arrange(species, numeric_id) %>% # Arrange by species & numeric ID
    select(-numeric_id) %>%
    ungroup

}

birds_inter <- create_inter_winter_dataset(birds)
```

```
## `summarise()` has grouped output by 'id', 'winter', 'age_at_ringing'. You can
## override using the `.groups` argument.
```

```r
create_intra_winter_dataset <- function(df) {
  #' Create Intra-Winter Dataset
  #'
  #' This function transforms a given dataset into an intra-winter format,
  #' pivoting and reformatting the data so that each winter has its own column.
  #' The dataset is expected to contain bird observation data, with columns
  #' representing years and months.
  #'
  #' @param df Data frame with bird observation data, with columns starting
  #' with "20" representing year and month.
  #' @return A data frame where each row represents an observation, with columns
  #'   for winter months (Oct, Nov, Dec, Jan, Feb, Mar, Apr) and additional
  #'   information like species and ID. If there are duplicated IDs due to
  #'   multiple captures in different years, only the first year is considered.
  #' @note This function assumes that the input data frame contains specific
  #'   columns, including a unique identifier 'id' and a species identifier.
  #'   Any rows where the values for months 1 to 7 are all zero will be filtered
  #'   out.

  # Check if df is a data.frame and has necessary columns
  if (!is.data.frame(df)) stop("df must be a dataframe.")
  if (!"id" %in% colnames(df)) stop("df must have 'id' column.")
  df_long <- birds %>%
    pivot_longer(cols = starts_with("20"),
                 names_to = "year_month",
                 values_to = "value") %>%
    separate(year_month, into = c("year", "month"), sep = 4)

  df_long$year <- as.numeric(df_long$year)
  df_long$month <- as.numeric(df_long$month)

  # Creating a new variable for winter
  df_long <- df_long %>%
    mutate(winter = ifelse(month >= 10, year, year - 1)) %>%
    mutate(winter_month = ifelse(month >= 10, month - 9, month + 3)) %>%
    select(-c(month, year))

  # Pivoting data so each winter has its own column
  df_wide <- df_long %>%
    pivot_wider(names_from = winter_month, values_from = value)  %>%
    filter(rowSums(.[,c('1', '2', '3', '4', '5', '6', '7')]) != 0) %>%
    group_by(id) %>%
    slice_min(row_number(), n = 1) %>% # Where the id column is duplicated
    # because of a capture appearing multiple times for different years, take
    # the first year
    ungroup() %>%
    # sort by id column
    mutate(numeric_id = as.numeric(str_extract(id, "\\d+"))) %>%
    arrange(species, numeric_id) %>%
    select(-numeric_id)

  # Renaming columns
  df_wide <- df_wide %>%
```

```
    rename(`Oct` = `1`,
           `Nov` = `2`,
           `Dec` = `3`,
           `Jan` = `4`,
           `Feb` = `5`,
           `Mar` = `6`,
           `Apr` = `7`)

  df_wide
}

birds_intra <- create_intra_winter_dataset(birds)

######### Separate out by species #####
blackcap_inter <- birds_inter %>% filter(species == 'blackcap') %>% ungroup
robin_inter <- birds_inter %>% filter(species == 'robin') %>% ungroup
chiffchaff_inter <- birds_inter %>% filter(species == 'chiffchaff') %>% ungroup


blackcap_intra <- birds_intra %>% filter(species == 'blackcap') %>% ungroup
robin_intra <- birds_intra %>% filter(species == 'robin') %>% ungroup
chiffchaff_intra <- birds_intra %>% filter(species == 'chiffchaff') %>% ungroup
########
set.seed(42)

n_iter <-10000 # Number of iterations
n_burnin <-2000 # Number of burnin samples
n_chains <-3
n_adapt <- 1000

marray <- function(CH){
  #' Create an m-array from capture history (CH) data
  #'
  #' This function takes a capture history (CH) matrix, where rows represent
  #' individual animals and columns represent capture occasions. It returns an
  #' m-array, which is a matrix representing the transition of marked individuals
  #' between capture occasions, including those not recaptured.
  #'
  #' @param CH A matrix of capture history (CH). Rows correspond to individuals,
  #'   and columns correspond to occasions. The value is 1 if captured, 0
  #'   otherwise.
  #' @return A matrix representing the m-array, with columns for each occasion
  #'   and an additional column for never recaptured individuals.
  #'
  n_ind <- dim(CH)[1] # Number of individuals
  n_occasions <- dim(CH)[2] # Number of occasions (time periods)

  # Initialise the m-array with zeros, with columns for each occasion and an
  # additional one for never recaptured
  m_array <- matrix(data = 0, ncol = n_occasions+1, nrow = n_occasions)

  # Iterate through each occasion, summing the captures for that time period
  for (t in 1:n_occasions){
```

```
    m_array[t,1] <- sum(CH[,t])
  }

  # Iterate through each individual
  for (i in 1:n_ind){
    pos <- which(CH[i,] != 0) # Find positions where captures occurred
    num_captures <- length(pos) # Count the number of captures for this
    # individual

    # Iterate through captures, incrementing the counts in the corresponding
    # m-array cells
    for (j in 1:(num_captures-1)){
      m_array[pos[j], pos[j+1]] <- m_array[pos[j], pos[j+1]] + 1
    }
  }

  # Iterate through each occasion again, calculating the number of individuals
  # that were never recaptured
  for (t in 1:n_occasions){
    m_array[t,n_occasions+1] <- m_array[t,1] - sum(m_array[t,2:n_occasions])
  }

  # Return the m-array, excluding the last row and first column as they contain
  # redundant information
  res <- m_array[1:(n_occasions-1),2:(n_occasions+1)]
  return(res)
}

## Choose random initial values
inits_m0_inter <- function(chain){
  list(
    phi_juv0 = runif(1),
    phi_adult0 = runif(1),
    p0 = runif(1))
}

inits_mt_age_inter <- function(chain){
  list(
    phi_juv = runif(10, 0, 1),
    phi_adult = runif(10, 0, 1),
    p = runif(10, 0, 1))
}

inits_m0_intra <- function(chain){
  list(
    phi0 = runif(1),
    p0 = runif(1))
}

inits_mt_intra <- function(chain){
  list(
    phi = runif(6, 0, 1),
    p = runif(6, 0, 1))
```

```
}

jags_model_multinomial_age_inter <- function(df, hyper_par, inits,
                                             model_type, params) {
  #' Run inter-winter JAGS model
  #'
  #' This function processes capture history data, separates juveniles and adults,
  #' and runs a JAGS model to estimate survival and recapture probabilities for
  #' different age classes. The JAGS model is defined in an external file.
  #'
  #' @param df A data frame containing capture history data, with columns for
  #'   individual ID, age at ringing (juvenile or adult), species, and capture
  #'   occasions.
  #' @param hyper_par A list containing hyperparameters for the JAGS model.
  #' @param inits A list containing initial values for the JAGS model parameters.
  #' @param model_type A character string indicating the model type
  #' @param params A character vector containing the names of parameters to be
  #'  returned from the JAGS model
  #' @return An object containing the JAGS model output.
  #'


  # Check if df is a data.frame and has necessary columns
  if (!is.data.frame(df)) stop("df must be a dataframe.")
  if (!all(c("id", "age_at_ringing", "species") %in% colnames(df))) {
    stop("df must have 'id', 'age_at_ringing' and 'species' columns.")
  }

  # Check if hyper_par and inits are lists
  if (!is.list(hyper_par)) stop("hyper_par must be a list.")
  if (!is.function(inits)) stop("inits must be a function")

  # Check if model_type is a character string
  if (!is.character(model_type)) stop("model_type must be a character string.")

  # Check if params is a character vector
  if (!is.character(params)) stop("params must be a character vector.")

  CH_juv <- df %>% filter(age_at_ringing == "juvenile") %>%
    select(c(-id, -age_at_ringing, -species)) # Juvenile capture history
  CH_adult <- df %>% filter(age_at_ringing == "adult") %>%
    select(c(-id, -age_at_ringing, -species)) # Adult capture history

  # Calculate the sum of captures for each juvenile
  cap_sum <- apply(CH_juv, 1, sum)

  # Identify juveniles that have been recaptured at least once
  idx <- which(cap_sum >= 2)

  # Separate juveniles into two categories: those recaptured at least once,
  # and those never recaptured
  CH_juv_recap <- CH_juv[idx,]  # Juveniles recaptured at least once
  CH_juv_never_recap <- CH_juv[-idx,] # Juveniles never recaptured
```

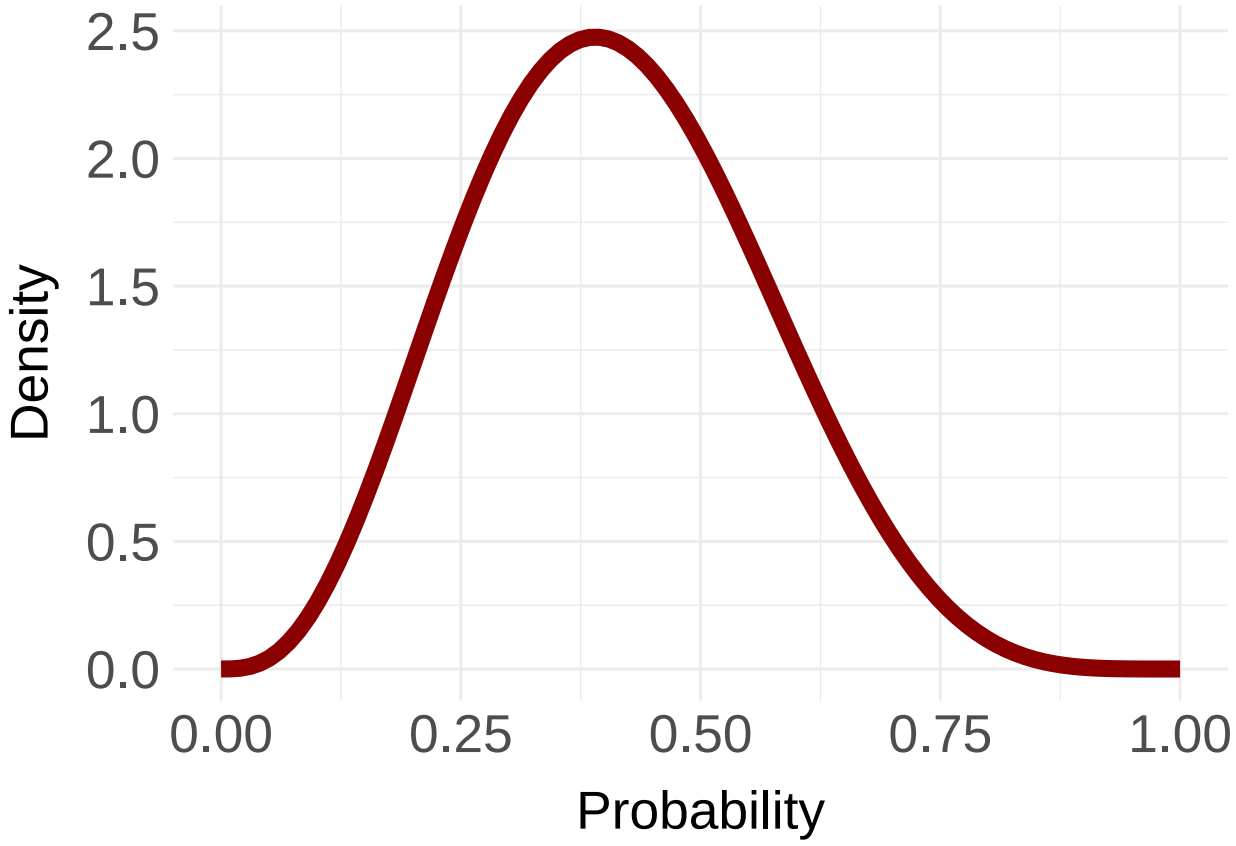
```r
# Process juveniles that have been recaptured at least once to remove the
# first capture
first <- numeric()
for (i in 1:dim(CH_juv_recap)[1]) {
  first[i] <- min(which(CH_juv_recap[i,] == 1)) # Identify the first recapture
  # for each juvenile
  CH_juv_recap[i, first[i]] <- 0            # Set the first recapture to 0
}

# Combine grown-up juveniles with adults and convert to m-array format
CH_adult_m <- rbind(CH_adult, CH_juv_recap)
marr_adult <- marray(CH_adult_m)

# Create a matrix to store the first and second recaptures for juveniles
CH_juv_recap2 <- matrix(0, nrow = dim(CH_juv_recap)[1],
                        ncol = dim(CH_juv_recap)[2])
second <- numeric()
for (i in 1:dim(CH_juv_recap)[1]) {
  second[i] <- min(which(CH_juv_recap[i,] == 1)) # Identify the second recapture
  # for each juvenile
  CH_juv_recap2[i, first[i]] <- 1          # Mark the first recapture
  CH_juv_recap2[i, second[i]] <- 1          # Mark the second recapture
}

# Convert the processed juvenile recapture data into m-array format
CH_juv_recap_marray <- marray(CH_juv_recap2)
CH_juv_recap_marray[, dim(CH_juv)[2]] <- 0 # Ensure that the last column is zero
# since all of them are released as adults

# Convert the juveniles that were never recaptured into m-array format
CH_juv_never_recap_marray <- marray(CH_juv_never_recap)

# Combine the m-arrays for recaptured and never-recaptured juveniles
marr_juv <- CH_juv_recap_marray + CH_juv_never_recap_marray

# Calculate the sum of releases for each capture occasion
releases_juv <- rowSums(marr_juv)
releases_adult <- rowSums(marr_adult)

# prepare JAGS data
jdata <- list(marr_juv = marr_juv, marr_adult = marr_adult,
              releases_juv = releases_juv, releases_adult = releases_adult,
              n_occasions=ncol(marr_juv),
              alpha_phi_juv = hyper_par$alpha_phi_juv,
              beta_phi_juv = hyper_par$beta_phi_juv,
              alpha_phi_adult = hyper_par$alpha_phi_adult,
              beta_phi_adult = hyper_par$beta_phi_adult,
              alpha_p = hyper_par$alpha_p,
              beta_p = hyper_par$beta_p)

# run JAGS model
out <- jags(jdata, inits, params,
            paste0("multinomial_",model_type,"_age_inter.jags"),
```

```
              DIC=TRUE, n.chains=n_chains, n.adapt=n_adapt, n.iter=n_iter,
              parallel = TRUE, seed = 42)
  out
}

jags_model_multinomial_intra <- function(df, hyper_par, inits,
                                         model_type, params) {
  #' Run intra-winter JAGS model
  #'
  #' This function prepares capture history data for intra-seasonal analysis,
  #' converting it into a suitable format and running a JAGS model to estimate
  #' survival and recapture probabilities. The JAGS model is defined in an
  #' external file.
  #'
  #' @param df A data frame containing capture history data, with columns for
  #'   individual ID, age at ringing, species, winter, and capture occasions.
  #' @param hyper_par A list containing hyperparameters for the JAGS model.
  #' @param inits A list containing initial values for the JAGS model parameters.
  #' @param model_type A character string indicating the model type
  #' @param params A character vector containing the names of parameters to be
  #'  returned from the JAGS model
  #'   by JAGS
  #' @return An object containing the JAGS model output.
  #'

  # Check if df is a data.frame and has necessary columns
  if (!is.data.frame(df)) stop("df must be a dataframe.")
  if (!all(c("id", "age_at_ringing", "species", "winter")
           %in% colnames(df))) {
    stop("df must have 'id', 'age_at_ringing', 'species' and 'winter' columns.")
  }

  # Check if hyper_par and inits are lists
  if (!is.list(hyper_par)) stop("hyper_par must be a list.")
  if (!is.function(inits)) stop("inits must be a function")

  # Check if model_type is a character string
  if (!is.character(model_type)) stop("model_type must be a character string.")

