R-code for "Sex-specific early survival drives adult sex ratio bias in snowy plovers and impacts mating system and population growth"

Luke J. Eberhart-Phillips, Clemens Küpper, Tom E. X. Miller, Medardo Cruz-López, Kathryn H. Maher, Natalie dos Remedios, Martin A. Stoffel, Joseph I. Hoffman, Oliver Krüger, Tamás Székely

June 7, 2017

In this document we provide all the necessary code for reproducing the analyses presented in our paper. To access the dataset and Rmarkdown file, please download this GitHub repository. Simply follow the link and click on *Download ZIP* on the right-hand side of the page. An explanation of the files in the repository can be found in the Readme file. Please don't hesitate to contact Luke at `luke.eberhart[at]gmail.com` if you have any questions.

The structure of the code we present here follows the analyses presented in the *Results* section of the paper.

Prerequisites:

- For running the complete code you need a `files` subfolder containing the raw data downloaded from **`data`** and **`output/bootstrap`** folders provided in the GitHub repository.
- The following packages are needed for analysis and can be easily installed from CRAN by uncommenting the `install.packages` functions:

```
# install.packages("RMark")
# install.packages("stringr")
# install.packages("ggplot2")
# install.packages("dplyr")
# install.packages("grid")
# install.packages("gridExtra")
# install.packages("reshape2")
# install.packages("RColorBrewer")
# install.packages("Rmisc")
# install.packages("stats")
# install.packages("lme4")
# install.packages("magrittr")
library(RMark)
library(stringr)
library(ggplot2)
library(dplyr)
library(gridExtra)
library(grid)
library(reshape2)
library(RColorBrewer)
library(Rmisc)
library(stats)
library(lme4)
library(magrittr)
```

Loading data

To start, please load the following datasets into your R environment:

- **data/chick_mark-recapture_data.txt** contains the mark-recapture field data of chicks. Each row is a single uniquely marked chick identified by their *ring*. The daily encounter history of an individual is expressed in their *ch*, where a "1" indicates that an individual was encountered, "0" indicates it was not encountered, and "." indicates that no survey took place on that day. *year* indicates the year during which an individual was monitored and *day_of_season* indicates the number of days since the start of the breeding season that an individual hatched. *sex* describes the molecular sex-type of an individual with "M" for males and "F" for females. *brood_ID* is a unique brood identifier for the family from which a chick hatched.
- **data/juvenile_adult_mark-recapture_data.txt** contains the mark-recapture field data of juveniles and adults. Each row is a single uniquely marked individual identified by their *ring*. The annual encounter history of an individual is expressed in their *ch*, where a "1" indicates that an individual was encountered and "0" indicates it was not encountered. *sex* describes the molecular sex-type of an individual with "M" for males and "F" for females. *stage* describes the stage at which an individual was initially captured, where "J" indicates it was first captured as a chick, and "A" indicates it was first captured as an adult.
- **data/breeding_data.txt** contains the individual reproductive histories of all marked breeding adults in the population. Each row is a nesting attempt uniquely identified by the nest *ID*. *no_chicks* expresses the number of chicks that hatched from the nest. *clutch_size* indicates the number of eggs in the nest when it was initially discovered. *year* describes the year in which the nest was active. *male* and *female* indicates the unique identity of the father and mother, respectively, with "male_NA" and "female_NA" describing cases in which the other mate was not identified.

```
setwd("~/Dropbox/Luke/R_projects/Ceuta_ASR_Matrix_Modeling")
chick <-
  read.table("data/chick_mark-recapture_data.txt",
             header = TRUE, colClasses = c("factor", "character","factor",
                                           "numeric","factor","factor", "numeric"))

juvenile_adult <-
  read.table("data/juvenile_adult_mark-recapture_data.txt",
             header = TRUE, colClasses = c("factor","character","factor","factor"))

breeding_data <-
  read.table("data/breeding_data.txt",
             header = TRUE)
```

Quantifying mating system

To put our estimate of ASR in the context of breeding behavior, we quantified sex bias in mating system based on behavioral obersvations from the field. Females of this species desert broods to seek serial mates (Page et al. 2009). Thus, we expected that females would have more mates per year than males.

Step one: wrangle the data remove any cases in which one mate was not identified (i.e., "NA")

```
mating_df <-
  breeding_data[which(!is.na(breeding_data$female) & !is.na(breeding_data$male)),]
```

determine the number of families used in the mating system analysis (i.e. the sample size)

```
length(unique(mating_df$brood_ID))
#> [1] 456
```

bind the two mates together to make a unique pair

```
mating_df$pair <- as.factor(paste(mating_df$female, mating_df$male, sep = "-"))
```

determine how many mating attempts each individual had each year

```
females <- reshape2::dcast(mating_df, female  ~ year)
males <- reshape2::dcast(mating_df, male  ~ year)
```

determine how many different mates each individual had over their lifetime in the popualtion

```
number_males_p_female <-
  stats::aggregate(male ~ female, mating_df, function(x) length(unique(x)))
number_females_p_male <-
  stats::aggregate(female ~ male, mating_df, function(x) length(unique(x)))
```

join these two dataframes together and define as numeric

```
females <- dplyr::inner_join(females, number_males_p_female)
females[,c(2:8)] <-
  lapply(females[,c(2:8)], as.numeric)
males <- dplyr::inner_join(males, number_females_p_male)
males[,c(2:8)] <-
  lapply(males[,c(2:8)], as.numeric)
```

calculate the total number of mating attempts over each individual's lifetime

```
females$attempts <- rowSums(females[, c(2:8)])
males$attempts <- rowSums(males[, c(2:8)])
```

calculate the number of years breeding

```
females$years <- rowSums(females[, c(2:8)] > 0)
males$years <- rowSums(males[, c(2:8)] > 0)
```

filter out all individuals that only had one mating attempt

```
females_no_1 <- dplyr::filter(females, male  != 1 | years != 1 | attempts != 1)
males_no_1 <- dplyr::filter(males, female  != 1 | years != 1 | attempts != 1)
```

tidy up dataframes then bind them together

```
females_no_1$sex <- "Female"
females_no_1$sex <- as.factor(females_no_1$sex)
colnames(females_no_1)[c(1,9)] <- c("focal", "mate")
males_no_1$sex <- "Male"
males_no_1$sex <- as.factor(males_no_1$sex)
colnames(males_no_1)[c(1,9)] <- c("focal", "mate")
mating <- rbind(females_no_1, males_no_1)
```

calculate the number of mates per year

```
mating$no_mates_per_year <- mating$mate/mating$years
```

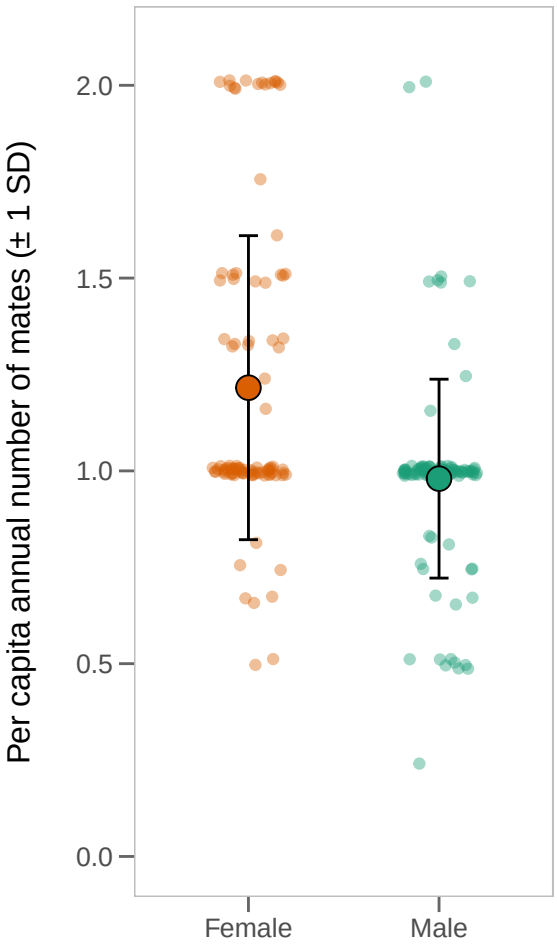
summarise the matings by sex and determine "h", the average annual number of mates per female

```
sex_specific_mating_system <-
  mating%>%
```

```
    dplyr::group_by(sex)%>%
    dplyr::summarise(mean_annual_no_mates = mean(no_mates_per_year),
                     var_annual_no_mates = var(no_mates_per_year),
                     median_annual_no_mates = median(no_mates_per_year),
                     sd_annual_no_mates = sd(no_mates_per_year))

# because the mating system is polyandrous, h must be less than 1,
# and therefore the inverse is calculated
h <-
  1/as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female"), 2])

# display the h value (used in the mating function of the matrix model)
h
#> [1] 0.8224986
```

Figure 1: plot the sex-specific distributions of mating system

```
# specifiy the color palette to use in plotting
cbPalette <- RColorBrewer::brewer.pal(8, "Dark2")[c(2,1)]

# define the dodge level
pd <- position_dodge(0.1)

# draw the plot
Sex_specific_mating_plot <-
ggplot2::ggplot() +
  geom_jitter(aes(y = no_mates_per_year, x = sex, fill = sex, color = sex),
              data = mating, width = 0.2, alpha = 0.4) +
  geom_errorbar(data = sex_specific_mating_system,
                aes(x = sex, y = mean_annual_no_mates,
                    ymin=mean_annual_no_mates-sd_annual_no_mates,
                    ymax=mean_annual_no_mates+sd_annual_no_mates),
                width=.1, position=pd, colour = "black") +
  geom_point(data = sex_specific_mating_system,
             aes(y = mean_annual_no_mates, x = sex, fill = sex),
             shape = 21, colour = "black", position = pd, size = 4) +
  theme_bw() +
  theme(text = element_text(size = 16),
        legend.position = "",
        axis.title.x = element_blank(),
        axis.text.x  = element_text(size = 10),
        axis.title.y = element_text(size = 12,
                                    margin = margin(0, 15, 0, 0)),
        axis.text.y = element_text(size = 10),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        panel.border = element_rect(linetype = "solid", colour = "grey")) +
  scale_fill_manual(values = cbPalette) +
  scale_color_manual(values = cbPalette) +
  ylab("Per capita annual number of mates (± 1 SD)") +
  scale_y_continuous(limits = c(0, 2.1))
Sex_specific_mating_plot
```



statistically test the sex-difference in the per capita annual number of mates by using a non-parametric Mann-Whitney-Wilcoxon Test.

```
wilcox.test(no_mates_per_year ~ sex, data = mating)
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  no_mates_per_year by sex
#> W = 6533, p-value = 2.994e-06
#> alternative hypothesis: true location shift is not equal to 0
```

Sex-specific fecundity

The objective here was to determine the average per capita annual fecundity for females. This vital rate was then incorporated into the one-sex matrix model (see section on the two-sex vs one-sex model comparison).

Step one: wrangle the data

Extract the female column from the breeding data, add a sex column, extract the male colum, add a sex column, then stack these two dataframes.

```
Sex <- rep("Female", nrow(breeding_data))
Ring <- breeding_data$female
```

```
females <- data.frame(Ring, Sex)
Sex <- rep("Male", nrow(breeding_data))
Ring <- breeding_data$male
males <- data.frame(Ring, Sex)
Individuals <- rbind(males, females)
```

replicate each row by 2 then cbind the stacked dataframe from the previous step

```
reproduction_df <- cbind(breeding_data[rep(row.names(breeding_data), 2),
                                       c("no_chicks", "clutch_size", "brood_ID", "year")],
                         Individuals)
```

change the order of the sex levels, so that females are first (for the plot)

```
reproduction_df$Sex <- factor(reproduction_df$Sex, levels = c("Female", "Male"))
```

subset the data to remove entries that have a NA in the Ring column

```
reproduction_df <- reproduction_df[!is.na(reproduction_df$Ring),]
```

subset the data to remove entries that have a NA in the no-chicks column

```
reproduction_df <- reproduction_df[!is.na(reproduction_df$no_chicks),]
```

group data according to Year, Sex, then Ring

```
reproduction_df <- dplyr::group_by(reproduction_df, year, Sex, Ring)
```

sum the total chicks produced per bird each year

```
reproduction_df_sum <-
  dplyr::ungroup(dplyr::summarise(reproduction_df,
                            total_chicks_p_year = sum(as.numeric(no_chicks))))
```

Step two: calculate fecundity

calculate avg total chicks produced per bird in each year

```
fecundity_annual_summary <-
  Rmisc::summarySE(reproduction_df_sum, measurevar = "total_chicks_p_year",
                   groupvars = c("Sex", "year"))
```

group data according to Sex then Ring

```
reproduction_df_sum <- dplyr::group_by(reproduction_df_sum, Sex, Ring)
```

calculate avg total chicks produced per bird each year

```
reproduction_df_sum_avg <-
  dplyr::ungroup(dplyr::summarise(reproduction_df_sum,
                            avg_chicks_p_year = mean(as.numeric(total_chicks_p_year))))
```

summarize the avg annual no_chicks by sex

```
fecundity_sex_summary <-
  Rmisc::summarySE(fecundity_annual_summary,
                   measurevar = "total_chicks_p_year", groupvars = c("Sex"))
```

Assign the value of female per capita annual fecundity to a constant that will be included in the one-sex matrix assessed later

```
RF <- fecundity_sex_summary[1,3]
RF
#> [1] 2.03688
```

Hatching sex ratio

The hatching sex ratio represents "rho" in the matrix model and is calculated from broods that met two criteria: 1) the brood size was the modal clutch size (3 in the case of snowy plovers), and 2) chicks were captured and sampled on the day of hatching. These criteria made sure to control for post-hatch brood mixing.

Step one: wrangle the data

Subset the chick mark-recapture data so that only chicks captured on the day of hatch are included. In this dataframe, the "ch" column refers to the capture history of an individual on each day of its life as a chick. Thus, if the first character of the "ch" string is a 1, it was captured on the day of hatch and is included in the hatch sex ratio dataset.

```
caught_at_hatch <- chick[which(substring(chick$ch, 1, 1) == "1"),]
```

sum the number of chicks that are included for each hatch ID

```
brood_ID_count <-
caught_at_hatch %>%
  dplyr::count(brood_ID)
```

join this data to the subset capture data

```
caught_at_hatch <- dplyr::left_join(caught_at_hatch, brood_ID_count, by = "brood_ID")
```

subset these data so that clutch size equals the number of chicks sampled from each nest

```
HSR_df <- caught_at_hatch[which(caught_at_hatch$clutch_size == caught_at_hatch$n),]
```

make new columns "Male" and "Female" that have 1 or 0 to describe the sex of the chick

```
HSR_df$male <- ifelse(HSR_df$sex == "M", 1, 0)
HSR_df$female <- ifelse(HSR_df$sex == "F", 1, 0)
```

define hatch ID as a factor

```
HSR_df$brood_ID <- as.factor(HSR_df$brood_ID)
```

Step two: mixed effects linear regression

Brood ID is used as a random effect to control for the non-independence of siblings

```
HSR_model <- lme4::glmer(cbind(male, female) ~ (1|brood_ID),
                         data = HSR_df, family = binomial)
```

check out the model results. P = 0.588, therefore hatching sex ratio doesn't deviate from parity

```
summary(HSR_model)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(male, female) ~ (1 | brood_ID)
#>    Data: HSR_df
```

```
#>
#>      AIC      BIC   logLik deviance df.resid
#>    475.0    482.7   -235.5    471.0      338
#>
#> Scaled residuals:
#>    Min     1Q Median     3Q    Max
#> -0.971 -0.971 -0.971  1.030  1.030
#>
#> Random effects:
#>  Groups   Name        Variance Std.Dev.
#>  brood_ID (Intercept) 0        0
#> Number of obs: 340, groups:  brood_ID, 116
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -0.05884    0.10851  -0.542    0.588
```

calculate what the average hatching sex ratio is summarize the data so that each row is a nest instead of an individual

```
HSR_df_summary <-
  HSR_df %>%
  dplyr::group_by(brood_ID) %>%
  dplyr::summarise(no_males = sum(male),
                   hatch_date_season = min(day_of_season),
                   clutch_size = mean(n),
                   year = first(year))
```

calculate the proportion of the brood that was male

```
HSR_df_summary$prop_male <- HSR_df_summary$no_males/HSR_df_summary$clutch_size
```

calculate the average hatching sex ratio across all nests and assign the result to a constant "HSR" to be used as rho in the matrix model

```
HSR <- mean(HSR_df_summary$prop_male)
HSR
#> [1] 0.4856322
```

calculate the 95% confidence interval of the hatching sex ratio

```
HSR_95CI <- c(mean(HSR_df_summary$prop_male)-
                ((sd(HSR_df_summary$prop_male)/
                    sqrt(length(HSR_df_summary$prop_male)))*1.96),
              mean(HSR_df_summary$prop_male)+
                ((sd(HSR_df_summary$prop_male)/
                    sqrt(length(HSR_df_summary$prop_male)))*1.96))
```

Bootstrapping proceedure of stage- and sex specific survival

Specify where RMark should look on your computer for Program MARK. This may vary based on your operating system (e.g., Windows, Linux, Mac OS X, etc.). This website provides a nice workflow for installing Program MARK and linking it to your R interface based on which operating system you have.

```
MarkPath <- "/usr/local/bin/mark"
MarkViewer <- "nano"
```

Step one: Assign functions

The following two functions are needed to setup the projection matrix and estimate ASR. Load these before implementing the bootstrap simulation.