  # Check if params is a character vector
  if (!is.character(params)) stop("params must be a character vector.")

  # Create m-array
  marr <-  marray(df %>% select(-c(id, age_at_ringing, species, winter,
                                   row_sum)))

  # Calculate the sum of releases for each capture occasion
  releases <- rowSums(marr)

  # prepare JAGS data
  jdata <- list(marr=marr,
                R = releases,
                n_occasions=ncol(marr),
```

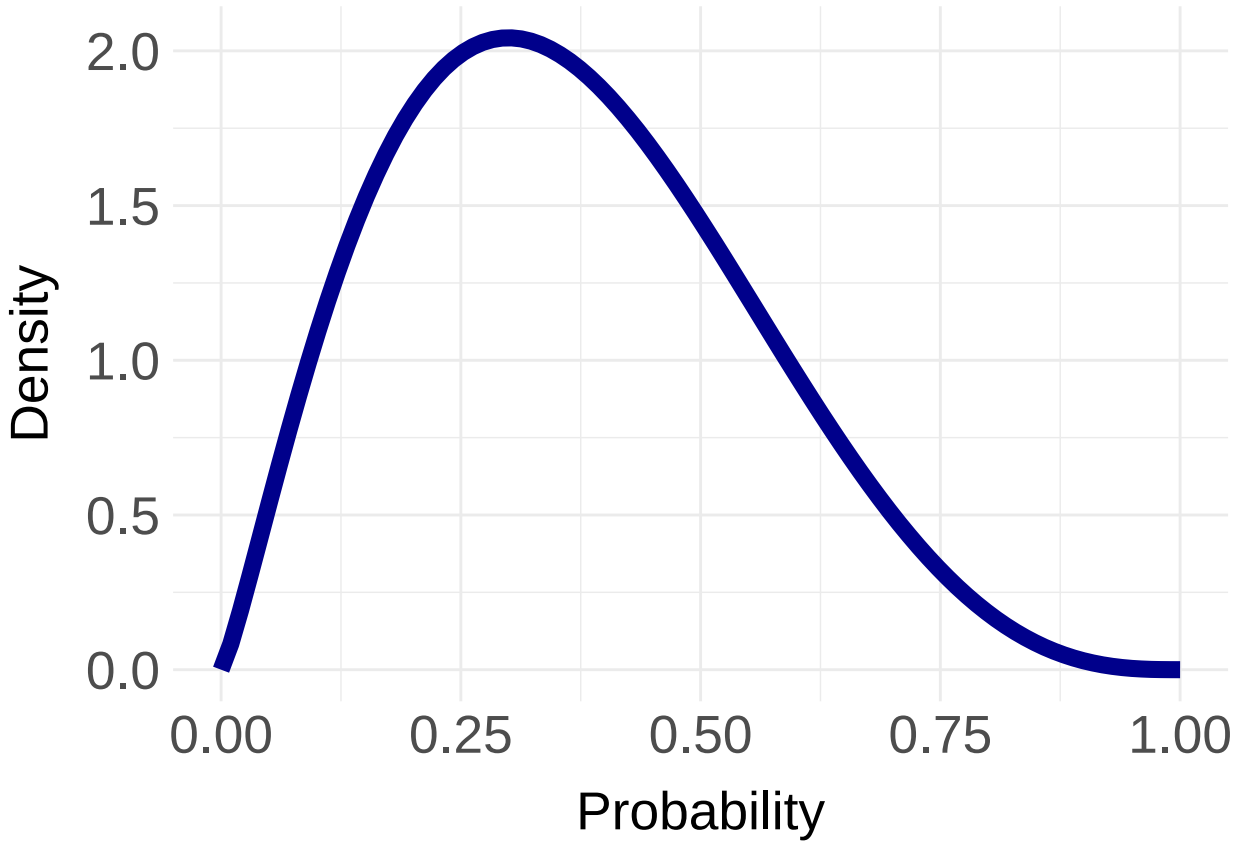
```
                alpha_phi= hyper_par$alpha_phi,
                beta_phi = hyper_par$beta_phi,
                alpha_p = hyper_par$alpha_p,
                beta_p = hyper_par$beta_p)

  # run JAGS model
  out <- jags(jdata, inits, params,
              paste0("multinomial_",model_type,"_intra.jags"),
              DIC=TRUE, n.chains=n_chains, n.adapt=n_adapt, n.iter=n_iter,
              parallel = TRUE, seed = 42)
  out
}


##############################
# PRIOR DISTRIBUTIONS
##############################
# Define parameters for the Beta distributions
alpha_phi_juv_blackcap <- alpha_phi_adult_blackcap <- 4
beta_phi_juv_blackcap <- beta_phi_adult_blackcap <- 5.69

alpha_phi_juv_chiffchaff <- alpha_phi_adult_chiffchaff <- 2.24
beta_phi_juv_chiffchaff  <- beta_phi_adult_chiffchaff <- 3.9

alpha_phi_juv_robin <- alpha_phi_adult_robin <- 3.4
beta_phi_juv_robin <- beta_phi_adult_robin <- 2.41

alpha_p <- 1.5
beta_p <- 3.5

params <- list(c(alpha_phi_juv_blackcap, beta_phi_juv_blackcap),
               c(alpha_phi_juv_chiffchaff, beta_phi_juv_chiffchaff),
               c(alpha_phi_juv_robin, beta_phi_juv_robin),
               c(alpha_p, beta_p))
names <- c("blackcaps", "chiffchaffs", "robins", "capture_probability")
colors <- c("#8B0000", "#00008B", "#006400", "black")

# Generate and plot prior distributions
for(i in 1:4){
  x <- seq(0, 1, length.out = 100)
  y <- dbeta(x, params[[i]][1], params[[i]][2])
  df <- data.frame(x = x, y = y)

  p <- ggplot(df, aes(x = x, y = y)) +
    geom_line(color = colors[i], linewidth = 3) +
    theme_minimal() +
    custom_theme +

    xlab("Probability") +
    ylab("Density")

  ggsave(filename = paste0("report/images/prior_distribution_for_", names[i],
                           ".png"))
```

```
  print(p)
}
```

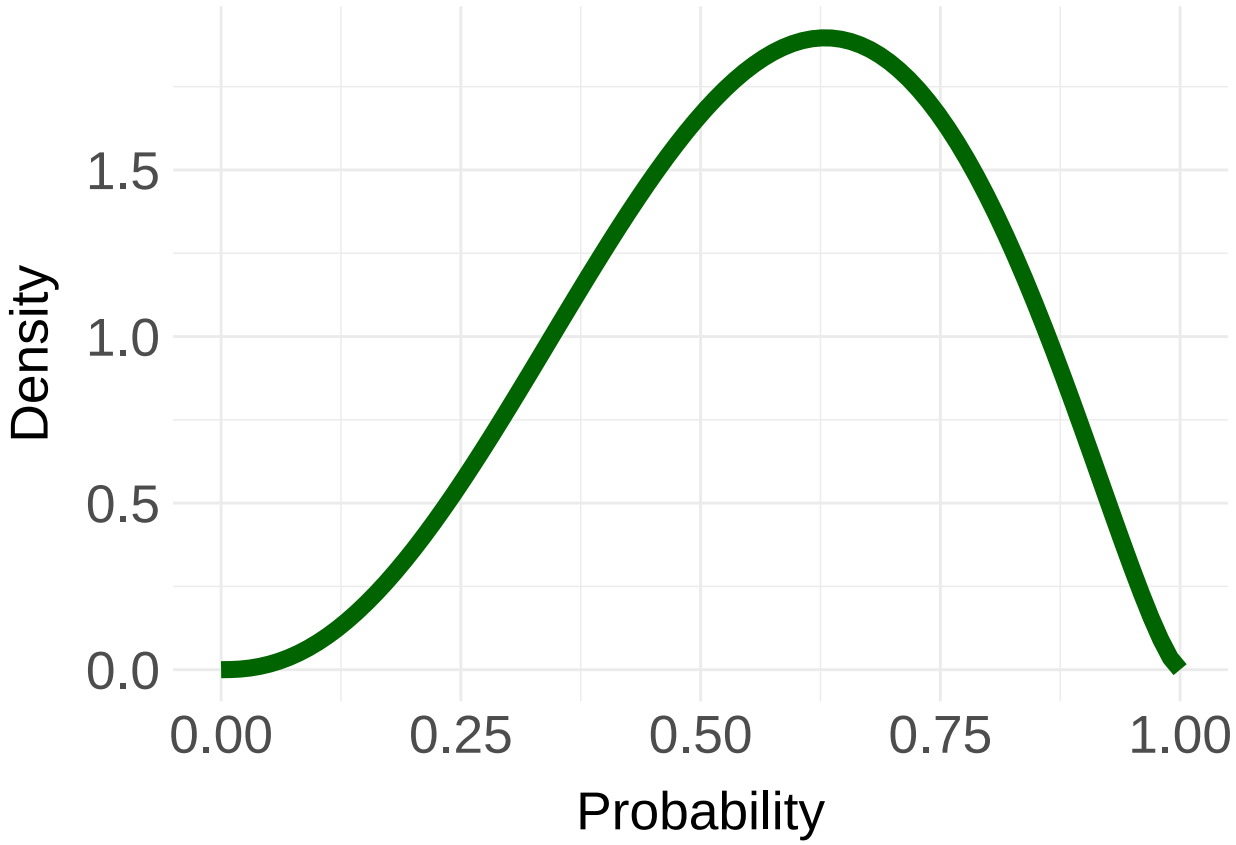
```
## Saving 6.5 x 4.5 in image
## Saving 6.5 x 4.5 in image
```

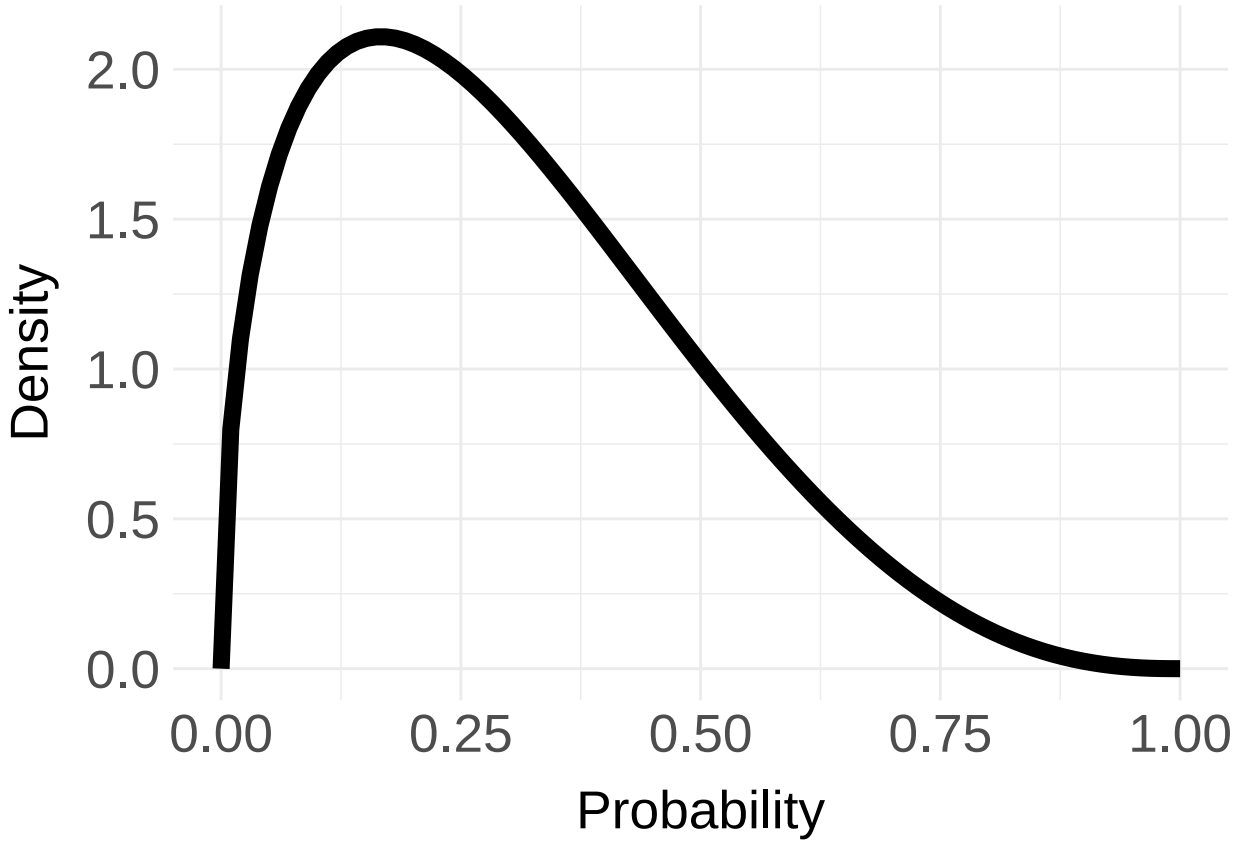



```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```


Density
1.5
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
Probability


Density
2.0
1.5
1.0
0.5
0.0
0.00
0.25
0.50
0.75
1.00
Probability

```
##############################
# INTER-WINTER MODELLING
##############################
########## M_0 blackcap model ############
blackcap_m0_age_inter <- jags_model_multinomial_age_inter(blackcap_inter,
                                    inits = inits_m0_inter,
                                    model_type = "m0",
                                    hyper_par = list(
                                    alpha_phi_juv = alpha_phi_juv_blackcap,
                                    beta_phi_juv = beta_phi_juv_blackcap,
                                    alpha_phi_adult= alpha_phi_adult_blackcap,
                                    beta_phi_adult= beta_phi_adult_blackcap,
                                    alpha_p = alpha_p,
                                    beta_p = beta_p),
                                    params = c("phi_juv0",
                                               "phi_adult0",
                                               "p0", "mean_phi_diff",
                                               "disc_obs_juv",
                                               "disc_rep_juv",
                                               "disc_obs_adult",
                                               "disc_rep_adult"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
blackcap_m0_age_inter
```

```
## JAGS output for model 'multinomial_m0_age_inter.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.042 minutes at time 2023-08-07 23:33:11.
##
##                   mean    sd    2.5%     50%   97.5% overlap0     f Rhat n.eff
## phi_juv0         0.324 0.100   0.164   0.311   0.550    FALSE 1.000    1 30000
## phi_adult0       0.361 0.065   0.243   0.359   0.493    FALSE 1.000    1 11370
## p0               0.107 0.036   0.052   0.101   0.194    FALSE 1.000    1 30000
## mean_phi_diff    0.037 0.105  -0.195   0.045   0.219     TRUE 0.668    1 23884
## disc_obs_juv    16.279 1.763  13.537  16.044  20.308    FALSE 1.000    1 30000
```

```
## disc_rep_juv     9.802 2.232   5.906   9.644  14.639    FALSE 1.000    1 10417
## disc_obs_adult   8.606 1.204   6.994   8.373  11.591    FALSE 1.000    1 30000
## disc_rep_adult   7.333 1.693   4.460   7.186  11.035    FALSE 1.000    1 16385
## deviance       116.070 2.026 113.753 115.560 121.250    FALSE 1.000    1 13053
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 2.1 and DIC = 118.122
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(blackcap_m0_age_inter$summary[,-11]),
   caption = "Blackcap $M_0$ inter-winter model diagnostic and summary output",
   label = "tab:blackcap_m0_inter_summary_output", type = "latex"),
      file = "report/model_outputs/blackcap_m0_age_inter.tex")

########## M_0 chiffchaff model ############
chiffchaff_m0_age_inter <- jags_model_multinomial_age_inter(chiffchaff_inter,
                                  inits = inits_m0_inter,
                                  model_type = "m0",
                                  hyper_par = list(
                                  alpha_phi_juv = alpha_phi_juv_chiffchaff,
                                  beta_phi_juv = beta_phi_juv_chiffchaff,
                                  alpha_phi_adult_= alpha_phi_adult_chiffchaff,
                                  beta_phi_adult = beta_phi_adult_chiffchaff,
                                  alpha_p = alpha_p,
                                  beta_p = beta_p),
                                  params = c("phi_juv0", "phi_adult0",
                                             "p0", "mean_phi_diff",
                                             "disc_obs_juv",
                                             "disc_rep_juv",
                                             "disc_obs_adult",
                                             "disc_rep_adult"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
```

```
## Done.
```

```
chiffchaff_m0_age_inter
```

```
## JAGS output for model 'multinomial_m0_age_inter.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.054 minutes at time 2023-08-07 23:33:14.
##
##                   mean    sd    2.5%     50%   97.5% overlap0     f  Rhat n.eff
## phi_juv0         0.223 0.075   0.108   0.213   0.396    FALSE 1.000 1.000 21414
## phi_adult0       0.409 0.066   0.286   0.407   0.543    FALSE 1.000 1.000 26961
## p0               0.184 0.055   0.094   0.178   0.309    FALSE 1.000 1.001  5284
## mean_phi_diff    0.187 0.087   0.006   0.190   0.346    FALSE 0.978 1.000 30000
## disc_obs_juv    12.087 1.682   9.534  11.846  16.013    FALSE 1.000 1.000 30000
## disc_rep_juv     8.298 1.921   4.978   8.161  12.464    FALSE 1.000 1.000 24539
## disc_obs_adult   9.833 1.110   8.401   9.599  12.599    FALSE 1.000 1.000 17585
## disc_rep_adult   9.241 2.042   5.615   9.108  13.609    FALSE 1.000 1.000 17073
## deviance       113.924 2.250 111.384 113.340 119.656    FALSE 1.000 1.000 30000
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 2.5 and DIC = 116.456
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(chiffchaff_m0_age_inter$summary[,-11]),
 caption = "Chiffchaff $M_0$ inter-winter model diagnostic and summary output",
 label = "tab:chiffchaff_m0_inter_summary_output", type = "latex"),
      file = "report/model_outputs/chiffchaff_m0_age_inter.tex")

########## M_0 robin model #############
robin_m0_age_inter <- jags_model_multinomial_age_inter(robin_inter,
                                        inits = inits_m0_inter,
                                        model_type = "m0",
                                        hyper_par = list(
                                        alpha_phi_juv = alpha_phi_juv_robin,
                                        beta_phi_juv = beta_phi_juv_robin,
                                        alpha_phi_adult= alpha_phi_adult_robin,
                                        beta_phi_adult= beta_phi_adult_robin,
                                        alpha_p = alpha_p,
                                        beta_p = beta_p),
                                        params = c("phi_juv0",
                                                   "phi_adult0",
                                                   "p0", "mean_phi_diff",
                                                   "disc_obs_juv",
                                                   "disc_rep_juv",
```

```
                                                    "disc_obs_adult",
                                                    "disc_rep_adult"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
robin_m0_age_inter
```

```
## JAGS output for model 'multinomial_m0_age_inter.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.038 minutes at time 2023-08-07 23:33:18.
##
##                   mean    sd    2.5%     50%   97.5% overlap0     f Rhat n.eff
## phi_juv0         0.705 0.110   0.493   0.705   0.912    FALSE 1.000    1 22612
## phi_adult0       0.776 0.040   0.698   0.775   0.854    FALSE 1.000    1 15038
## p0               0.063 0.013   0.041   0.061   0.090    FALSE 1.000    1 24620
## mean_phi_diff    0.071 0.117  -0.151   0.070   0.298     TRUE 0.717    1 16691
## disc_obs_juv    28.450 1.624  25.906  28.248  32.142    FALSE 1.000    1 30000
## disc_rep_juv    18.921 2.781  13.853  18.772  24.776    FALSE 1.000    1 23151
## disc_obs_adult  18.145 1.682  15.492  17.944  22.008    FALSE 1.000    1 25258
## disc_rep_adult  16.387 2.258  12.294  16.286  21.121    FALSE 1.000    1 19059
## deviance       228.992 2.303 226.451 228.383 235.032    FALSE 1.000    1 13411
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 2.7 and DIC = 231.644
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(robin_m0_age_inter$summary[,-11]),
       caption = "Robin $M_0$ inter-winter model diagnostic and summary output",
       label = "tab:robin_m0_inter_summary_output", type = "latex"),
```

```
      file = "report/model_outputs/robin_m0_age_inter.tex")

########## M_t blackcap model ############
blackcap_mt_age_inter <- jags_model_multinomial_age_inter(blackcap_inter,
                                      inits = inits_mt_age_inter,
                                      model_type = "mt",
                                      hyper_par = list(
                                      alpha_phi_juv = alpha_phi_juv_blackcap,
                                      beta_phi_juv = beta_phi_juv_blackcap,
                                      alpha_phi_adult= alpha_phi_adult_blackcap,
                                      beta_phi_adult= beta_phi_adult_blackcap,
                                      alpha_p = alpha_p,
                                      beta_p = beta_p),
                                      params = c("phi_juv", "phi_adult",
                                                 "p", "phi_diff",
                                                 "disc_obs_juv",
                                                 "disc_rep_juv",
                                                 "disc_obs_adult",
                                                 "disc_rep_adult"))
```

```
##
## Processing function input.......
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
blackcap_mt_age_inter
```

```
## JAGS output for model 'multinomial_mt_age_inter.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.088 minutes at time 2023-08-07 23:33:20.
##
##                  mean    sd   2.5%    50%   97.5% overlap0     f  Rhat n.eff
## phi_juv[1]      0.319 0.129  0.111  0.305   0.600    FALSE 1.000 1.001  3223
## phi_juv[2]      0.257 0.119  0.077  0.239   0.534    FALSE 1.000 1.000 13928
## phi_juv[3]      0.416 0.133  0.180  0.409   0.692    FALSE 1.000 1.000 15704
## phi_juv[4]      0.403 0.136  0.165  0.394   0.687    FALSE 1.000 1.000 14194
## phi_juv[5]      0.251 0.127  0.064  0.230   0.550    FALSE 1.000 1.001  3489
## phi_juv[6]      0.310 0.137  0.091  0.295   0.613    FALSE 1.000 1.000 30000
## phi_juv[7]      0.305 0.131  0.096  0.290   0.597    FALSE 1.000 1.000  4700
```

```
## phi_juv[8]      0.527 0.130  0.282  0.527   0.778    FALSE 1.000 1.000 23982
## phi_juv[9]      0.357 0.131  0.136  0.346   0.638    FALSE 1.000 1.000 29221
## phi_juv[10]     0.469 0.136  0.216  0.465   0.739    FALSE 1.000 1.000 30000
## phi_adult[1]    0.312 0.137  0.091  0.298   0.612    FALSE 1.000 1.000 23696
## phi_adult[2]    0.438 0.138  0.190  0.431   0.720    FALSE 1.000 1.000  5703
## phi_adult[3]    0.327 0.130  0.114  0.313   0.612    FALSE 1.000 1.000 21316
## phi_adult[4]    0.358 0.129  0.145  0.345   0.638    FALSE 1.000 1.000 16668
## phi_adult[5]    0.407 0.141  0.160  0.399   0.699    FALSE 1.000 1.000 11273
## phi_adult[6]    0.369 0.143  0.128  0.358   0.669    FALSE 1.000 1.000 11400
## phi_adult[7]    0.321 0.135  0.103  0.307   0.621    FALSE 1.000 1.000  7363
## phi_adult[8]    0.326 0.134  0.107  0.312   0.622    FALSE 1.000 1.000 30000
## phi_adult[9]    0.455 0.129  0.224  0.448   0.725    FALSE 1.000 1.000 30000
## phi_adult[10]   0.316 0.131  0.108  0.299   0.608    FALSE 1.000 1.000 30000
## p[1]            0.117 0.075  0.024  0.100   0.309    FALSE 1.000 1.002  2388
## p[2]            0.083 0.052  0.017  0.071   0.215    FALSE 1.000 1.001  6244
## p[3]            0.106 0.062  0.024  0.093   0.263    FALSE 1.000 1.000 11482
## p[4]            0.313 0.128  0.117  0.295   0.611    FALSE 1.000 1.000 30000
## p[5]            0.100 0.065  0.021  0.085   0.270    FALSE 1.000 1.001  3304
## p[6]            0.045 0.041  0.003  0.033   0.155    FALSE 1.000 1.001 30000
## p[7]            0.129 0.084  0.026  0.110   0.342    FALSE 1.000 1.000 30000
## p[8]            0.128 0.070  0.030  0.115   0.299    FALSE 1.000 1.000 19743
## p[9]            0.209 0.086  0.080  0.195   0.410    FALSE 1.000 1.000 19755
## p[10]           0.241 0.104  0.091  0.224   0.493    FALSE 1.000 1.000 22126
## phi_diff[1]    -0.007 0.175 -0.351 -0.009   0.343     TRUE 0.520 1.000  5923
## phi_diff[2]     0.181 0.170 -0.160  0.183   0.509     TRUE 0.858 1.000  5095
## phi_diff[3]    -0.089 0.174 -0.423 -0.092   0.262     TRUE 0.701 1.000 16657
## phi_diff[4]    -0.045 0.175 -0.382 -0.048   0.307     TRUE 0.609 1.000 30000
## phi_diff[5]     0.157 0.175 -0.198  0.157   0.491     TRUE 0.820 1.000 19500
## phi_diff[6]     0.059 0.189 -0.318  0.059   0.427     TRUE 0.627 1.000 30000
## phi_diff[7]     0.016 0.174 -0.327  0.014   0.360     TRUE 0.536 1.000 14932
## phi_diff[8]    -0.201 0.181 -0.539 -0.206   0.166     TRUE 0.867 1.000 25483
## phi_diff[9]     0.098 0.171 -0.239  0.098   0.432     TRUE 0.717 1.000 30000
## phi_diff[10]   -0.153 0.172 -0.481 -0.156   0.193     TRUE 0.814 1.000 30000
## disc_obs_juv   11.053 2.317  7.160 10.826  16.167    FALSE 1.000 1.000  7279
## disc_rep_juv    9.554 2.206  5.777  9.370  14.378    FALSE 1.000 1.000 30000
## disc_obs_adult  9.090 1.502  6.553  8.954  12.376    FALSE 1.000 1.000 27471
## disc_rep_adult  7.016 1.675  4.171  6.864  10.643    FALSE 1.000 1.000 13191
## deviance       95.390 6.024 84.969 94.919 108.449    FALSE 1.000 1.000  7955
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 18.1 and DIC = 113.529
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(blackcap_mt_age_inter$summary[,-11]),
   caption = "Blackcap $M_t$ inter-winter model diagnostic and summary output",
   label = "tab:blackcap_mt_age_inter_summary_output", type = "latex"),
```

```
      file = "report/model_outputs/blackcap_mt_age_inter.tex")

########## M_t chiffchaff model ############
chiffchaff_mt_age_inter <- jags_model_multinomial_age_inter(chiffchaff_inter,
                                inits = inits_mt_age_inter,
                                model_type = "mt",
                                hyper_par = list(
                                alpha_phi_juv = alpha_phi_juv_chiffchaff,
                                beta_phi_juv = beta_phi_juv_chiffchaff,
                                alpha_phi_adult = alpha_phi_adult_chiffchaff,
                                beta_phi_adult = beta_phi_adult_chiffchaff,
                                alpha_p = alpha_p,
                                beta_p = beta_p),
                                params = c("phi_juv", "phi_adult", "p",
                                           "phi_diff",
                                           "disc_obs_juv",
                                           "disc_rep_juv",
                                           "disc_obs_adult",
                                           "disc_rep_adult"))
```

```
##
## Processing function input.......
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
chiffchaff_mt_age_inter
```

```
## JAGS output for model 'multinomial_mt_age_inter.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.085 minutes at time 2023-08-07 23:33:26.
##
##                  mean    sd   2.5%    50%   97.5% overlap0     f  Rhat n.eff
## phi_juv[1]       0.198 0.134  0.026  0.167   0.535    FALSE 1.000 1.001  2605
## phi_juv[2]       0.217 0.124  0.046  0.193   0.524    FALSE 1.000 1.000  7394
## phi_juv[3]       0.413 0.171  0.116  0.403   0.764    FALSE 1.000 1.000 30000
## phi_juv[4]       0.173 0.105  0.036  0.150   0.438    FALSE 1.000 1.001  3735
## phi_juv[5]       0.387 0.166  0.108  0.376   0.737    FALSE 1.000 1.000 30000
## phi_juv[6]       0.184 0.109  0.037  0.163   0.458    FALSE 1.000 1.000 10721
## phi_juv[7]       0.331 0.141  0.109  0.312   0.648    FALSE 1.000 1.000 30000
```

```