plover_matrix() builds the two-sex Lefkovitch matrix using the vital rates specified in the *demographic_rates* object.

```
plover_matrix <-
  function(demographic_rates, two_sex = TRUE){
    if(two_sex){
      # Define plover life-stages of the Ceuta snowy plover matrix model
      stages <- c("F_1st_yr",  "F_Adt",  "M_1st_yr",  "M_Adt")
      # Build the 4x4 matrix
      result <-
        matrix(c(
                 # top row of matrix
                 0, NA, 0, NA,
                 # second row of matrix
                 (demographic_rates$F_Chk_survl*demographic_rates$F_Juv_survl),
                 demographic_rates$F_Adt_survl,
                 0, 0,
                 # third row of matrix
                 0, NA, 0, NA,
                 # fourth row of matrix
                 0, 0,
                 (demographic_rates$M_Chk_survl*demographic_rates$M_Juv_survl),
                 demographic_rates$M_Adt_survl),
               nrow = length(stages), byrow = TRUE,
               dimnames = list(stages, stages))
    }
    else{
      # Define plover life-stages of the Ceuta snowy plover matrix model
      stages <- c("1st_yr",  "Adt")
      # Build the 4x4 matrix
      result <-
        matrix(c(
                 # top row of matrix
                 0, RF,
                 # second row of matrix
                 (demographic_rates$Chk_survl*demographic_rates$Juv_survl),
                 demographic_rates$Adt_survl),
               nrow = length(stages), byrow = TRUE,
               dimnames = list(stages, stages))
    }
      result
  }
```

matrix_ASR() calculates the ASR of the population based on the two-sex two-stage projection matrix built by the *plover_matrix()* function. Arguments in the function include: A is an two sex x by x projection matrix n is an x lengthed vector representing starting stage distribution (the default is a vector with 10 individuals in each stage)

```
matrix_ASR <-
  function(M, n = rep(10, nrow(M)), h = 1, k = 3,
           iterations = 1000, HSR = 0.5, plot = FALSE){
    # Number of stages in matrix
    x <- length(n)
    # Number of time steps to simulate
    t <- iterations
    # an empty t by x matrix to store the stage distributions
    stage <- matrix(numeric(x * t), nrow = x)
    # an empty t vector to store the population sizes
    pop <- numeric(t)
    # for loop that goes through each of t time steps
    for (i in 1:t) {
      # stage distribution at time t
      stage[,i] <- n
      # population size at time t
      pop[i] <- sum(n)
      # number of male adults at time t
      M2 <- stage[4, i]
      # number of female adults at time t
      F2 <- stage[2, i]
      # Female freq-dep fecundity of Female chicks
      M[1,x/2]        <- (k*M2)/(M2+(F2*h))*HSR
      # Female freq-dep fecundity of Male chicks
      M[(x/4)*3,x/2]  <- (k*M2)/(M2+(F2*h))*HSR
      # Male freq-dep fecundity of Female chicks
      M[1,x]          <- (k*F2)/(M2+(F2*h))*HSR
      # Male freq-dep fecundity of Male chicks
      M[(x/4)*3,x]    <- (k*F2)/(M2+(F2*h))*HSR
      # define the new n (i.e., new stage distribution at time t)
      n <- M %*% n
      # define rownames of stage matrix
      rownames(stage) <- rownames(M)
      # define colnames of stage matrix
      colnames(stage) <- 0:(t - 1)
      # calculate the proportional stable stage distribution
      stage <- apply(stage, 2, function(x) x/sum(x))
      # define stable stage as the last stage
      stable.stage <- stage[, t]
    }
    # calc ASR as the proportion of the adult stable stage class that is male
    ASR <- stable.stage[x]/(stable.stage[x/2] + stable.stage[x])

    if(plot)
    {
      # plot distrubution to assure that it is not chaotic
      matplot(rownames(t(stage)), t(stage), type='l', lwd=2, las=1)
    }
    # make a list of results
    pop.proj <- list(ASR = ASR,
                     lambda = pop[t]/pop[t - 1],
                     stable.stage = stable.stage,
                     stage.vectors = stage,
```

```
                    SSD_M2 = stable.stage[4],
                    SSD_F2 = stable.stage[2])
    # print the list as output to the function
    pop.proj
  }
```

Step two: running the bootstrap

Each iteration will do the following computational steps:

A) Load the following function **bootstrap_data()** to randomly sample with replacement from the *chick* and *juvenile_adult* datasets, while making sure that if an individual existing in both datasets was sampled from the *chick* data it was also sampled in the *juvenile_adult* data. Each bootstrapped sample has the same length as the original data.

```
bootstrap_data <- function(juvenile_adult, chick) {

  # sample a new chick mark-recapture dataset the same size as the original,
  # with replacement
  chick_boot <- chick[sample(1:nrow(chick),
                             size = nrow(chick),
                             replace = TRUE), ]

  # determine if there are any individuals in the new chick data that are in the
  # adult data
  present <- juvenile_adult$bird_ID %in% chick_boot$bird_ID

  # extract these individuals from the adult data
  juvenile_adult_boot1 <- juvenile_adult[present, ]

  # determine the left over adults
  spare_juvenile_adult <- juvenile_adult[!present, ]

  # randomly sample from these left over adults
  juvenile_adult_boot2 <-
    spare_juvenile_adult[sample(1:nrow(spare_juvenile_adult),
                                size = nrow(juvenile_adult) -
                                  nrow(juvenile_adult_boot1),
                                replace = TRUE), ]

  # bind these two adult samples together
  juvenile_adult_boot <- rbind(juvenile_adult_boot1, juvenile_adult_boot2)

  # make a list of these two datasets, which will be used in the next function
  out <- list(chick_boot = chick_boot, juvenile_adult_boot = juvenile_adult_boot)
}
```