## phi_juv[8]       0.375 0.147  0.133  0.361   0.695    FALSE 1.000 1.000 30000
## phi_juv[9]       0.306 0.167  0.052  0.283   0.677    FALSE 1.000 1.000 17076
## phi_juv[10]      0.376 0.157  0.117  0.359   0.722    FALSE 1.000 1.000 30000
## phi_adult[1]     0.287 0.164  0.046  0.263   0.658    FALSE 1.000 1.000 10500
## phi_adult[2]     0.224 0.129  0.046  0.200   0.533    FALSE 1.000 1.001  5053
## phi_adult[3]     0.515 0.156  0.220  0.513   0.814    FALSE 1.000 1.000 30000
## phi_adult[4]     0.475 0.151  0.204  0.468   0.781    FALSE 1.000 1.000 13198
## phi_adult[5]     0.306 0.145  0.086  0.285   0.638    FALSE 1.000 1.000 25143
## phi_adult[6]     0.460 0.154  0.184  0.453   0.774    FALSE 1.000 1.000  9215
## phi_adult[7]     0.295 0.135  0.090  0.274   0.613    FALSE 1.000 1.000  8885
## phi_adult[8]     0.341 0.144  0.112  0.324   0.664    FALSE 1.000 1.000 30000
## phi_adult[9]     0.489 0.156  0.204  0.485   0.797    FALSE 1.000 1.000  5049
## phi_adult[10]    0.384 0.156  0.128  0.368   0.725    FALSE 1.000 1.001  3282
## p[1]             0.103 0.094  0.007  0.076   0.357    FALSE 1.000 1.002  1980
## p[2]             0.236 0.138  0.050  0.208   0.577    FALSE 1.000 1.000 12163
## p[3]             0.149 0.091  0.026  0.131   0.373    FALSE 1.000 1.000  8601
## p[4]             0.283 0.122  0.098  0.266   0.565    FALSE 1.000 1.000 30000
## p[5]             0.153 0.099  0.024  0.131   0.400    FALSE 1.000 1.000  8445
## p[6]             0.282 0.128  0.088  0.262   0.578    FALSE 1.000 1.000  6582
## p[7]             0.290 0.126  0.099  0.271   0.586    FALSE 1.000 1.000 11302
## p[8]             0.312 0.128  0.112  0.294   0.608    FALSE 1.000 1.000  5215
## p[9]             0.130 0.084  0.021  0.112   0.341    FALSE 1.000 1.001 11755
## p[10]            0.327 0.136  0.117  0.308   0.640    FALSE 1.000 1.001  3870
## phi_diff[1]      0.089 0.202 -0.315  0.083   0.501     TRUE 0.678 1.000 30000
## phi_diff[2]      0.006 0.163 -0.325  0.005   0.342     TRUE 0.514 1.000  6836
## phi_diff[3]      0.102 0.229 -0.355  0.105   0.532     TRUE 0.672 1.000 30000
## phi_diff[4]      0.303 0.167 -0.019  0.300   0.634     TRUE 0.967 1.001  6675
## phi_diff[5]     -0.081 0.214 -0.496 -0.081   0.346     TRUE 0.650 1.000 30000
## phi_diff[6]      0.276 0.177 -0.066  0.273   0.623     TRUE 0.945 1.000 10936
## phi_diff[7]     -0.036 0.174 -0.387 -0.035   0.313     TRUE 0.585 1.000 27447
## phi_diff[8]     -0.033 0.190 -0.407 -0.033   0.344     TRUE 0.571 1.000 30000
## phi_diff[9]      0.183 0.223 -0.271  0.191   0.600     TRUE 0.793 1.000  7618
## phi_diff[10]     0.008 0.203 -0.392  0.008   0.411     TRUE 0.516 1.000  6480
## disc_obs_juv    10.162 2.039  6.748  9.968  14.645    FALSE 1.000 1.000 23087
## disc_rep_juv     7.719 1.853  4.648  7.551  11.819    FALSE 1.000 1.000 30000
## disc_obs_adult   8.298 1.187  6.354  8.169  10.972    FALSE 1.000 1.000 14383
## disc_rep_adult   7.803 1.848  4.645  7.645  11.828    FALSE 1.000 1.000  9550
## deviance       100.199 5.935 89.907 99.736 113.122    FALSE 1.000 1.000 10538
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 17.6 and DIC = 117.811
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(chiffchaff_mt_age_inter$summary[,-11]),
 caption = "Chiffchaff $M_t$ inter-winter model diagnostic and summary output",
 label = "tab:chiffchaff_mt_age_inter_summary_output", type = "latex"),
```

```
      file = "report/model_outputs/chiffchaff_mt_age_inter.tex")

########## M_t robin model ############
robin_mt_age_inter <- jags_model_multinomial_age_inter(robin_inter,
                                       inits = inits_mt_age_inter,
                                       model_type = "mt",
                                       hyper_par = list(
                                       alpha_phi_juv = alpha_phi_juv_robin,
                                       beta_phi_juv = beta_phi_juv_robin,
                                       alpha_phi_adult = alpha_phi_adult_robin,
                                       beta_phi_adult = beta_phi_adult_robin,
                                       alpha_p = alpha_p,
                                       beta_p = beta_p),
                                       params = c("phi_juv", "phi_adult",
                                                  "p", "phi_diff",
                                                  "disc_obs_juv",
                                                  "disc_rep_juv",
                                                  "disc_obs_adult",
                                                  "disc_rep_adult"))
```

```
##
## Processing function input.......
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
robin_mt_age_inter
```

```
## JAGS output for model 'multinomial_mt_age_inter.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.102 minutes at time 2023-08-07 23:33:32.
##
##                  mean    sd    2.5%     50%   97.5% overlap0     f  Rhat n.eff
## phi_juv[1]      0.729 0.120   0.484   0.735   0.936    FALSE 1.000 1.000 14777
## phi_juv[2]      0.453 0.136   0.222   0.441   0.748    FALSE 1.000 1.000  6627
## phi_juv[3]      0.762 0.123   0.487   0.776   0.953    FALSE 1.000 1.000 30000
## phi_juv[4]      0.505 0.172   0.197   0.498   0.847    FALSE 1.000 1.000  5147
## phi_juv[5]      0.474 0.178   0.163   0.463   0.836    FALSE 1.000 1.000 30000
## phi_juv[6]      0.291 0.155   0.068   0.266   0.660    FALSE 1.000 1.000  5529
## phi_juv[7]      0.697 0.139   0.406   0.705   0.935    FALSE 1.000 1.000 30000
```

```
## phi_juv[8]       0.564 0.173   0.233   0.563   0.887    FALSE 1.000 1.000  4299
## phi_juv[9]       0.647 0.171   0.283   0.662   0.927    FALSE 1.000 1.000  5031
## phi_juv[10]      0.465 0.192   0.131   0.455   0.850    FALSE 1.000 1.000  4759
## phi_adult[1]     0.763 0.121   0.493   0.777   0.954    FALSE 1.000 1.000 30000
## phi_adult[2]     0.774 0.108   0.542   0.784   0.951    FALSE 1.000 1.000 30000
## phi_adult[3]     0.774 0.108   0.546   0.783   0.951    FALSE 1.000 1.000 18845
## phi_adult[4]     0.735 0.124   0.471   0.745   0.942    FALSE 1.000 1.000  7650
## phi_adult[5]     0.692 0.136   0.412   0.698   0.929    FALSE 1.000 1.001  1439
## phi_adult[6]     0.568 0.158   0.282   0.562   0.876    FALSE 1.000 1.001  3566
## phi_adult[7]     0.518 0.165   0.223   0.509   0.851    FALSE 1.000 1.000 12649
## phi_adult[8]     0.587 0.160   0.289   0.583   0.891    FALSE 1.000 1.000 27761
## phi_adult[9]     0.623 0.159   0.311   0.626   0.909    FALSE 1.000 1.000 24875
## phi_adult[10]    0.637 0.164   0.306   0.646   0.920    FALSE 1.000 1.001  3193
## p[1]             0.065 0.028   0.022   0.061   0.132    FALSE 1.000 1.000 23710
## p[2]             0.037 0.018   0.011   0.034   0.080    FALSE 1.000 1.000 14043
## p[3]             0.175 0.045   0.099   0.171   0.277    FALSE 1.000 1.000 13940
## p[4]             0.049 0.022   0.016   0.045   0.103    FALSE 1.000 1.000 30000
## p[5]             0.119 0.044   0.053   0.112   0.221    FALSE 1.000 1.001  1392
## p[6]             0.166 0.068   0.067   0.155   0.330    FALSE 1.000 1.001  3526
## p[7]             0.119 0.049   0.046   0.111   0.233    FALSE 1.000 1.000 19900
## p[8]             0.116 0.055   0.037   0.107   0.248    FALSE 1.000 1.000 16172
## p[9]             0.236 0.097   0.089   0.222   0.461    FALSE 1.000 1.000  9708
## p[10]            0.211 0.096   0.073   0.194   0.444    FALSE 1.000 1.001  3311
## phi_diff[1]      0.034 0.168  -0.299   0.035   0.355     TRUE 0.585 1.000 24460
## phi_diff[2]      0.321 0.159  -0.008   0.328   0.614     TRUE 0.972 1.000  5475
## phi_diff[3]      0.012 0.161  -0.295   0.007   0.340     TRUE 0.518 1.000 30000
## phi_diff[4]      0.230 0.199  -0.162   0.233   0.605     TRUE 0.868 1.001  3649
## phi_diff[5]      0.217 0.213  -0.208   0.223   0.620     TRUE 0.840 1.000  6784
## phi_diff[6]      0.277 0.197  -0.128   0.279   0.651     TRUE 0.918 1.000  5366
## phi_diff[7]     -0.179 0.202  -0.552  -0.186   0.230     TRUE 0.808 1.000 13943
## phi_diff[8]      0.023 0.220  -0.399   0.021   0.457     TRUE 0.537 1.000  4236
## phi_diff[9]     -0.024 0.234  -0.466  -0.027   0.441     TRUE 0.545 1.000 26286
## phi_diff[10]     0.172 0.238  -0.301   0.176   0.619     TRUE 0.761 1.000 19536
## disc_obs_juv    21.207 2.429  16.925  21.048  26.463    FALSE 1.000 1.000 30000
## disc_rep_juv    15.462 2.756  10.579  15.290  21.321    FALSE 1.000 1.000 11865
## disc_obs_adult  15.978 1.973  12.547  15.829  20.279    FALSE 1.000 1.000 30000
## disc_rep_adult  14.393 2.403  10.087  14.249  19.539    FALSE 1.000 1.000 30000
## deviance       203.618 6.998 191.449 203.095 218.735    FALSE 1.000 1.000 21766
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 24.5 and DIC = 228.103
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(robin_mt_age_inter$summary[,-11]),
     caption = "Robin $M_t$ inter-winter model diagnostic and summary output",
     label = "tab:robin_mt_age_inter_summary_output", type = "latex"),
```

```
      file = "report/model_outputs/robin_mt_age_inter.tex")

##############################
# INTER-MODEL SENSITIVITY ANALYSIS
##############################

sensitivity_analysis_inter_model <- jags_model_multinomial_age_inter(robin_inter,
                                    inits = inits_mt_age_inter,
                                    model_type = "mt",
                                    hyper_par = list(
                                    alpha_phi_juv = alpha_phi_juv_robin + 1,
                                    beta_phi_juv = beta_phi_juv_robin + 1,
                                    alpha_phi_adult= alpha_phi_adult_robin + 1,
                                    beta_phi_adult= beta_phi_adult_robin + 1,
                                    alpha_p = alpha_p + 1,
                                    beta_p = beta_p + 1),
                                    params = c("phi_juv", "phi_adult", "p",
                                               "phi_diff",
                                               "disc_obs_juv",
                                               "disc_rep_juv",
                                               "disc_obs_adult",
                                               "disc_rep_adult"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
sensitivity_analysis_inter_model
```

```
## JAGS output for model 'multinomial_mt_age_inter.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.091 minutes at time 2023-08-07 23:33:39.
##
##                  mean    sd    2.5%     50%   97.5% overlap0     f  Rhat n.eff
## phi_juv[1]      0.684 0.113   0.458   0.688   0.893    FALSE 1.000 1.000 20731
## phi_juv[2]      0.430 0.123   0.220   0.420   0.697    FALSE 1.000 1.000 12840
## phi_juv[3]      0.718 0.120   0.460   0.729   0.919    FALSE 1.000 1.000 22395
## phi_juv[4]      0.477 0.152   0.202   0.470   0.789    FALSE 1.000 1.000 26083
```

```
## phi_juv[5]       0.451 0.156   0.177   0.442   0.777    FALSE 1.000 1.000 23511
## phi_juv[6]       0.297 0.134   0.088   0.279   0.605    FALSE 1.000 1.000 29570
## phi_juv[7]       0.641 0.132   0.379   0.645   0.882    FALSE 1.000 1.000  7792
## phi_juv[8]       0.526 0.156   0.236   0.524   0.831    FALSE 1.000 1.000 30000
## phi_juv[9]       0.614 0.157   0.289   0.625   0.885    FALSE 1.000 1.000 18313
## phi_juv[10]      0.448 0.168   0.153   0.438   0.787    FALSE 1.000 1.000 16308
## phi_adult[1]     0.719 0.119   0.465   0.728   0.919    FALSE 1.000 1.000 13486
## phi_adult[2]     0.734 0.104   0.518   0.740   0.915    FALSE 1.000 1.000 27351
## phi_adult[3]     0.745 0.103   0.531   0.751   0.922    FALSE 1.000 1.000 30000
## phi_adult[4]     0.694 0.115   0.462   0.698   0.902    FALSE 1.000 1.000  5381
## phi_adult[5]     0.665 0.125   0.417   0.669   0.891    FALSE 1.000 1.001  2962
## phi_adult[6]     0.552 0.139   0.299   0.547   0.832    FALSE 1.000 1.000  4021
## phi_adult[7]     0.503 0.144   0.245   0.496   0.799    FALSE 1.000 1.001  2738
## phi_adult[8]     0.556 0.142   0.293   0.553   0.831    FALSE 1.000 1.000 11569
## phi_adult[9]     0.601 0.143   0.324   0.603   0.870    FALSE 1.000 1.000 30000
## phi_adult[10]    0.602 0.147   0.309   0.606   0.870    FALSE 1.000 1.000  9901
## p[1]             0.079 0.031   0.030   0.075   0.151    FALSE 1.000 1.000  5387
## p[2]             0.048 0.021   0.016   0.045   0.099    FALSE 1.000 1.000 30000
## p[3]             0.199 0.048   0.117   0.195   0.304    FALSE 1.000 1.000 11295
## p[4]             0.064 0.027   0.024   0.060   0.127    FALSE 1.000 1.001  2751
## p[5]             0.146 0.049   0.068   0.140   0.259    FALSE 1.000 1.000  8222
## p[6]             0.196 0.071   0.087   0.186   0.361    FALSE 1.000 1.000  6055
## p[7]             0.145 0.054   0.061   0.137   0.270    FALSE 1.000 1.001  6519
## p[8]             0.147 0.061   0.054   0.139   0.292    FALSE 1.000 1.000 16005
## p[9]             0.286 0.102   0.121   0.273   0.518    FALSE 1.000 1.000 18830
## p[10]            0.258 0.102   0.100   0.244   0.495    FALSE 1.000 1.000 11607
## phi_diff[1]      0.034 0.160  -0.282   0.037   0.339     TRUE 0.587 1.000 13069
## phi_diff[2]      0.304 0.148  -0.002   0.310   0.574     TRUE 0.974 1.000 30000
## phi_diff[3]      0.027 0.156  -0.269   0.024   0.337     TRUE 0.560 1.000 14987
## phi_diff[4]      0.217 0.179  -0.144   0.221   0.555     TRUE 0.883 1.000 30000
## phi_diff[5]      0.213 0.191  -0.170   0.220   0.568     TRUE 0.863 1.000  6317
## phi_diff[6]      0.256 0.175  -0.096   0.258   0.592     TRUE 0.925 1.000 12644
## phi_diff[7]     -0.138 0.183  -0.479  -0.143   0.229     TRUE 0.775 1.000 19995
## phi_diff[8]      0.030 0.200  -0.355   0.029   0.420     TRUE 0.558 1.000 30000
## phi_diff[9]     -0.013 0.213  -0.417  -0.016   0.410     TRUE 0.530 1.000 30000
## phi_diff[10]     0.155 0.212  -0.269   0.159   0.550     TRUE 0.764 1.000 30000
## disc_obs_juv    23.078 2.596  18.451  22.914  28.654    FALSE 1.000 1.000 30000
## disc_rep_juv    15.268 2.786  10.261  15.105  21.211    FALSE 1.000 1.000  8520
## disc_obs_adult  16.792 2.079  13.149  16.642  21.299    FALSE 1.000 1.000 10204
## disc_rep_adult  14.308 2.440  10.005  14.140  19.511    FALSE 1.000 1.000 11256
## deviance       210.526 7.757 197.047 209.962 227.257    FALSE 1.000 1.000 15778
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 30.1 and DIC = 240.607
## DIC is an estimate of expected predictive error (lower is better).
```

```r
print(xtable(as.data.frame(sensitivity_analysis_inter_model$summary[,-11]),
     caption = "Sensitivity analysis check on robin inter-winter $M_t$ model",
     label = "tab:sensitivity_analysis_robin_age_inter", type = "latex"),
      file = "report/model_outputs/sensitivity_analysis_robin_age_inter.tex")

##############################
# INTRA-WINTER MODELLING
##############################
########## M_0 blackcap model ############
blackcap_m0_intra <- jags_model_multinomial_intra(blackcap_intra,
                                                  inits = inits_m0_intra,
                                                  model_type = "m0",
                                                  hyper_par = list(
                                                  alpha_phi = alpha_phi_juv_robin,
                                                  beta_phi = beta_phi_juv_robin,
                                                  alpha_p = alpha_p,
                                                  beta_p = beta_p),
                                                  params = c("mean_phi",
                                                             "mean_p",
                                                             "disc_obs",
                                                             "disc_rep"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```r
blackcap_m0_intra
```

```
## JAGS output for model 'multinomial_m0_intra.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.014 minutes at time 2023-08-07 23:33:45.
##
##             mean    sd   2.5%    50%   97.5% overlap0 f Rhat n.eff
## mean_phi   0.482 0.061  0.368  0.480   0.606    FALSE 1    1  6000
## mean_p     0.119 0.028  0.073  0.116   0.180    FALSE 1    1  7685
## disc_obs  22.729 1.572 20.441 22.458  26.430    FALSE 1    1 30000
## disc_rep   6.922 2.192  3.469  6.663  11.995    FALSE 1    1 30000
## deviance 95.262 1.924 93.388 94.677 100.402    FALSE 1    1 27506
```

```
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 1.9 and DIC = 97.113
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(blackcap_m0_intra$summary[,-11]),
   caption = "Blackcap $M_t$ intra-winter model diagnostic and summary output",
   label = "tab:blackcap_mt_intra", type = "latex"),
      file = "report/model_outputs/blackcap_mt_intra.tex")

########## M_0 chiffchaff model #############
chiffchaff_m0_intra <- jags_model_multinomial_intra(chiffchaff_intra,
                                                    inits = inits_m0_intra,
                                                    model_type = "m0",
                                                    hyper_par = list(
                                                    alpha_phi = alpha_phi_juv_robin,
                                                    beta_phi = beta_phi_juv_robin,
                                                    alpha_p = alpha_p,
                                                    beta_p = beta_p),
                                                    params = c("mean_phi",
                                                               "mean_p",
                                                               "disc_obs",
                                                               "disc_rep"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
chiffchaff_m0_intra
```

```
## JAGS output for model 'multinomial_m0_intra.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
```

```
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.017 minutes at time 2023-08-07 23:33:46.
##
##             mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## mean_phi   0.513 0.072  0.382  0.509  0.664    FALSE 1 1.004   654
## mean_p     0.077 0.022  0.041  0.074  0.127    FALSE 1 1.005   591
## disc_obs   5.862 0.847  4.974  5.622  8.071    FALSE 1 1.001  2482
## disc_rep   7.301 2.080  3.821  7.121 11.926    FALSE 1 1.000 27508
## deviance 52.731 1.931 50.844 52.130 57.959    FALSE 1 1.002  1827
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 1.9 and DIC = 54.594
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(chiffchaff_m0_intra$summary[,-11]),
 caption = "Chiffchaff $M_t$ intra-winter model diagnostic and summary output",
 label = "tab:blackcap_m0_intra", type = "latex"),
      file = "report/model_outputs/chiffchaff_mt_intra.tex")

########## M_0 robin model #############
robin_m0_intra <- jags_model_multinomial_intra(robin_intra,
                                               inits = inits_m0_intra,
                                               model_type = "m0",
                                               hyper_par = list(
                                               alpha_phi = alpha_phi_juv_robin,
                                               beta_phi = beta_phi_juv_robin,
                                               alpha_p = alpha_p,
                                               beta_p = beta_p),
                                               params = c("mean_phi",
                                                          "mean_p",
                                                          "disc_obs",
                                                          "disc_rep"))
```

```
##
## Processing function input.......
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
```

```
##
## Done.
```

```
print(xtable(as.data.frame(robin_m0_intra$summary[,-11]),
       caption = "Robin $M_t$ intra-winter model diagnostic and summary output",
       label = "tab:robin_mt_intra", type = "latex"),
      file = "report/model_outputs/robin_mt_intra.tex")

########## M_t blackcap model ############
blackcap_mt_intra <- jags_model_multinomial_intra(blackcap_intra,
                                                  inits = inits_mt_intra,
                                                  model_type = "mt",
                                                  hyper_par = list(
                                                  alpha_phi = alpha_phi_juv_blackcap,
                                                  beta_phi = beta_phi_juv_blackcap,
                                                  alpha_p = alpha_p,
                                                  beta_p = beta_p),
                                                  params = c("phi",
                                                             "p",
                                                             "disc_obs",
                                                             "disc_rep"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
blackcap_mt_intra
```

```
## JAGS output for model 'multinomial_mt_intra.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.025 minutes at time 2023-08-07 23:33:48.
##
##            mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## phi[1]    0.308 0.117  0.118  0.294  0.568    FALSE 1 1.000 30000
## phi[2]    0.468 0.116  0.260  0.462  0.708    FALSE 1 1.000 11620
## phi[3]    0.354 0.096  0.193  0.345  0.571    FALSE 1 1.001  4150
## phi[4]    0.478 0.111  0.281  0.471  0.711    FALSE 1 1.000  5319
## phi[5]    0.503 0.127  0.272  0.499  0.757    FALSE 1 1.000  9469
## phi[6]    0.334 0.147  0.097  0.318  0.651    FALSE 1 1.001  1505
```

```
## p[1]      0.051 0.047  0.003  0.038  0.174    FALSE 1 1.003 30000
## p[2]      0.178 0.064  0.082  0.168  0.328    FALSE 1 1.000 22539
## p[3]      0.145 0.056  0.062  0.137  0.279    FALSE 1 1.000 30000
## p[4]      0.251 0.074  0.132  0.242  0.423    FALSE 1 1.001  2425
## p[5]      0.190 0.066  0.093  0.179  0.349    FALSE 1 1.000  7241
## p[6]      0.106 0.074  0.025  0.085  0.305    FALSE 1 1.002  1244
## disc_obs 13.120 2.051  9.826 12.875 17.811    FALSE 1 1.000 30000
## disc_rep  6.295 1.978  3.126  6.057 10.795    FALSE 1 1.000 19664
## deviance 76.329 4.509 68.973 75.845 86.506    FALSE 1 1.000 19117
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 10.2 and DIC = 86.494
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(blackcap_mt_intra$summary[,-11]),
   caption = "Blackcap $M_t$ intra-winter model diagnostic and summary output",
   label = "tab:blackcap_m0_intra", type = "latex"),
      file = "report/model_outputs/blackcap_mt_intra.tex")

########## M_t chiffchaff model #############
chiffchaff_mt_intra <- jags_model_multinomial_intra(chiffchaff_intra,
                                          inits = inits_mt_intra,
                                          model_type = "mt",
                                          hyper_par = list(
                                          alpha_phi = alpha_phi_juv_chiffchaff,
                                          beta_phi = beta_phi_juv_chiffchaff,
                                          alpha_p = alpha_p,
                                          beta_p = beta_p),
                                          params = c("phi", "p",
                                                     "disc_obs",
                                                     "disc_rep"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
```

```
## Done.
```

```
chiffchaff_mt_intra
```

```
## JAGS output for model 'multinomial_mt_intra.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.024 minutes at time 2023-08-07 23:33:50.
##
##             mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## phi[1]    0.424 0.155  0.160  0.412  0.750    FALSE 1 1.000 30000
## phi[2]    0.415 0.124  0.210  0.403  0.690    FALSE 1 1.000  6457
## phi[3]    0.516 0.137  0.269  0.510  0.795    FALSE 1 1.001  2690
## phi[4]    0.315 0.127  0.126  0.294  0.617    FALSE 1 1.001  2185
## phi[5]    0.280 0.132  0.092  0.256  0.599    FALSE 1 1.000  7380
## phi[6]    0.294 0.160  0.065  0.267  0.671    FALSE 1 1.000 30000
## p[1]      0.171 0.098  0.039  0.151  0.416    FALSE 1 1.001  6778
## p[2]      0.106 0.044  0.044  0.097  0.214    FALSE 1 1.001  2908
## p[3]      0.068 0.029  0.027  0.062  0.138    FALSE 1 1.001  4978
## p[4]      0.210 0.092  0.077  0.195  0.427    FALSE 1 1.000 10715
## p[5]      0.353 0.157  0.113  0.330  0.710    FALSE 1 1.000 10566
## p[6]      0.209 0.137  0.037  0.175  0.566    FALSE 1 1.000 28273
## disc_obs  4.792 1.217  2.910  4.626  7.632    FALSE 1 1.000 30000
## disc_rep  5.853 1.988  2.754  5.596 10.441    FALSE 1 1.000 11150
## deviance 52.012 4.223 45.239 51.505 61.695    FALSE 1 1.000 30000
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 8.9 and DIC = 60.93
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(chiffchaff_mt_intra$summary[,-11]),
 caption = "Chiffchaff $M_t$ intra-winter model diagnostic and summary output",
 label = "tab:chiffchaff_mt_intra", type = "latex"),
      file = "report/model_outputs/chiffchaff_mt_intra.tex")

########## M_t robin model ############
robin_mt_intra <- jags_model_multinomial_intra(robin_intra,
                                               inits = inits_mt_intra,
                                               model_type = "mt",
                                               hyper_par = list(
                                               alpha_phi = alpha_phi_juv_robin,
                                               beta_phi = beta_phi_juv_robin,
                                               alpha_p = alpha_p,
                                               beta_p = beta_p),
                                               params = c("phi", "p",
```

```
                                                    "disc_obs",
                                                    "disc_rep"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
robin_mt_intra
```

```
## JAGS output for model 'multinomial_mt_intra.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.021 minutes at time 2023-08-07 23:33:52.
##
##             mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## phi[1]    0.557 0.184  0.209  0.559  0.894    FALSE 1 1.000 12734
## phi[2]    0.313 0.177  0.067  0.277  0.732    FALSE 1 1.000  5004
## phi[3]    0.308 0.177  0.065  0.273  0.730    FALSE 1 1.000  6363
## phi[4]    0.368 0.192  0.082  0.339  0.793    FALSE 1 1.000 30000
## phi[5]    0.400 0.203  0.079  0.376  0.829    FALSE 1 1.000 19507
## phi[6]    0.475 0.207  0.112  0.470  0.869    FALSE 1 1.000  6837
## p[1]      0.120 0.082  0.019  0.102  0.331    FALSE 1 1.001 13339
## p[2]      0.104 0.084  0.020  0.080  0.338    FALSE 1 1.001  3984
## p[3]      0.037 0.041  0.002  0.024  0.152    FALSE 1 1.000  6619
## p[4]      0.127 0.101  0.019  0.097  0.406    FALSE 1 1.000 30000
## p[5]      0.077 0.082  0.004  0.051  0.309    FALSE 1 1.001  6468
## p[6]      0.080 0.081  0.004  0.056  0.303    FALSE 1 1.001  2725
## disc_obs  5.635 2.070  2.421  5.360 10.394    FALSE 1 1.000 30000
## disc_rep  4.708 1.551  2.201  4.533  8.274    FALSE 1 1.000 10051
## deviance 24.435 4.636 17.151 23.846 35.121    FALSE 1 1.000 30000
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
```

```
## pD = 10.7 and DIC = 35.182
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(robin_mt_intra$summary[,-11]),
       caption = "Robin $M_t$ intra-winter model diagnostic and summary output",
       label = "tab:robin_m0_intra", type = "latex"),
      file = "report/model_outputs/robin_mt_intra.tex")

##############################
# INTRA-WINTER SENSITIVITY ANALYSYS
##############################
sensitivity_analysis_intra_model <- jags_model_multinomial_intra(robin_intra,
                                                 inits = inits_mt_intra,
                                                 model_type = "mt",
                                                 hyper_par = list(
                                                 alpha_phi = alpha_phi_juv_robin + 1,
                                                 beta_phi = beta_phi_juv_robin + 1,
                                                 alpha_p = alpha_p + 1,
                                                 beta_p = beta_p + 1),
                                                 params = c("phi", "p",
                                                            "disc_obs",
                                                            "disc_rep"))
```

```
##
## Processing function input.......
```

```
## Warning in gen.inits(inits, n.chains, seed, parallel): The 'seed' argument
## will be deprecated in the next version. You can set it yourself with set.seed()
## instead.
```

```
##
## Done.
##
## Beginning parallel processing using 3 cores. Console output will be suppressed.
##
## Parallel processing completed.
##
## Calculating statistics.......
##
## Done.
```

```
sensitivity_analysis_intra_model
```

```
## JAGS output for model 'multinomial_mt_intra.jags', generated by jagsUI.
## Estimates based on 3 chains of 10000 iterations,
## adaptation = 1000 iterations (sufficient),
## burn-in = 0 iterations and thin rate = 1,
## yielding 30000 total samples from the joint posterior.
## MCMC ran in parallel for 0.02 minutes at time 2023-08-07 23:33:53.
##
##            mean    sd   2.5%    50%  97.5% overlap0 f  Rhat n.eff
## phi[1]    0.505 0.163  0.203  0.500  0.820    FALSE 1 1.000  5269
## phi[2]    0.261 0.140  0.068  0.234  0.600    FALSE 1 1.000 15519
## phi[3]    0.254 0.141  0.063  0.226  0.597    FALSE 1 1.000 30000
## phi[4]    0.322 0.157  0.089  0.297  0.686    FALSE 1 1.001  2474
## phi[5]    0.346 0.166  0.091  0.324  0.720    FALSE 1 1.000  6790
```

```
## phi[6]    0.432 0.180  0.123  0.422  0.794    FALSE 1 1.000  6242
## p[1]      0.166 0.094  0.038  0.148  0.400    FALSE 1 1.000  6095
## p[2]      0.134 0.089  0.031  0.110  0.374    FALSE 1 1.001  3472
## p[3]      0.072 0.062  0.008  0.054  0.239    FALSE 1 1.002 12757
## p[4]      0.178 0.110  0.039  0.153  0.457    FALSE 1 1.001  1811
## p[5]      0.136 0.101  0.019  0.110  0.399    FALSE 1 1.000 10835
## p[6]      0.139 0.100  0.020  0.114  0.400    FALSE 1 1.000 10055
## disc_obs  8.024 2.433  4.023  7.762 13.468    FALSE 1 1.000 30000
## disc_rep  5.058 1.630  2.452  4.865  8.767    FALSE 1 1.000 22048
## deviance 29.797 5.493 20.718 29.226 42.027    FALSE 1 1.000 30000
##
## Successful convergence based on Rhat values (all < 1.1).
## Rhat is the potential scale reduction factor (at convergence, Rhat=1).
## For each parameter, n.eff is a crude measure of effective sample size.
##
## overlap0 checks if 0 falls in the parameter's 95% credible interval.
## f is the proportion of the posterior with the same sign as the mean;
## i.e., our confidence that the parameter is positive or negative.
##
## DIC info: (pD = var(deviance)/2)
## pD = 15.1 and DIC = 44.886
## DIC is an estimate of expected predictive error (lower is better).
```

```
print(xtable(as.data.frame(sensitivity_analysis_intra_model$summary[,-11]),
       caption = "Sensitivity analysis check on robin intra-winter $M_t$ model",
       label = "tab:sensitivity_analysis_robin_age_intra", type = "latex"),
      file = "report/model_outputs/sensitivity_analysis_robin_intra.tex")

################################
# INTER-WINTER MODELLING CHECKS
################################

# Create a data frame for model comparison between three species, storing the
# DIC values for two models (M_0 and M_t)
model_comparison <- data.frame(
  Species = c("Blackcap", "Chiffchaff", "Robin"),
  M_0 = c(blackcap_m0_age_inter$DIC,
          chiffchaff_m0_age_inter$DIC,
          robin_m0_age_inter$DIC),
  M_t = c(blackcap_mt_age_inter$DIC,
          chiffchaff_mt_age_inter$DIC,
          robin_mt_age_inter$DIC)
)
model_comparison
```

```
##      Species      M_0      M_t
## 1   Blackcap 118.1224 113.5294
## 2 Chiffchaff 116.4563 117.8106
## 3      Robin 231.6439 228.1033
```

```
print(xtable(model_comparison, type = "latex",
             caption = "Inter-winter model comparison",
             label = "tab:model_comparison_inter"),
      file = "report/model_outputs/model_comparison_inter.tex")
```

```r
# Function to calculate Bayesian p-values
calc_pvalue <- function(model, suffix = ""){
  return(sum(model$sims.list[[paste0("disc_obs", suffix)]] >
               model$sims.list[[paste0("disc_rep", suffix)]]) /
           length(model$sims.list[[paste0("disc_rep", suffix)]]))
}

# Creating a dataframe of all p-values
p_values <- data.frame(
  species = c("Blackcap", "Chiffchaff", "Robin"),
  juveniles_m0 = c(calc_pvalue(blackcap_m0_age_inter, "_juv"),
                   calc_pvalue(chiffchaff_m0_age_inter, "_juv"),
                   calc_pvalue(robin_m0_age_inter, "_juv")),
  adults_m0 = c(calc_pvalue(blackcap_m0_age_inter, "_adult"),
                calc_pvalue(chiffchaff_m0_age_inter, "_adult"),
                calc_pvalue(robin_m0_age_inter, "_adult")),
  juveniles_mt = c(calc_pvalue(blackcap_mt_age_inter, "_juv"),
                   calc_pvalue(chiffchaff_mt_age_inter, "_juv"),
                   calc_pvalue(robin_mt_age_inter, "_juv")),
  adults_mt = c(calc_pvalue(blackcap_mt_age_inter, "_adult"),
                calc_pvalue(chiffchaff_mt_age_inter, "_adult"),
                calc_pvalue(robin_mt_age_inter, "_adult"))
)

## Add pooled p-values
p_values <- cbind(p_values[1:3],
                  overall_m0 = rowMeans(p_values[, c("juveniles_m0",
                                                     "adults_m0")]),
                  p_values[4:5],
                  overall_mt = rowMeans(p_values[, c("juveniles_mt",
                                                     "adults_mt")]))

p_values
```

```
##      species juveniles_m0 adults_m0 overall_m0 juveniles_mt adults_mt
## 1   Blackcap    0.9974667 0.8308333  0.9141500    0.6962667 0.8597667
## 2 Chiffchaff    0.9824333 0.6341333  0.8082833    0.8472000 0.6100667
## 3      Robin    0.9985333 0.8107333  0.9046333    0.9517333 0.7230667
##   overall_mt
## 1  0.7780167
## 2  0.7286333
## 3  0.8374000
```

```r
# Creating a LaTeX table using xtable and saving it to a file
latex_pvalues <- xtable(p_values,
      caption="Inter-winter model goodness of fit results (Bayesian p-values)",
      label="tab:gof_results_inter")
print(latex_pvalues, type="latex",
      file="report/model_outputs/gof_results_age_inter.tex")

##############################
# INTRA-WINTER MODEL CHECKS
##############################
```

```r
# Create a data frame for model comparison between three species, storing the
# DIC values for two models (M_0 and M_t)
model_comparison_intra <- data.frame(
  Species = c("Blackcap", "Chiffchaff", "Robin"),
  M_0 = c(blackcap_m0_intra$DIC, chiffchaff_m0_intra$DIC,
          robin_m0_intra$DIC),
  M_t = c(blackcap_mt_intra$DIC, chiffchaff_mt_intra$DIC,
          robin_mt_intra$DIC)
)
model_comparison_intra
```

```
##      Species      M_0      M_t
## 1   Blackcap 97.11257 86.49449
## 2 Chiffchaff 54.59430 60.92969
## 3      Robin 25.36545 35.18216
```

```r
print(xtable(model_comparison_intra, type = "latex",
             caption = "Intra-winter model comparisons",
             label = "tab:model_comparison_intra"),
      file = "report/model_outputs/model_comparison_intra.tex")

# Creating a dataframe of all Bayesian p-values
p_values_intra <- data.frame(
  species = c("Blackcap", "Chiffchaff", "Robin"),

  m0 = c(calc_pvalue(blackcap_m0_intra, ""),
         calc_pvalue(chiffchaff_m0_intra, ""),
         calc_pvalue(robin_m0_intra, "")),

  mt = c(calc_pvalue(blackcap_mt_intra, ""),
         calc_pvalue(chiffchaff_mt_intra, ""),
         calc_pvalue(robin_mt_intra, ""))
)
p_values_intra
```