B) The next function, **bootstrap_survival_ASR()**, runs the survival analyses and estimates the ASR of the bootstrapped sample created from **bootstrap_data()**. In the function, *plover_boot_list* is the output list from **bootstrap_data()** and *num_boot* is the bootstrap number in the loop (leave unspecified).

```
bootstrap_survival_ASR <- function(plover_boot_list, num_boot) {

  # specify the bootstrapped data samples (from the previous function)
  chick <- plover_boot_list[["chick_boot"]]
```

```r
juvenile_adult <- plover_boot_list[["juvenile_adult_boot"]]

# remove ring column
juvenile_adult <- juvenile_adult[,-1]
chick <- chick[,-1]

# Create processed RMark data formatted as Cormack-Jolly_Seber with 2 groups
# (sex and stage initally ringed), starting at year 2006, two stage groups
# (first-years and adults) in which the first-year stage only lasts for
# one year.
juvenile_adult.proc <- RMark::process.data(juvenile_adult, model = "CJS",
                                           groups = c("sex", "stage"),
                                           begin.time = 2006, age.var = 2,
                                           initial.age = c(1, 0))

# Create processed RMARK data format as Cormack-Jolly_Seber with 3 groups
# (sex, year, and brood ID).
chick.proc <-  RMark::process.data(chick, model = "CJS",
                                   groups = c("sex", "year", "brood_ID"))

# Create the design matrix from the processed mark-recapture datasets
juvenile_adult.ddl <- RMark::make.design.data(juvenile_adult.proc)
chick.ddl <- RMark::make.design.data(chick.proc)

# adds first-year / adult stage field to design data in column "Age"
juvenile_adult.ddl <- RMark::add.design.data(data = juvenile_adult.proc,
                                             ddl = juvenile_adult.ddl,
                                             parameter = "Phi",
                                             type = "age",
                                             bins = c(0, 1, 7), right = FALSE,
                                             name = "age", replace = TRUE)

# create a dummy field in the design matrix called marked.as.adult
# which is "0" for the group initally ringed as chicks and "1" for the group
# marked as adults.
juvenile_adult.ddl$Phi$marked.as.adult = 0
juvenile_adult.ddl$Phi$marked.as.adult[juvenile_adult.ddl$Phi$initial.age.class=="A"]=1
juvenile_adult.ddl$p$marked.as.adult = 0
juvenile_adult.ddl$p$marked.as.adult[juvenile_adult.ddl$p$initial.age.class=="A"]=1

# check parameter matrices to see if groups were binned correctly
# (uncomment the next three lines to assess)
# PIMS(mark(juvenile_adult.proc, juvenile_adult.ddl,
#           model.parameters = list(Phi = list(formula = ~ stage + sex)),
#           output = F), "Phi")

# Create quadratic time variable so that it can be tested for temporal variation
# chick survival (i.e. non-linear relationship between daily chick survival and age)
time <- c(0:(chick.proc$nocc[1] - 1))
quadratic <- time^2
quad_time <- data.frame(time, quadratic)
chick.ddl$p <-
  RMark::merge_design.covariates(chick.ddl$Phi,
```

```
                                    quad_time, bygroup = FALSE, bytime = TRUE)
chick.ddl$Phi <-
  RMark::merge_design.covariates(chick.ddl$Phi,
                                    quad_time, bygroup = FALSE, bytime = TRUE)

# create the function that specifies the candidate models of juvenile and adult
# resight probability
juvenile_adult_survival = function()
{
  # sex- and stage-specific survival:
  Phi.agexsex = list(formula = ~ stage * sex)

  # Models exploring variation in encounter probability
  # constant:
  p.dot = list(formula =  ~ 1)
  # sex-dependent:
  p.sex = list(formula =  ~ sex)
  # stage-dependent:
  p.stage = list(formula =  ~ stage)
  # factorial variation across year:
  p.Time = list(formula =  ~ Time)
  # interaction between sex and factorial year:
  p.sexxTime = list(formula =  ~ sex * Time)
  # interaction between stage and factorial year:
  p.agexTime = list(formula =  ~ stage * Time)
  # interaction between stage and sex:
  p.agexsex = list(formula =  ~ sex * stage)
  # additive effects of sex and factorial year:
  p.sex_Time = list(formula =  ~ sex + Time)
  # additive effects of stage and factorial year:
  p.age_Time = list(formula =  ~ stage + Time)
  # additive effects of stage and sex:
  p.age_sex = list(formula =  ~ sex + stage)
  # additive effects of sex, stage, factorial year:
  p.Time_age_sex = list(formula =  ~ sex + stage + Time)
  # additive effect of year and interaction between stage and sex:
  p.Time_age_x_sex = list(formula =  ~ sex * stage + Time)

  # create a list of candidate models for all the a models above that begin with
  # either "Phi." or "p."
  cml <-  RMark::create.model.list("CJS")

  # specify the data, design matrix, delete unneeded output files, and
  # run the models in Program MARK
  model.list <-  RMark::mark.wrapper(cml, data = juvenile_adult.proc,
                                     ddl = juvenile_adult.ddl, delete = TRUE)

  # output the model list and sotre the results
  return(model.list)
}

# Run the models on the bootstrapped data
juvenile_adult_survival_run <-
```

```
  juvenile_adult_survival()

# Extract the AIC model table from the model output
AIC_table_juvenile_adult <-
  juvenile_adult_survival_run$model.table

# Find the model number for the first ranked model of the AIC table
model_juvenile_adult_num <-
  as.numeric(rownames(juvenile_adult_survival_run$model.table[1,]))

# extract and format survival rates from juvenile and adult model output
juvenile_adult_reals <-
  juvenile_adult_survival_run[[model_juvenile_adult_num]]$results$real

# format the output to tidy up the sex- and stage-specific effects
Groups <- data.frame(str_split_fixed(rownames(juvenile_adult_reals), " ", n = 5))
juvenile_adult_reals <- cbind(Groups, juvenile_adult_reals)
juvenile_adult_reals <-
  juvenile_adult_reals[which(juvenile_adult_reals$X1 == "Phi"),]
juvenile_adult_reals$stage <-
  unlist(str_extract_all(juvenile_adult_reals$X2,"[AJ]"))
juvenile_adult_reals$stage <-
  as.factor(ifelse(juvenile_adult_reals$stage == "A","Adult","juvenile"))
juvenile_adult_reals$sex <-
  unlist(str_extract_all(juvenile_adult_reals$X2,"[FM]"))
juvenile_adult_reals$sex <-
  as.factor(ifelse(juvenile_adult_reals$sex == "F","Female","Male"))
juvenile_adult_reals$sex_stage <-
  paste(juvenile_adult_reals$sex,juvenile_adult_reals$stage,sep = "_")
juvenile_adult_survival_real <-
  juvenile_adult_reals[,c("sex_stage", "estimate")]
row.names(juvenile_adult_survival_real) <- NULL

# Do the same for chicks. create the function that specifies the candidate models
# of chick resight probability
chick_survival = function()
{
  # sex- and quadratic age-specific survival:
  Phi.quadratic.x.sex = list(formula = ~ sex * quadratic)

  # Models exploring variation in encounter probability
  # constant:
  p.dot = list(formula = ~ 1)
  # quadratic across age
  p.quadratic = list(formula = ~ quadratic)
  # annual variation
  p.year = list(formula = ~ year)
  # sex-specific
  p.sex = list(formula = ~ sex)
  # interaction between year and quadratic age
  p.year.x.quadratic = list(formula = ~ quadratic * year)
  # interaction between year and quadratic age
  p.sex.x.quadratic = list(formula = ~ sex * quadratic)
```

```
  # additive effects of sex and linear age
  p.sex.quadratic = list(formula = ~ sex + quadratic)
  # additive effects of year and quadratic age
  p.year.quadratic = list(formula = ~ quadratic + year)
  # additive effects of year, sex, and quadratic age
  p.year.quadratic.Sex = list(formula = ~ sex + quadratic + year)
  # additive effect of year and interaction between sex and quadratic age
  p.year.quadratic.x.Sex = list(formula = ~ sex * quadratic + year)

  # create a list of candidate models for all the a models above that begin with
  # either "Phi." or "p."
  cml <-  RMark::create.model.list("CJS")

  # specify the data, design matrix, delete unneeded output files, and
  # run the models in Program MARK
  model.list <-  RMark::mark.wrapper(cml, data = chick.proc,
                                     ddl = chick.ddl, delete = TRUE)

  # output the model list and sotre the results
  return(model.list)
}

# Run the models on the bootstrapped data
chick_survival_run <- chick_survival()

# Extract the AIC model table from the model output
AIC_table_chick <- chick_survival_run$model.table

# Find the model number for the first ranked model of the AIC table
model_chick_num <- as.numeric(rownames(chick_survival_run$model.table[1,]))

# extract real parameter estimates from top models
chick_reals <- chick_survival_run[[model_chick_num]]$results$real

# format the output to tidy up the sex- and age-specific effects
Groups <- data.frame(str_split_fixed(rownames(chick_reals), " ", n = 5))
chick_reals <- cbind(Groups, chick_reals)
chick_reals <- chick_reals[which(chick_reals$X1 == "Phi"),]
chick_reals$sex <- unlist(str_extract_all(chick_reals$X2,"[FM]"))
chick_reals$sex <- as.factor(ifelse(chick_reals$sex == "F","Female","Male"))

# transform the daily chick survival (DCS) to apparent hatching success
# by calculating the product of all DCS estimates:
  plover_Survival_to_Fledge_F <-
    prod(chick_reals[which(chick_reals$sex == "Female"),
                     c("estimate")][c(1:26)])
  plover_Survival_to_Fledge_M <-
    prod(chick_reals[which(chick_reals$sex == "Male"),
                     c("estimate")][c(1:26)])

# tidy up the output and put it in a dataframe.
estimate <- c(plover_Survival_to_Fledge_F, plover_Survival_to_Fledge_M)
sex <- c("Female", "Male")
```

```
  stage <- c("Chick", "Chick")
  sex_stage <- paste(sex, stage, sep = "_")
  chick_survival_real <- data.frame(sex_stage, estimate)

  # Bind the juvenile and adult dataframe with the chicks
  survival_rates <- rbind(juvenile_adult_survival_real, chick_survival_real)

  # Create a list of demographic rates from the survival analyses above
  demographic_rates <- list(F_Chk_survl = survival_rates[5,2],
                            F_Juv_survl = survival_rates[3,2],
                            F_Adt_survl = survival_rates[1,2],
                            M_Chk_survl = survival_rates[6,2],
                            M_Juv_survl = survival_rates[4,2],
                            M_Adt_survl = survival_rates[2,2],
                            # Define hatching sex ratio
                            HSR = HSR,
                            # Define the mating system (h), and clutch size (k)
                            h = h,
                            k = 3)

  # Build matrix based on rates specified in the list above
  demographic_matrix <- plover_matrix(demographic_rates)

  # Determine the ASR at the stable stage distribution
  ASR_SSD <- matrix_ASR(M = demographic_matrix, h = demographic_rates$h,
                        HSR = demographic_rates$HSR, iterations = 1000)

  # Extract ASR
  ASR_estimate <- ASR_SSD$ASR

  # make a list of all the results from this iteration
  bootstrap_results_list <-
    list(AIC_table_chick,
         AIC_table_juvenile_adult,
         survival_rates,
         ASR_estimate)
}
```

C) Create a function to run the **bootstrap_data()** and **bootstrap_survival_ASR()** functions in sequence.

```
run_bootstrap_survival_ASR <- function(num_boot, juvenile_adult, chick)
  {
  # run the sampling function and specify the datasets
  bootstrap_data_list <- bootstrap_data(juvenile_adult, chick)

  # run the survival analysis and ASR deduction on the sampled data
  result <- bootstrap_survival_ASR(bootstrap_data_list, num_boot)
}
```

D) Specify the number of iterations to run in the bootstrap (1000 was used in our analysis).

```
niter <- 1000
```

E) start the bootstrap (takes approx. 130 hours on an Intel XEON E5v2 series sever with 40 threads)

```
# uncomment this to run the bootstrap. To bypass this, load the bootstrap output datasets
# below to continue analysis

# survival_ASR_bootstrap_result <-
#   sapply(1:niter, run_bootstrap_survival_ASR, juvenile_adult, chick)
```

F) Extract data from the bootstrap output (uncomment these sections if you ran the bootstrap)

AIC tables of chick survival for each interation

```
# AIC_table_chick_boot <-
# do.call(rbind, lapply(seq(from = 1, to = niter * 4, by = 4),
#                       function(x) survival_ASR_bootstrap_result[[x]]))
# num_mods <- nrow(AIC_table_chick_boot)/niter
# AIC_table_chick_boot$iter <- rep(1:niter, each = num_mods)
```

AIC tables of juvenile and adult survival for each interation

```
# AIC_table_juvenile_adult_boot <-
# do.call(rbind, lapply(seq(from = 2, to = niter * 4, by = 4),
#                       function(x) survival_ASR_bootstrap_result[[x]]))
# num_mods <- nrow(AIC_table_juvenile_adult_boot)/niter
# AIC_table_juvenile_adult_boot$iter <- rep(1:niter, each = num_mods)
```

Survival rates for each iteration

```
# survival_rates_boot <-
# do.call(rbind, lapply(seq(from = 3, to = niter * 4, by = 4),
#                       function(x) survival_ASR_bootstrap_result[[x]]))
# survival_rates_boot$iter <- rep(1:niter, each = 6)
```

ASR estimate for each iteration

```
# ASR_boot <-
# sapply(seq(from = 4, to = niter * 4, by = 4),
#        function(x) survival_ASR_bootstrap_result[[x]])
# ASR_boot <- data.frame(ASR_boot = unname(ASR_boot), iter = 1:niter)
```

To save your time with re-running the bootstrap, here are the four datasets produced by the bootstrap:

- **output/bootstrap/AIC_table_chick_boot_out.txt** contains the bootstrap output for model selection of chick survival based on the mark-recapture analysis run in Program MARK. Each row is a *model* fitted via maximum likelihood to the bootstrapped data sample of each iteration (*iter*). *Phi* describes the model structure for fitting daily survival. *p* describes the model structure for fitting daily encounter probability. *npar* reveals the number of parameters used in a given model. *AICc* is the Akaike Information Criteria statistic corrected for small sample size. *DeltaAICc* is the difference in AICc between a given model and the best fit model of a given iteration. *weight* describes the AIC weight of a given model. *Deviance* describes the deviance of a given model.
- **output/bootstrap/AIC_table_juvenile_adult_boot_out.txt** contains the bootstrap output for model selection of juvenile and adult survival based on the mark-recapture analysis run in Program MARK. Each row is a *model* fitted via maximum likelihood to the bootstrapped data sample of each iteration (*iter*). *Phi* describes the model structure for fitting annual survival. *p* describes the model structure for fitting annual encounter probability. *npar* reveals the number of parameters used in a given model. *AICc* is the Akaike Information Criteria statistic corrected for small sample size. *DeltaAICc* is the difference in AICc between a given model and the best fit model of a given iteration. *weight* describes the AIC weight of a given model. *Deviance* describes the deviance of a given model.

- **output/bootstrap/ASR_boot_out.txt** contains the adult sex ratio estimates (*ASR_boot*) of each iteration of the bootstrap procedure. Each row represents an iteration (*iter*).
- **output/bootstrap/survival_rates_boot_out.txt** contains the sex- and stage-specific survival estimates (*estimate*) of each iteration (*iter*) in the bootstrap procedure. Each row represents a given sex and stage (*sex_stage*) in a given iteration.

```
setwd("~/Dropbox/Luke/R_projects/Ceuta_ASR_Matrix_Modeling")
chick_AIC_tables <-
  read.table("output/bootstrap/AIC_table_chick_boot_out.txt", header = TRUE)

juv_ad_AIC_tables <-
  read.table("output/bootstrap/AIC_table_juvenile_adult_boot_out.txt", header = TRUE)

survival_rates_boot <-
  read.table("output/bootstrap/survival_rates_boot_out.txt", header = TRUE)

ASR_boot <-
  read.table("output/bootstrap/ASR_boot_out.txt", header = TRUE)
```

Visualizations of bootstrap results

Sex-biases in survial across chicks, juveniles, and adults

We visualized sex-bias in stage-specific survival rates with violin plots. These plots are useful for illustrating the spread of the bootstrap distribution. We have also added the inter-quartile ranges as horizontal bars within the violins. Before plotting, the sex-bias at each stage for each bootstrap iteration needs to be calculated. This is done with the **sex_diff_surv()** function and specifying the output list from the bootstrap above.

```
sex_diff_survival <- function(survival_rates_boot) {

  # make an empty datarame to store the results
  sex_diff_surv_output <- data.frame(Adult = numeric(niter),
                                     Juvenile = numeric(niter),
                                     Chick = numeric(niter))

  # for loop to go through each iteration and calculate the differece between
  # female and male survival rates for each stage.
  for(i in 1:niter){
    Adult <-
      survival_rates_boot[which(survival_rates_boot$iter == i), 2][2] -
      survival_rates_boot[which(survival_rates_boot$iter == i), 2][1]
    Juvenile <-
      survival_rates_boot[which(survival_rates_boot$iter == i), 2][4] -
      survival_rates_boot[which(survival_rates_boot$iter == i), 2][3]
    Chick <-
      survival_rates_boot[which(survival_rates_boot$iter == i), 2][6] -
      survival_rates_boot[which(survival_rates_boot$iter == i), 2][5]

    sex_diff_surv_output[i, 1] <- Adult
    sex_diff_surv_output[i, 2] <- Juvenile
    sex_diff_surv_output[i, 3] <- Chick
  }
```

```
  # restructure the output and lable columns
  sex_diff_surv_output <- reshape2::melt(data = sex_diff_surv_output)
  colnames(sex_diff_surv_output) <- c("stage", "difference")

  # return the output
  sex_diff_surv_output
}
```

run the function on the bootstrap list from above

```
sex_diff_survival_output <- sex_diff_survival(survival_rates_boot)
```

calculate some summary statistics

```
sex_diff_survival_summary <-
    sex_diff_survival_output %>%
    dplyr::group_by(stage) %>%
    dplyr::summarise(avg = mean(difference),
                     median = median(difference),
                     var = var(difference))
```

specify custom color palette to distingush first-year stages (i.e. chicks and juveniles) from adults

```
cbPalette <- c("#A6A6A6", "#D9D9D9", "#D9D9D9")
```

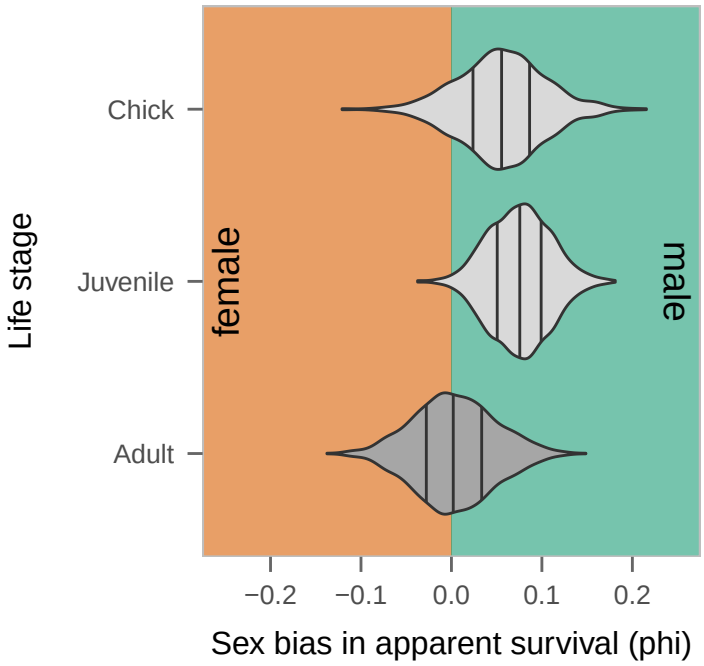
reorder the levels of the stage factors

```
sex_diff_survival_output$stage <-
  factor(sex_diff_survival_output$stage, levels = c("Adult", "Juvenile", "Chick"))
```

Figure 2a: plot the sex-biases in survival across the three stages

```
Background <-
  ggplot(aes(y = difference, x = stage, fill = stage), data = sex_diff_survival_output) +
  coord_flip() +
  theme_bw() +
  annotate("rect", xmin=-Inf, xmax=Inf, ymin=-Inf, ymax=0, alpha=0.6,
           fill=brewer.pal(8, "Dark2")[c(2)]) +
  annotate("rect", xmin=-Inf, xmax=Inf, ymin=0, ymax=Inf, alpha=0.6,
           fill=brewer.pal(8, "Dark2")[c(1)]) +
  annotate("text", x = 2, y = -0.25,
           label = c("female"), size = 5,
           vjust = c(0.5), hjust = c(0.5), angle = 90) +
  annotate("text", x = 2, y = 0.25,
           label = c("male"), size = 5,
           vjust = c(0.5), hjust = c(0.5), angle = 270) +
  theme(text = element_text(color = "white"),
        legend.position = "none",
        axis.title.x = element_text(size=12, margin = margin(10, 0, 0, 0)),
        axis.text.x  = element_text(size=10, margin = margin(5, 0, 0, 0), color = "white"),
        axis.title.y = element_text(size=12, margin = margin(0, 15, 0, 0)),
        axis.text.y  = element_text(size=10, angle = 0, hjust = 1,
                                    margin = margin(0, 5, 0, 0), color = "white"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_blank(),
        axis.ticks.x = element_line(size = 0.5, colour = "white"),
```

```
        axis.ticks.length = unit(0.2, "cm"),
        panel.border = element_blank(),
        plot.margin = unit(c(1,0.5,0.5,0.5), "cm"),
        panel.spacing = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  scale_x_continuous(limits=c(0,4),breaks=c(0,1,2), labels=c("Chick", "Juvenile", "Adult")) +
  scale_y_continuous(limits=c(-0.25,0.25)) +
  xlab("Life stage") +
  ylab("Sex bias in apparent survival (phi)")

Bootstrap_sex_diff_VR_plot <-
  ggplot(aes(y = difference, x = stage, fill = stage), data = sex_diff_survival_output) +
  coord_flip() +
  theme_bw() +
  geom_violin(draw_quantiles = c(0.25, 0.5, 0.75)) +
  theme(legend.position = "none",
        panel.background = element_rect(fill = "transparent",colour = NA),
        plot.background = element_rect(fill = "transparent",colour = NA),
        axis.title.x = element_text(size=12, margin = margin(10, 0, 0, 0)),
        axis.text.x  = element_text(size=10, margin = margin(5, 0, 0, 0)),
        axis.title.y = element_text(size=12, margin = margin(0, 15, 0, 0)),
        axis.text.y  = element_text(size=10, angle = 0, hjust = 1,
                                    margin = margin(0, 5, 0, 0)),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        axis.ticks.x = element_line(size = 0.5, colour = "grey40"),
        panel.border = element_rect(linetype = "solid", colour = "grey"),
        plot.margin = unit(c(1,0.5,0.5,0.5), "cm"),
        panel.spacing = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  scale_fill_manual(values = cbPalette) +
  scale_y_continuous(limits=c(-0.25,0.25)) +
  xlab("Life stage") +
  ylab("Sex bias in apparent survival (phi)")

grid.newpage()
pushViewport( viewport( layout = grid.layout( 1 , 1 , widths = unit( 1 , "npc" ) ) ) )
print( Background + theme(legend.position="none") ,
       vp = viewport( layout.pos.row = 1 , layout.pos.col = 1 ) )
print( Bootstrap_sex_diff_VR_plot + theme(legend.position="none") ,
       vp = viewport( layout.pos.row = 1 , layout.pos.col = 1 ) )
```