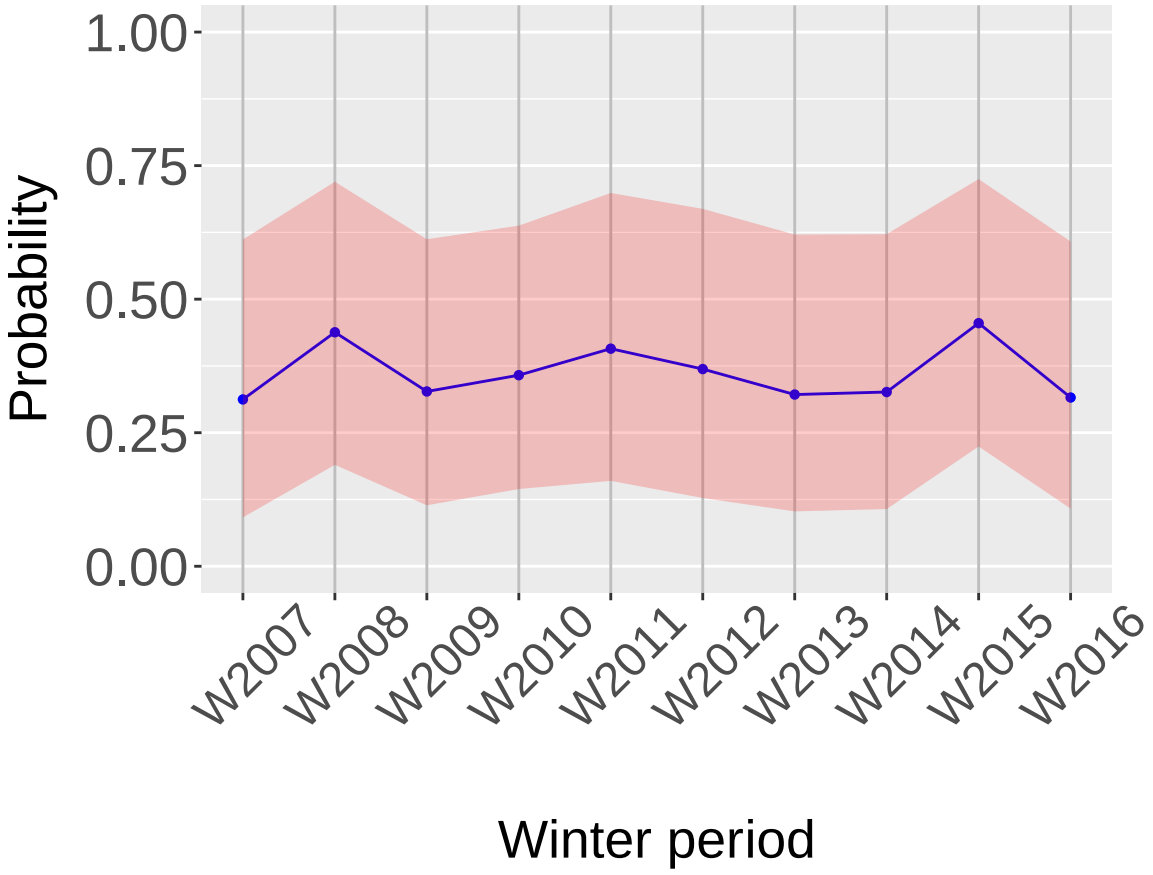
```
##      species        m0        mt
## 1   Blackcap 1.0000000 0.9922667
## 2 Chiffchaff 0.2572000 0.3307333
## 3      Robin 0.7254333 0.6540333
```

```r
latex_pvalues_intra <- xtable(p_values_intra,
      caption="Intra-winter model goodness of fit results (Bayesian p-values)",
      label="tab:gof_results_intra")
print(latex_pvalues_intra, type="latex",
      file="report/model_outputs/gof_results_intra.tex")

##############################
# INTER-WINTER MODEL RESULTS
##############################

species_list <- list(blackcap_mt_age_inter, chiffchaff_mt_age_inter,
                     robin_mt_age_inter)
species_names <- c("Blackcap", "Chiffchaff", "Robin")

# Path to save the plots
```