Adult sex ratio distribution

calculate the confidence interval, mean, and median of the ASR bootstraps

```
CI <- 0.95
ASR_boot_95CI <-
  stats::quantile(ASR_boot$ASR_boot, c((1 - CI)/2, 1 - (1 - CI)/2), na.rm = TRUE)
ASR_boot_mean <- mean(ASR_boot$ASR_boot)
ASR_boot_median <- median(ASR_boot$ASR_boot)
```

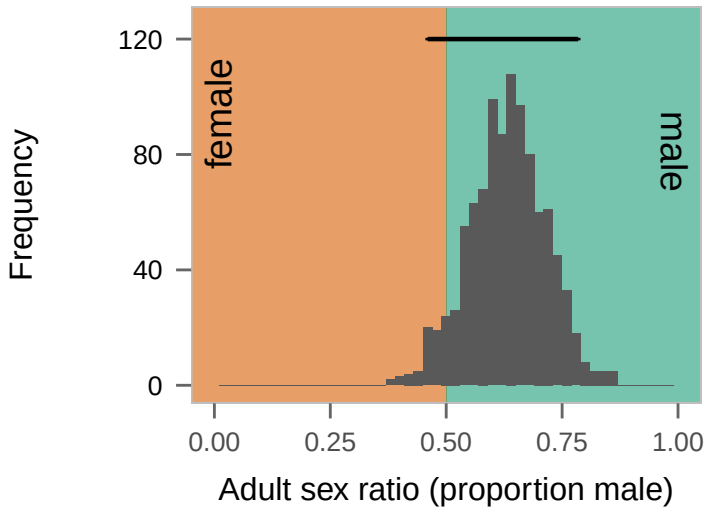
consolidate the results

```
ASR_boot_summary <- as.data.frame(cbind(ASR_boot_95CI[1], ASR_boot_95CI[2],
                                        ASR_boot_mean, ASR_boot_median))
rownames(ASR_boot_summary) <- NULL
colnames(ASR_boot_summary) <- c("lcl", "ucl", "mean", "median")
ASR_boot_summary
#>         lcl       ucl      mean    median
#> 1 0.4600267 0.7846838 0.6322504 0.6351836
```

Figure 2b: We visualized the bootstrapped results of adult sex ratio with a histogram. The horizontal black bar above the distribution illustrates the 95% confidence interval of the 1000 iterations.

```
ASR_bootstrap_histogram <-
ggplot2::ggplot() +
          annotate("rect", xmin=-Inf, xmax=0.5, ymin=-Inf, ymax=Inf, alpha=0.6,
                   fill=brewer.pal(8, "Dark2")[c(2)]) +
          annotate("rect", xmin=0.5, xmax=Inf, ymin=-Inf, ymax=Inf, alpha=0.6,
                   fill=brewer.pal(8, "Dark2")[c(1)]) +
          annotate("text", x = c(-Inf,Inf), y = c(75, 95),
                   label = c("female", "male"), size = 5,
                   vjust = c(1.5,1.5), hjust = c(0,0), angle = c(90, 270)) +
          geom_histogram(binwidth = 0.02, data = ASR_boot, aes(x = ASR_boot)) +
          geom_errorbarh(data = ASR_boot_summary,
                         aes(y = 120, x = lcl, xmin = lcl, xmax = ucl),
```

```
                        color = "black", size = 0.8, linetype = "solid") +
          theme_bw() +
          theme(legend.position = c(0, 1),
                legend.justification = c(0, 1),
                legend.text=element_text(size=11),
                legend.title=element_blank(),
                legend.key.height=unit(0.8,"line"),
                legend.key.width=unit(0.8,"line"),
                legend.background = element_rect(fill=NA),
                axis.title.x = element_text(size=12, margin = margin(10, 0, 0, 0)),
                axis.text.x  = element_text(size=10, margin = margin(5, 0, 0, 0)),
                axis.title.y = element_text(size=12, margin = margin(0, 30, 0, 0)),
                axis.text.y  = element_text(size=10, angle = 0, hjust = 1,
                                            margin = margin(0, 5, 0, 0), color = "black"),
                axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
                axis.ticks.x = element_line(size = 0.5, colour = "grey40"),
                axis.ticks.length = unit(0.2, "cm"),
                panel.grid.major = element_blank(),
                panel.grid.minor = element_blank(),
                panel.border = element_rect(linetype = "solid", colour = "grey"),
                plot.margin = unit(c(0.5,0.5,0.5,0.5), "cm"),
                strip.background = element_blank(),
                strip.text = element_blank(),
                panel.spacing = unit(0.75, "lines")) +
          ylab("Frequency") +
          xlab("Adult sex ratio (proportion male)") +
          scale_x_continuous(limits = c(0.0, 1)) +
          scale_y_continuous(limits = c(0, 125))
ASR_bootstrap_histogram
```



AIC model selection summary (panels in Supporting Information Figure 2)

To illustrate the mark-recapture model selection going on during the bootstrap, we summarized AIC statistics for each model included in the survival analysis and visualized with ranked boxplots (Figure S2)

First, wrangle the bootstrap AIC table output

```
# define the model number
chick_AIC_tables$model_number <- as.numeric(chick_AIC_tables$model)
```

```
juv_ad_AIC_tables$model_number <- as.numeric(juv_ad_AIC_tables$model)

# summarize the average AIC stats for each candidate model across all 1000 iterations
chick_AIC_tables_summary <-
  chick_AIC_tables %>%
  dplyr::group_by(model) %>%
  dplyr::summarise(avg_Delta = mean(DeltaAICc),
            IQR_Delta = IQR(DeltaAICc),
            avg_Weight = mean(weight),
            IQR_Weight = IQR(weight))

juv_ad_AIC_tables_summary <-
  juv_ad_AIC_tables %>%
  dplyr::group_by(model) %>%
  dplyr::summarise(avg_Delta = mean(DeltaAICc),
            IQR_Delta = IQR(DeltaAICc),
            avg_Weight = mean(weight),
            IQR_Weight = IQR(weight))

# rank the output by delta AIC and determine model number
chick_AIC_tables_summary <- dplyr::arrange(chick_AIC_tables_summary, avg_Delta)
chick_AIC_tables_summary$model_number <- as.numeric(chick_AIC_tables_summary$model)

juv_ad_AIC_tables_summary <- dplyr::arrange(juv_ad_AIC_tables_summary, avg_Delta)
juv_ad_AIC_tables_summary$model_number <- as.numeric(juv_ad_AIC_tables_summary$model)

# merge the two datasets for plotting
chick_AIC_tables <-
  dplyr::left_join(chick_AIC_tables_summary, chick_AIC_tables, by = "model_number")

juv_ad_AIC_tables <-
  dplyr::left_join(juv_ad_AIC_tables_summary, juv_ad_AIC_tables, by = "model_number")

# extract the model structure explaining resighting probability
chick_AIC_tables$p <-
  factor(chick_AIC_tables$p,
         levels = str_sub(as.character(chick_AIC_tables_summary$model),
                          start = 24, end = str_length(chick_AIC_tables_summary$model)-1))

juv_ad_AIC_tables$p <-
  factor(juv_ad_AIC_tables$p,
         levels = str_sub(as.character(juv_ad_AIC_tables_summary$model),
                          start = 18, end = str_length(juv_ad_AIC_tables_summary$model)-1))
```