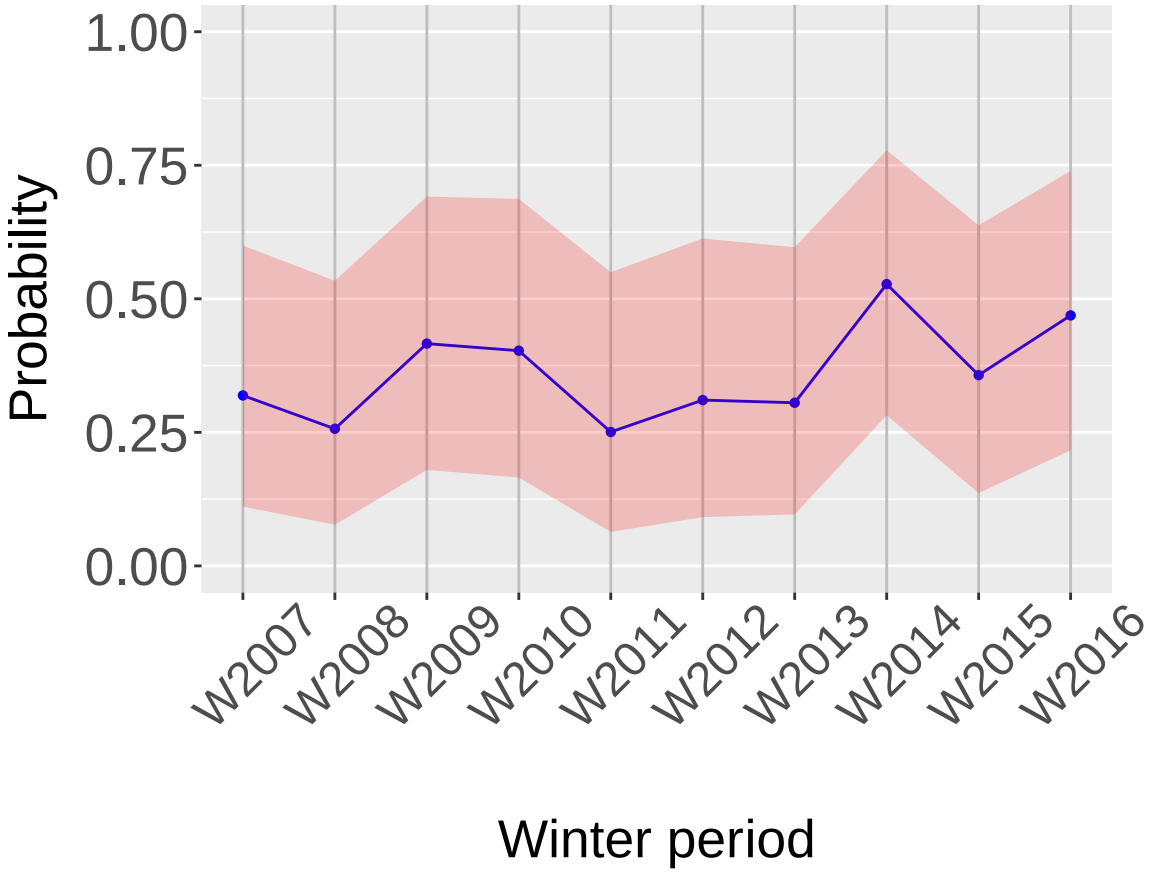
```r
path_to_save <- "report/images/"
# Iterate through the first three species in the list
for (i in 1:3) {
  # Extract the result for the current species
  model_result <- species_list[[i]]

  # Determine the number of time points in the adult survival probability
  T <- length(model_result$mean$phi_adult)

  # Initialize vectors to store lower and upper quantiles for adults,
  # juveniles, and their differences
  lower_adult <- upper_adult <- lower_juv <- upper_juv <- lower_diff <- upper_diff <- numeric(T)

  # Calculate 2.5% and 97.5% quantiles for adult, juvenile, and difference
  # in survival probabilities
  for (t in 1:T) {
    lower_adult[t] <- quantile(model_result$sims.list$phi_adult[,t], 0.025)
    upper_adult[t] <- quantile(model_result$sims.list$phi_adult[,t], 0.975)
    lower_juv[t] <- quantile(model_result$sims.list$phi_juv[,t], 0.025)
    upper_juv[t] <- quantile(model_result$sims.list$phi_juv[,t], 0.975)
    lower_diff[t] <- quantile(model_result$sims.list$phi_diff[,t], 0.025)
    upper_diff[t] <- quantile(model_result$sims.list$phi_diff[,t], 0.975)
  }

  # Define the years for the x-axis
  years <- seq(2007, 2016)

  # Iterate through the categories (adult, juvenile, difference) to plot
  # the survival probabilities
  for (j in c("adult", "juv", "diff")) {
    # Extract y-values and corresponding lower and upper bounds
    y_values <- model_result$mean[[paste0("phi_", j)]]
    lower_values <- get(paste0("lower_", j))
    upper_values <- get(paste0("upper_", j))

    # Create a data frame to store the values for plotting
    df <- data.frame(Year = years, Value = y_values, Lower = lower_values,
                     Upper = upper_values)

    # Create the plot using ggplot2
    p <- ggplot(df, aes(x = Year, y = Value)) +
      geom_line(color = "blue") + # Line plot
      geom_point(shape = 16, color = "blue") + # Points
      geom_ribbon(aes(ymin = Lower, ymax = Upper),
                  fill = "red", alpha = 0.2) + # Confidence band
      (if (j == "diff") ylim(-1, 1) else ylim(0, 1)) + # Set y limits
      labs(y = "Probability", x = "") + # Labels
      custom_theme +
      theme(axis.text.x = element_text(angle = 45,
                                       margin = margin(t = 20),
                                       size = size * (3.5/4)),
            panel.grid.major.x = element_line(color = "grey"),
            panel.grid.minor.x = element_blank(),
```

```
            plot.margin = unit(c(0, 0.75, 0, 0),
                               "inches")) +
      scale_x_continuous(breaks = years, labels = function(x) paste0("W", x)) +
      xlab("Winter period")


    print(p)
    ggsave(filename = paste0(path_to_save, species_names[i], "_", j,
                             "_inter.png"), plot = p)
  }
}
```