plot the overall model ranks of the chick survival anlaysis based on Delta AIC

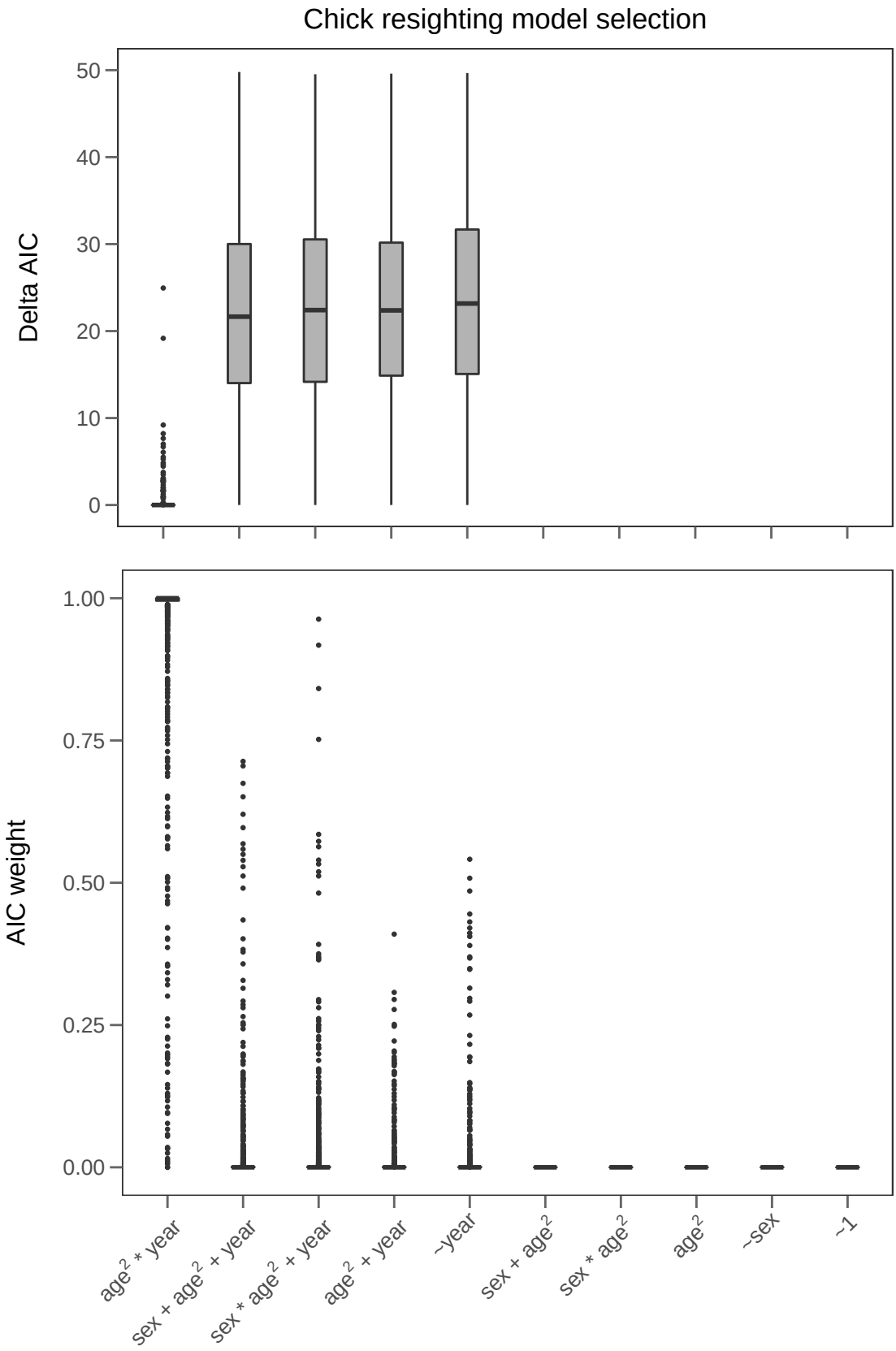
```
Bootstrap_Delta_AIC_plot_C <-
  ggplot(aes(y = DeltaAICc, x = p), data = chick_AIC_tables) +
  theme_bw() +
  geom_boxplot(width = 0.3, fill = "grey70", outlier.size = 0.5) +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x  = element_blank(),
        axis.title.y = element_text(size=12, margin = margin(0, 18, 0, 0)),
```

```
        axis.text.y  = element_text(size=10),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        axis.ticks.x = element_line(size = 0.5, colour = "grey40"),
        plot.margin = unit(c(0.5,0.5,0,0.5), "cm"),
        panel.spacing = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(limits=c(0,50)) +
  xlab("Model") +
  ylab("Delta AIC") +
  ggtitle("Chick resighting model selection")
```

plot the overall model ranks of the chick survival anlaysis based on AIC weight

```
Bootstrap_AIC_weight_plot_C <-
  ggplot(aes(y = weight, x = p), data = chick_AIC_tables) +
  theme_bw() +
  geom_boxplot(width = 0.3, fill = "grey70", outlier.size = 0.5) +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x  = element_text(size=10, angle = 45, hjust = 1),
        axis.title.y = element_text(size=12, margin = margin(0, 15, 0, 0)),
        axis.text.y  = element_text(size=10),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        axis.ticks.x = element_line(size = 0.5, colour = "grey40"),
        plot.margin = unit(c(0.5,0.5,0.5,0.3), "cm"),
        panel.spacing = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  scale_y_continuous(limits=c(0,1)) +
  xlab("Model") +
  ylab("AIC weight") +
  scale_x_discrete(labels = c("~year * quadratic" =
                                expression(paste("age"^2," * year")),
                              "~year + quadratic + sex" =
                                expression(paste("sex + age"^2," + year")),
                              "~year + quadratic * sex" =
                                expression(paste("sex * age"^2," + year")),
                              "~year + quadratic" =
                                expression(paste("age"^2," + year")),
                              "~year" = "~year",
                              "~sex + quadratic" =
                                expression("sex + age"^2),
                              "~sex * quadratic" =
                                expression("sex * age"^2),
                              "~quadratic" =
                                expression("age"^2),
```

```
                                "~sex" = "~sex",
                                "~1" = "~1"))

grid.newpage()
pushViewport(viewport(layout = grid.layout(5, 1)))
vplayout <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)
print(Bootstrap_Delta_AIC_plot_C, vp = vplayout(1:2, 1))
print(Bootstrap_AIC_weight_plot_C, vp = vplayout(3:5, 1))
```



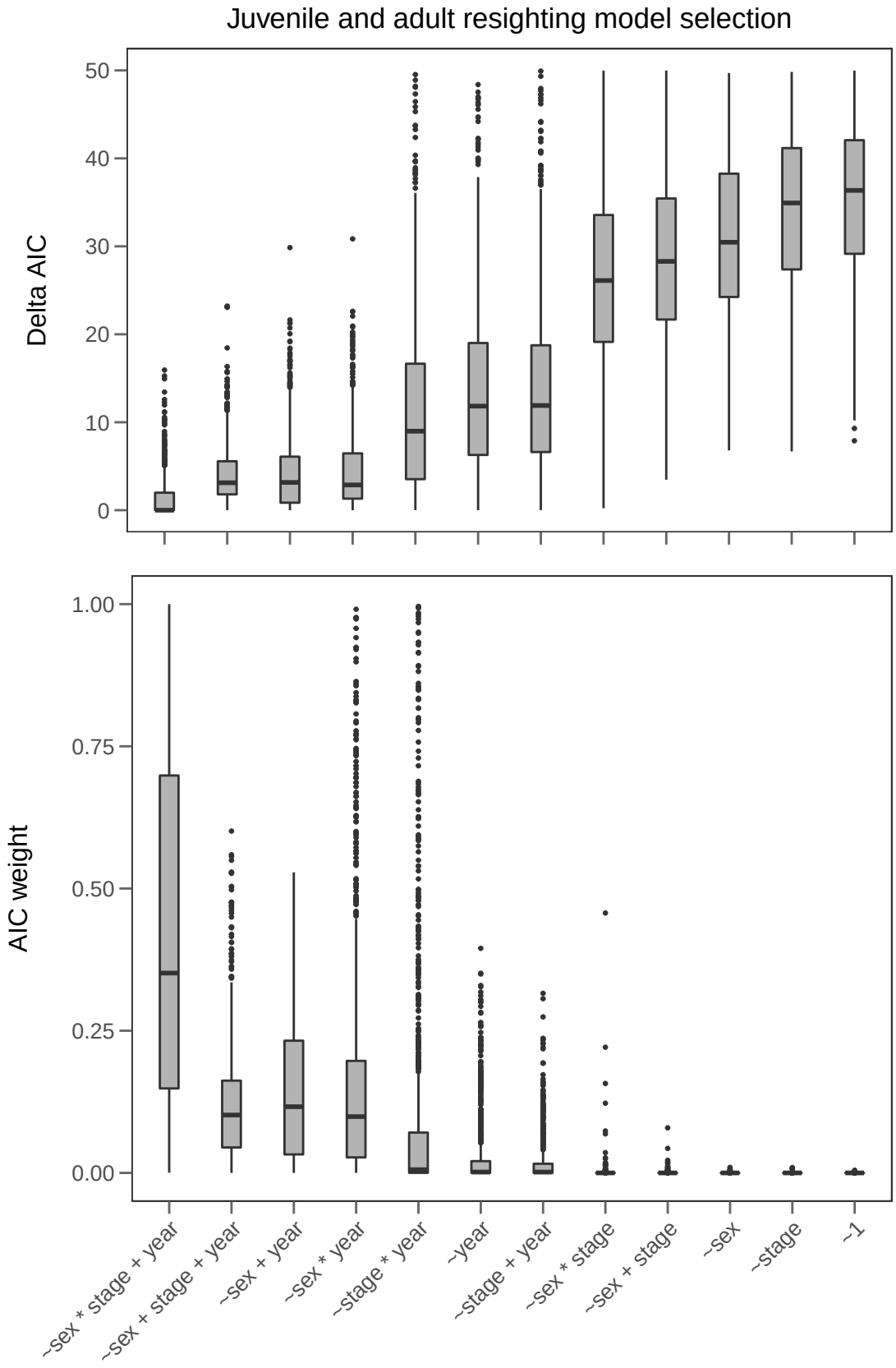
plot the overall model ranks of the juvenile and adult survival anlaysis based on Delta AIC

```
Bootstrap_Delta_AIC_plot_F_A <-
  ggplot(aes(y = DeltaAICc, x = p), data = juv_ad_AIC_tables) +
  theme_bw() +
  geom_boxplot(width = 0.3, fill = "grey70", outlier.size = 0.5) +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x  = element_blank(),
        axis.title.y = element_text(size=12, margin = margin(0, 18, 0, 0)),
        axis.text.y  = element_text(size=10),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        axis.ticks.x = element_line(size = 0.5, colour = "grey40"),
        plot.margin = unit(c(0.5,0.5,0,0.8), "cm"),
        panel.spacing = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank(),
        plot.title = element_text(hjust = 0.5)) +
  scale_y_continuous(limits=c(0,50)) +
  xlab("Model") +
  ylab("Delta AIC") +
  ggtitle("Juvenile and adult resighting model selection")
```