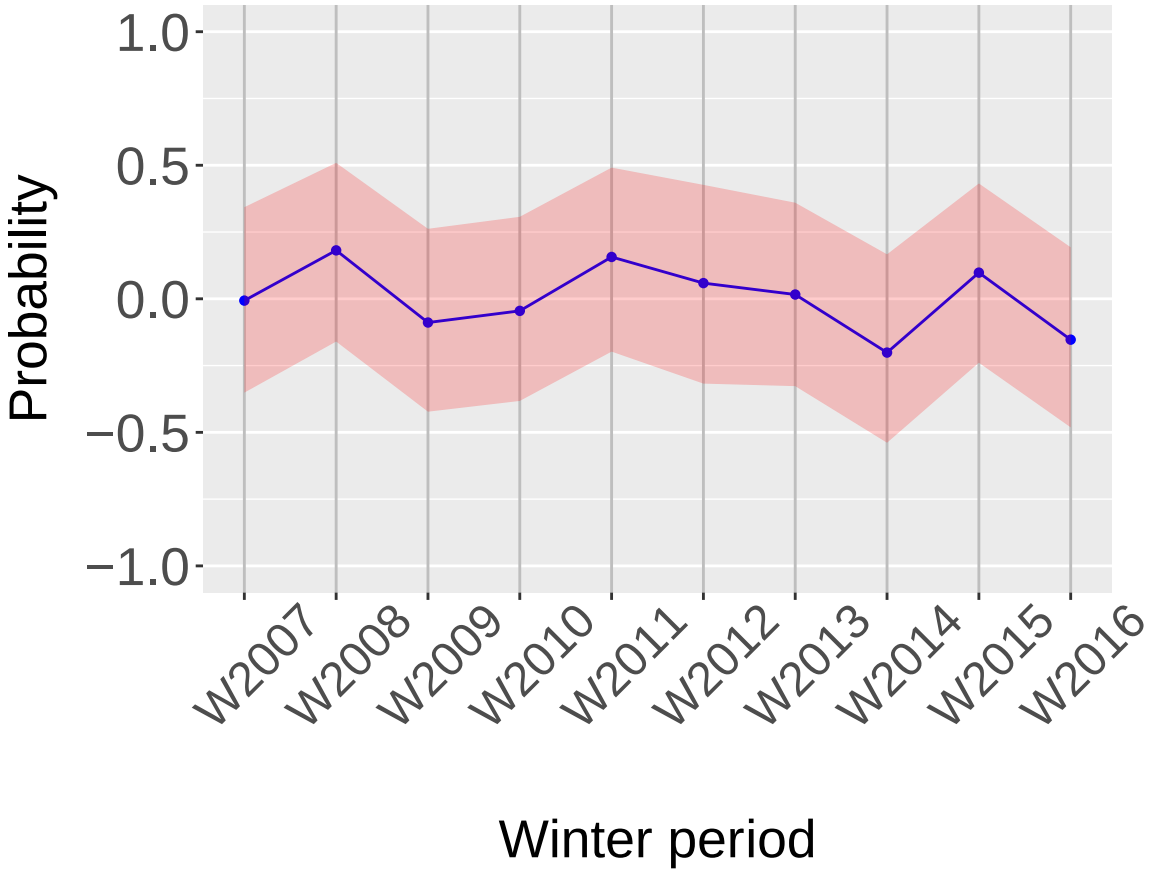
```
## Saving 6.5 x 4.5 in image
```



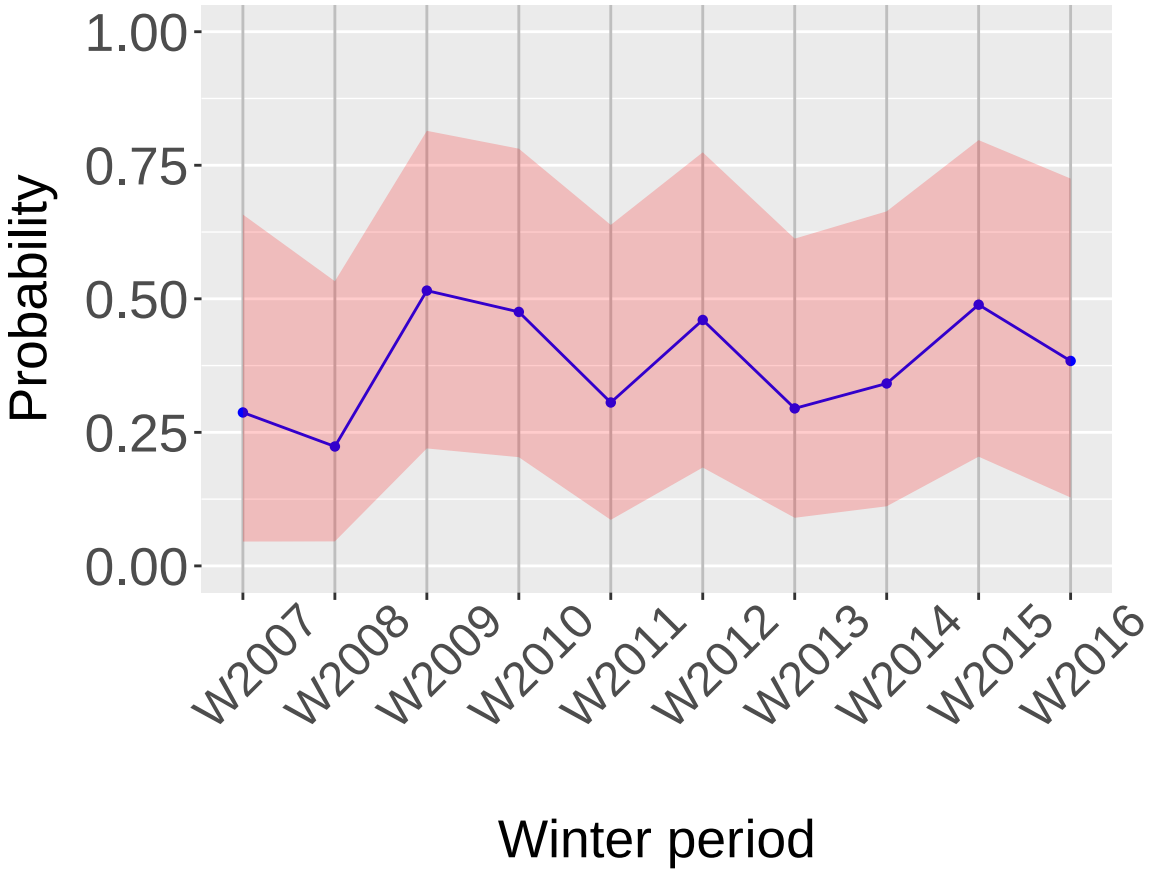

```
## Saving 6.5 x 4.5 in image
```



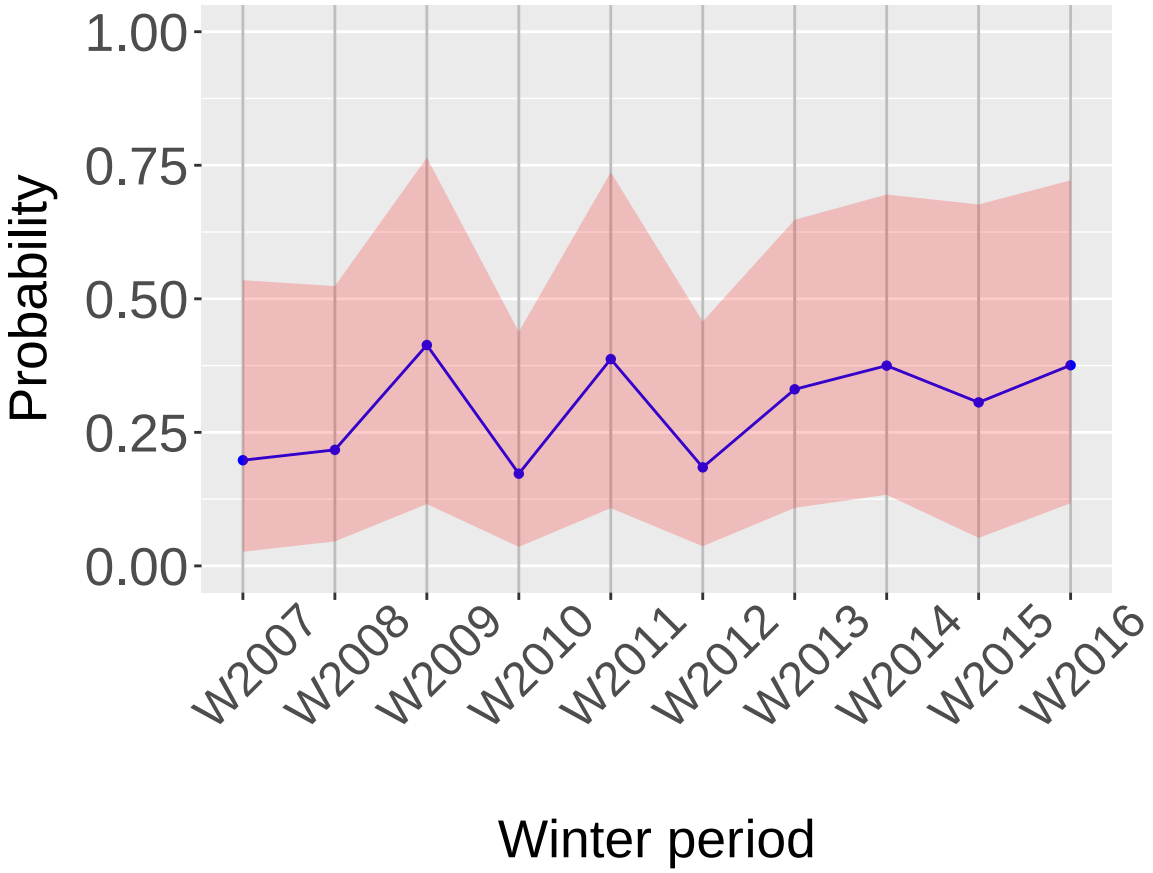

```
## Saving 6.5 x 4.5 in image
```



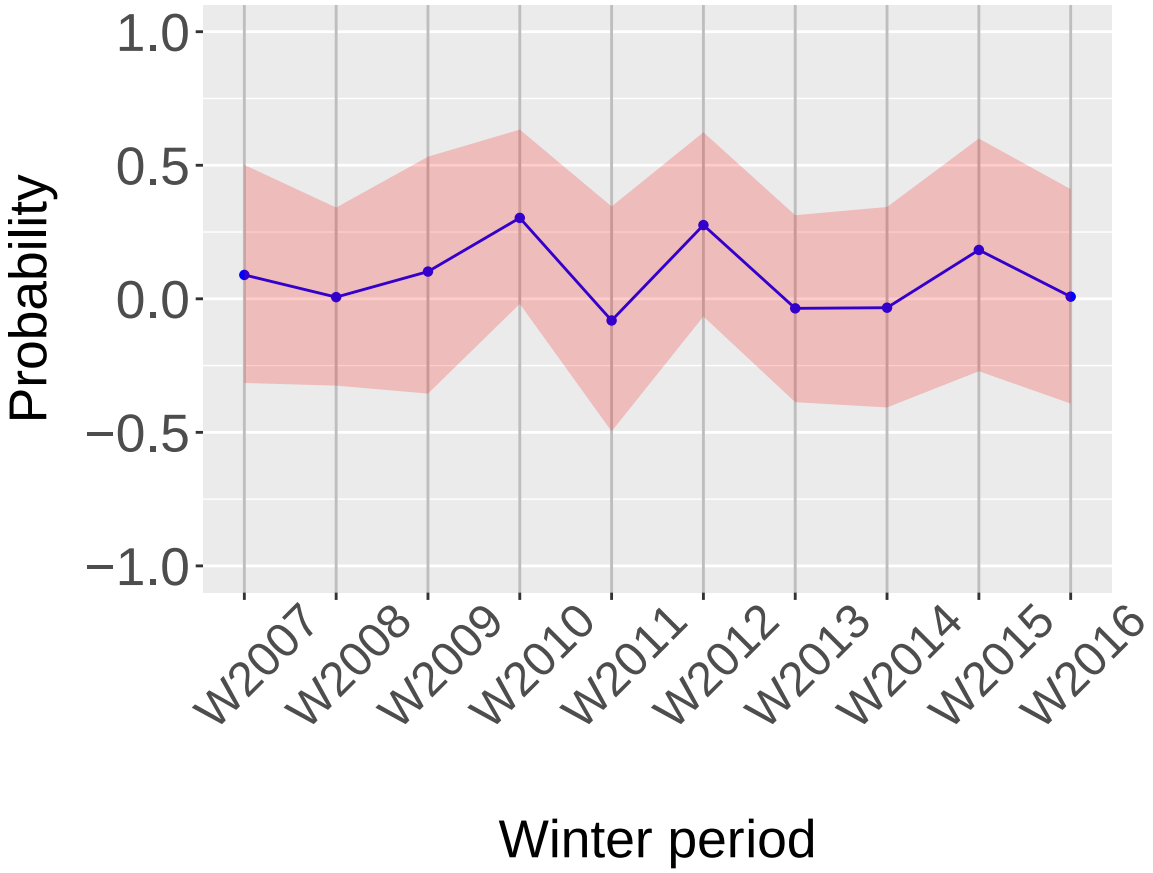

```
## Saving 6.5 x 4.5 in image
```



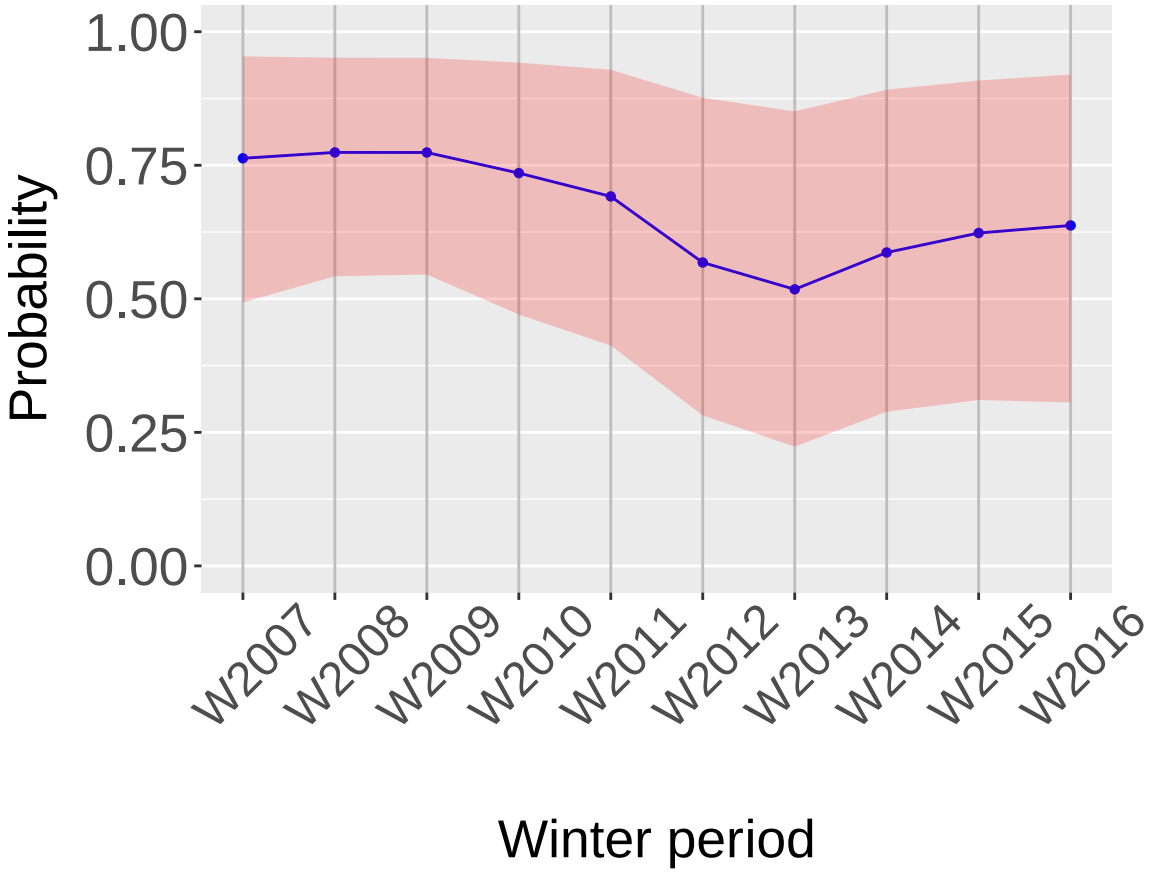

```
## Saving 6.5 x 4.5 in image
```



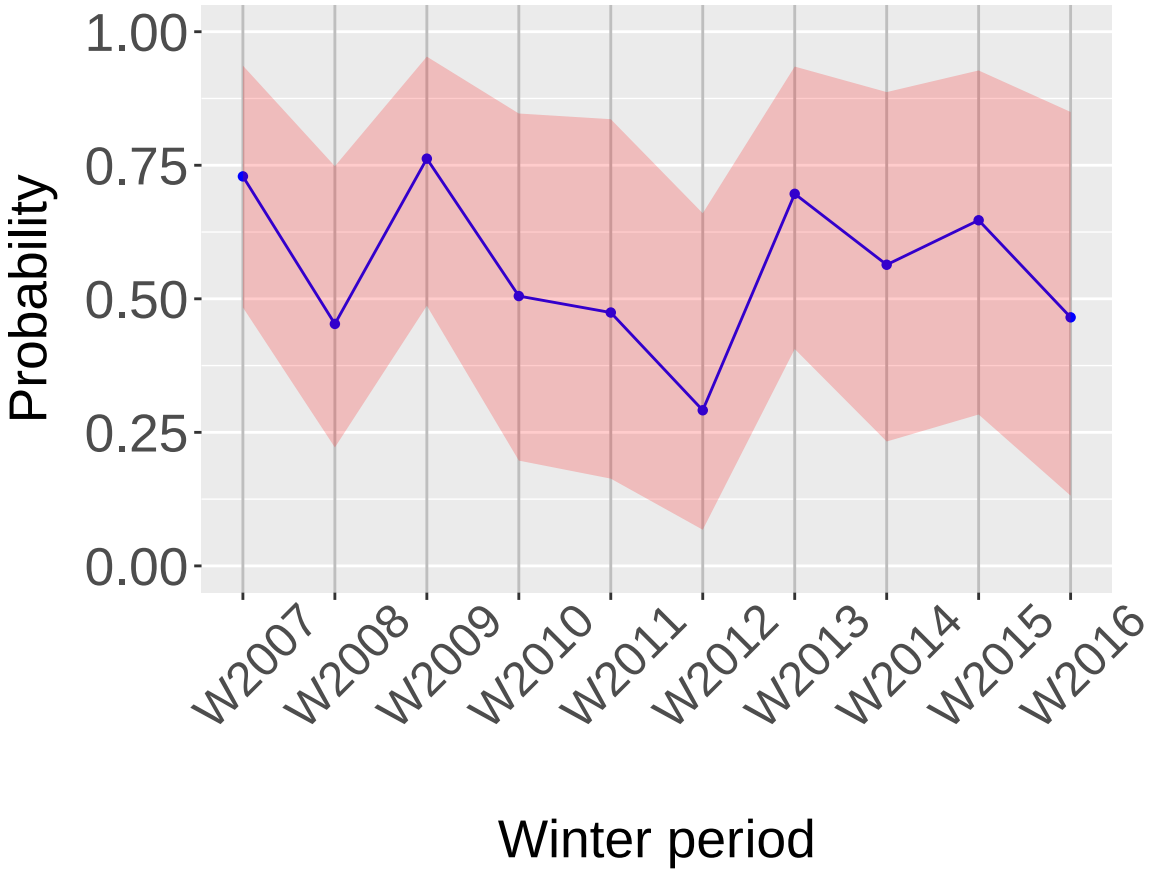

```
## Saving 6.5 x 4.5 in image
```



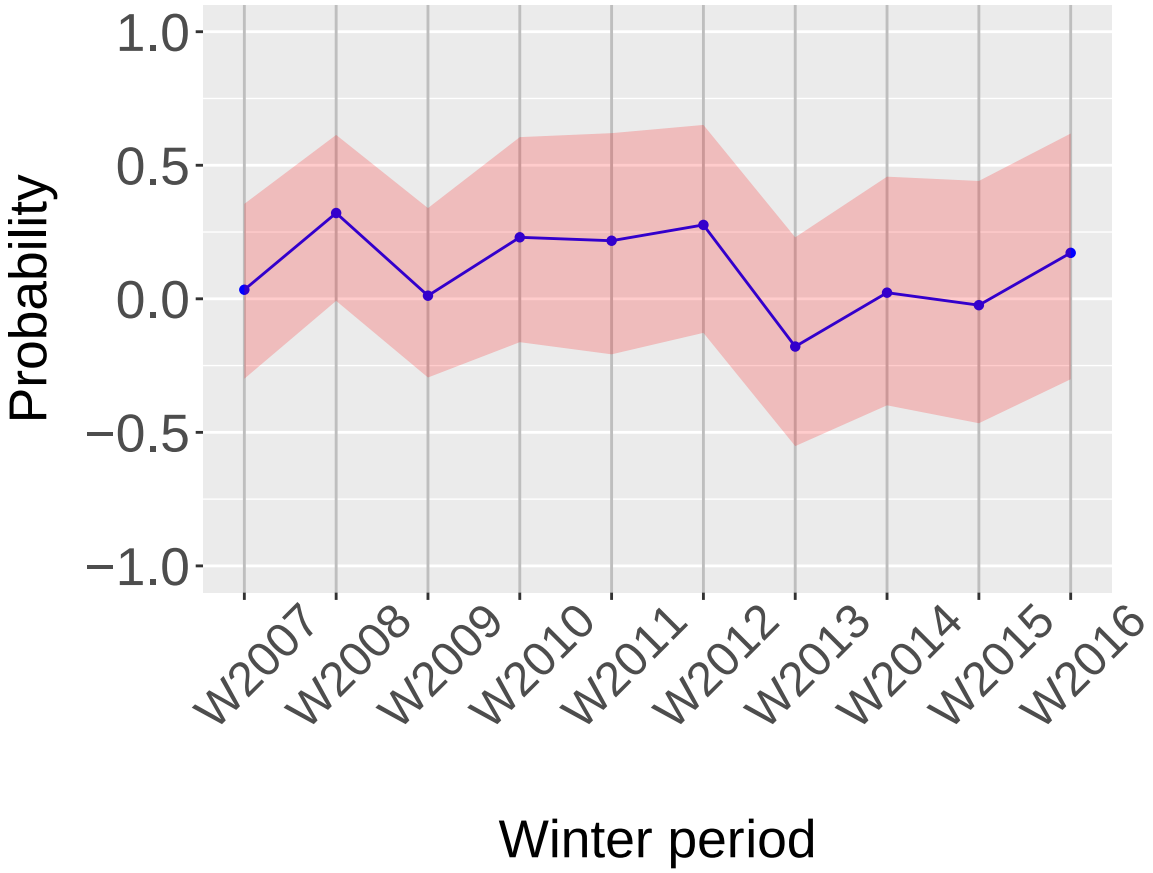

```
## Saving 6.5 x 4.5 in image
```




```
## Saving 6.5 x 4.5 in image
```

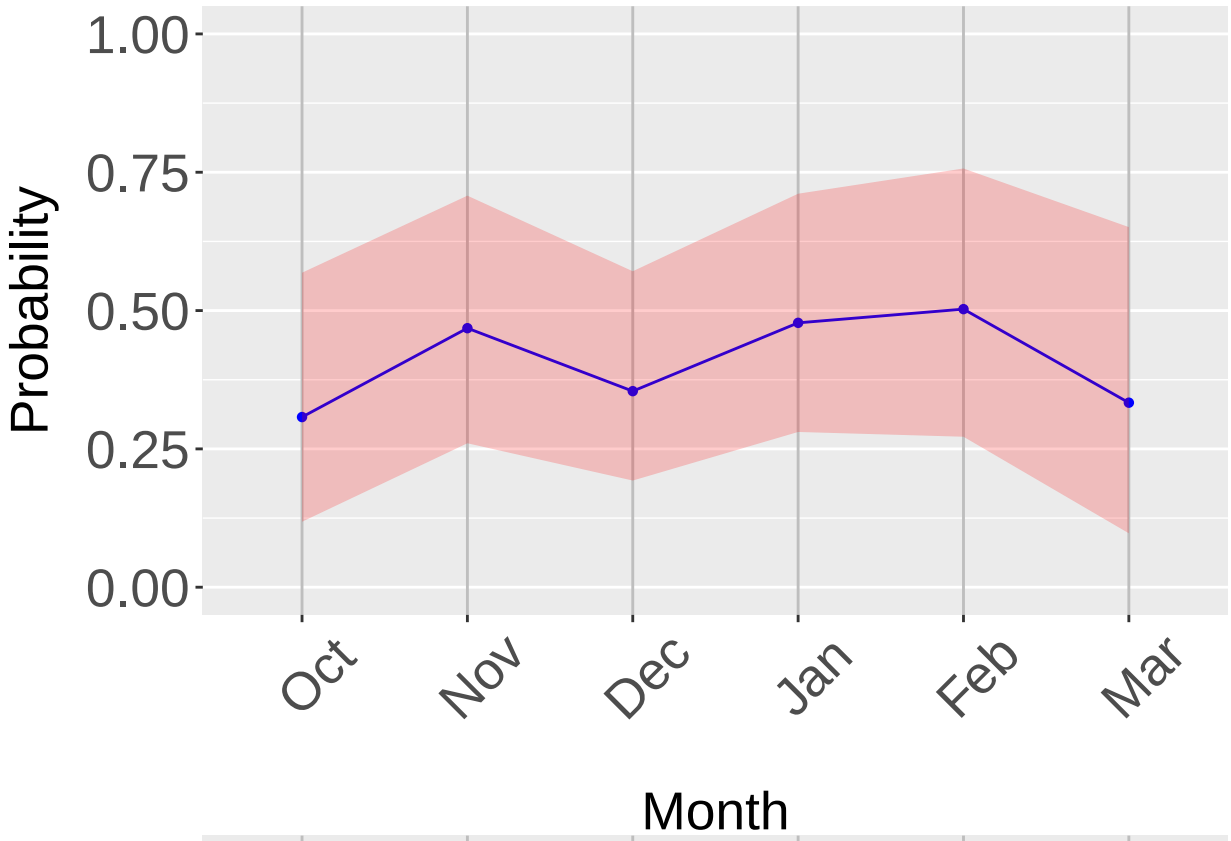


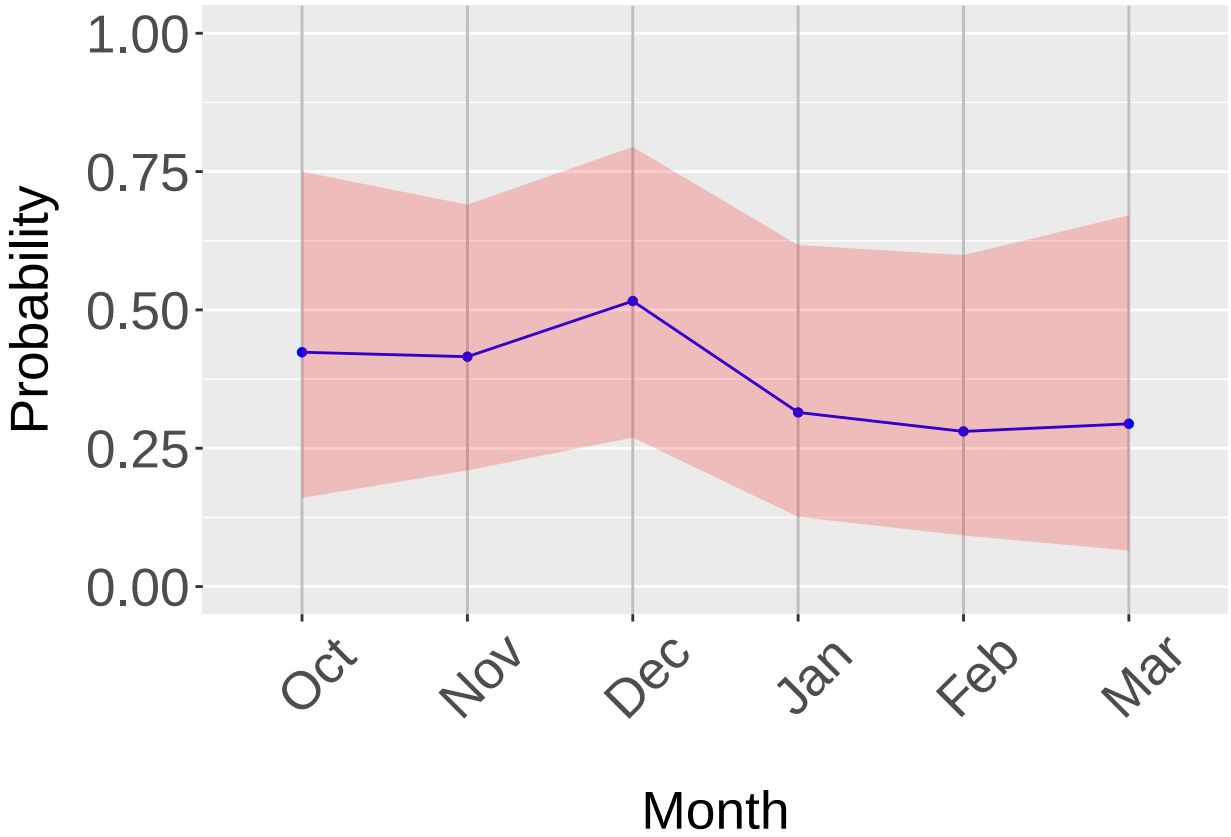

```
## Saving 6.5 x 4.5 in image
```

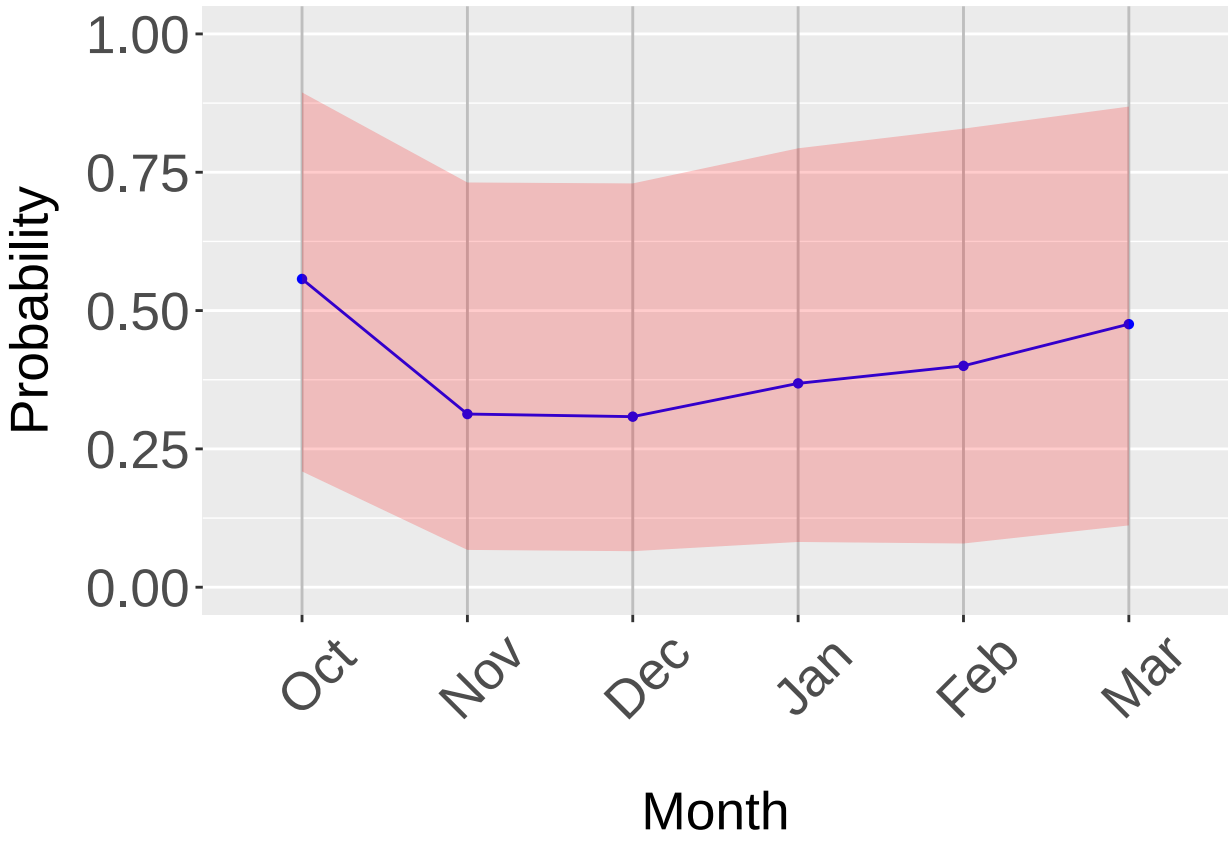



```
##############################
# INTRA-WINTER MODEL RESULTS
##############################

species_list_intra <- list(blackcap_mt_intra, chiffchaff_mt_intra,
                           robin_mt_intra)
# Iterate through the first three species in the list
for (i in 1:3) {
  # Extract the result for the current species
  model_result <- species_list_intra[[i]]

  # Determine the number of time points in the survival probability
  T <- length(model_result$mean$phi)

  # Initialise vectors to store lower and upper quantiles for survival
  # probabilities
  lower_phi <- upper_phi <- numeric(T)

  # Calculate 2.5% and 97.5% quantiles for survival probabilities
  for (t in 1:T) {
    lower_phi[t] <- quantile(model_result$sims.list$phi[,t], 0.025)
    upper_phi[t] <- quantile(model_result$sims.list$phi[,t], 0.975)
  }

  y_values <- model_result$mean$phi
```

```r
  # Define the months for the x-axis
  months <- c("Oct", "Nov", "Dec", "Jan", "Feb", "Mar")

  # Create a data frame to store the values for plotting
  df <- data.frame(Month = months, Value = y_values, Lower = lower_phi,
                   Upper = upper_phi)

  # Order the months factor to follow the given sequence
  df$Month <- factor(df$Month, levels = months)

  p <- ggplot(df, aes(x = Month, y = Value, group = 1)) +
    geom_line(color = "blue") +
    geom_point(shape = 16, color = "blue") +
    geom_ribbon(aes(ymin = Lower, ymax = Upper),
                fill = "red", alpha = 0.2) + # Confidence band
    ylim(0, 1) +
    labs(y = "Probability", x = "") +
    xlab("Month") +
    custom_theme +
    theme(axis.text.x = element_text(angle = 45,
                                     margin = margin(t = 15, r = 20),
                                     size = size),
          panel.grid.major.x = element_line(color = "grey"),
          panel.grid.minor.x = element_blank())

  print(p)

  ggsave(filename = paste0(path_to_save, species_names[i], "_", "intra.png"),
         plot = p, width = 10, height = 8)
}
```


1.00
0.75
0.50
0.25
0.00
Probability
Oct
Nov
Dec
Jan
Feb
Mar
Month


1.00
0.75
0.50
0.25
0.00
Probability
Oct
Nov
Dec
Jan
Feb
Mar
Month


1.00
0.75
0.50
0.25
0.00
Probability
Oct
Nov
Dec
Jan
Feb
Mar
Month