plot the overall model ranks of the juvenile and adult survival anlaysis based on AIC weight

```
Bootstrap_AIC_weight_plot_F_A <-
  ggplot(aes(y = weight, x = p), data = juv_ad_AIC_tables) +
  theme_bw() +
  geom_boxplot(width = 0.3, fill = "grey70", outlier.size = 0.5) +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x  = element_text(size=10, angle = 45, hjust = 1),
        axis.title.y = element_text(size=12, margin = margin(0, 18, 0, 0)),
        axis.text.y  = element_text(size=10),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        axis.ticks.x = element_line(size = 0.5, colour = "grey40"),
        plot.margin = unit(c(0.5,0.5,0.5,0.5), "cm"),
        panel.spacing = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank(),
        plot.background = element_rect(fill = "transparent",colour = NA)) +
  scale_y_continuous(limits=c(0,1)) +
  xlab("Model") +
  ylab("AIC weight") +
  scale_x_discrete(labels = c("~Time + age * sex" = "~sex * stage + year",
                              "~Time + age + sex" = "~sex + stage + year",
                              "~sex + Time" = "~sex + year",
                              "~sex * Time" = "~sex * year",
                              "~age * Time" = "~stage * year",
                              "~Time" = "~year",
```

```
                                "~age + Time" = "~stage + year",
                                "~age * sex" = "~sex * stage",
                                "~age + sex" = "~sex + stage",
                                "~sex" = "~sex",
                                "~age" = "~stage",
                                "~1" = "~1"))

grid.newpage()
pushViewport(viewport(layout = grid.layout(5, 1)))
vplayout <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)
print(Bootstrap_Delta_AIC_plot_F_A, vp = vplayout(1:2, 1))
print(Bootstrap_AIC_weight_plot_F_A, vp = vplayout(3:5, 1))
```


Juvenile and adult resighting model selection
Delta AIC
50
40
30
20
10
0
AIC weight
1.00
0.75
0.50
0.25
0.00
~sex * stage + year
~sex + stage + year
~sex + year
~sex * year
~stage * year
~year
~stage + year
~sex * stage
~sex + stage
~sex
~stage
~1

Life table response experiment on ASR

Perturbation analyses provide important information about the relative effect that each component of a matrix model has on the population-level response, in our case ASR. To assess how influential a sex bias in parameters associated with each of the three life stages was on ASR dynamics, we employed a life-table response experiment (LTRE). A LTRE decomposes the difference in response between two or more "treatments" by weighting the difference in parameter values by the parameter's contribution to the response (i.e. its sensitivity), and summing over all parameters (Caswell 2001). Here, we compared the observed scenario, to a hypothetical scenario whereby all female survival rates were set equal to the male rates and the hatching sex ratio was set to 0.5. Thus, our LTRE identifies the drivers of ASR bias by decomposing the difference between the ASR predicted by our model and an unbiased ASR (Veran & Beissinger 2009).

The following two functions need to be specified first:

sensitivity_analysis() determines the sensitivities of each parameter in the non-linear two-sex matrix model. It does this by perturbing each parameter independently and simulating the matrix until the stable stage is achieved and the ASR or lambda can be determined. After all perturbations have been tested, a spline of the response vs. perturbated values is found and the tangent of this spline at the observed parameter value is defined as a given parameter's sensitivity.

```
sensitivity_analysis <-
  function(vital_rates, matrix_str, h = 1, k = 3, HSR, niter = 1000, ASR, lambda){

    # make a list of all parameters
    vr <-
      list(F_Chk_survl = vital_rates$F_Chk_survl,
           F_Juv_survl = vital_rates$F_Juv_survl,
           F_Adt_survl = vital_rates$F_Adt_survl,
           M_Chk_survl = vital_rates$M_Chk_survl,
           M_Juv_survl = vital_rates$M_Juv_survl,
           M_Adt_survl = vital_rates$M_Adt_survl)

    # number of stages in the matrix
    no_stages <- sqrt(length(matrix_str))

    # Define plover life-stages of the Ceuta snowy plover matrix model
    stages <- c("F_1st_yr",  "F_Adt",  "M_1st_yr",  "M_Adt")

    # an empty t by x matrix
    stage <- matrix(numeric(no_stages * niter), nrow = no_stages)

    # an empty t vector to store the population sizes
    pop <- numeric(niter)

    # dataframe to store the perturbation results
    ASR_pert_results <-
      data.frame(parameter = c("F_Chk_survl", "F_Juv_survl", "F_Adt_survl",
                               "M_Chk_survl", "M_Juv_survl", "M_Adt_survl",
                               "h", "k", "HSR"),
                 sensitivities = numeric(9),
                 elasticities = numeric(9))

    lambda_pert_results <-
      data.frame(parameter = c("F_Chk_survl", "F_Juv_survl", "F_Adt_survl",
                               "M_Chk_survl", "M_Juv_survl", "M_Adt_survl",
```

```
                              "h", "k", "HSR"),
                sensitivities = numeric(9),
                elasticities = numeric(9))

  # specifiy how many survival rates there are
  n <- length(vr)

  # create vectors of perturbations to test on parameters of the matrix model
  vr_nums <- seq(0, 1, 0.01) # proportional changes in survival and HSR (i.e., between 0 an 1)
  h_nums <- seq(0, 2, 0.02) # proportional changes in h index (i.e., between 0 and 2)
  k_nums <- seq(2, 4, 0.02) # proportional changes in k (i.e, between 2 and 4)

  # create empty dataframes to store the perturbation results for ASR and lambda
  vr_pert_ASR <- matrix(numeric(n * length(vr_nums)),
                    ncol = n, dimnames = list(vr_nums, names(vr)))
  h_pert_ASR <- matrix(numeric(length(h_nums)),
                   ncol = 1, dimnames = list(h_nums, "h"))
  k_pert_ASR <- matrix(numeric(length(k_nums)),
                   ncol = 1, dimnames = list(k_nums, "k"))
  HSR_pert_ASR <- matrix(numeric(length(vr_nums)),
                     ncol = 1, dimnames = list(vr_nums, "HSR"))
  vr_pert_lambda <- matrix(numeric(n * length(vr_nums)),
                       ncol = n, dimnames = list(vr_nums, names(vr)))
  h_pert_lambda <- matrix(numeric(length(h_nums)),
                      ncol = 1, dimnames = list(h_nums, "h"))
  k_pert_lambda <- matrix(numeric(length(k_nums)),
                      ncol = 1, dimnames = list(k_nums, "k"))
  HSR_pert_lambda <- matrix(numeric(length(vr_nums)),
                        ncol = 1, dimnames = list(vr_nums, "HSR"))

  # perturbation of vital rates survival rates
  for (g in 1:n) # pick a column (i.e., a variable)
  {
    vr2 <- vr # reset the vital rates to the original
    for (i in 1:length(vr_nums)) # pick a perturbation level
    {
      vr2[[g]] <- vr_nums[i] # specify the vital rate with the new perturbation level
      A <- matrix(sapply(matrix_str, eval, vr2, NULL),
                  nrow = sqrt(length(matrix_str)), byrow=TRUE,
                  dimnames = list(stages, stages)) # build the matrix with the new value
      # reset the starting stage distribution for simulation (all with 10 individuals)
      m <- rep(10, no_stages)
      for (j in 1:niter) { # project the matrix through t iteration
        # stage distribution at time t
        stage[,j] <- m
        # population size at time t
        pop[j] <- sum(m)
        # number of male adults at time t
        M2 <- stage[4, j]
        # number of female adults at time t
        F2 <- stage[2, j]
        # Female freq-dep fecundity of Female chicks
        A[1,no_stages/2]        <- ((k*M2)/(M2+(F2/h)))*HSR
```

```
      # Female freq-dep fecundity of Male chicks
      A[(no_stages/4)*3,no_stages/2]  <- ((k*M2)/(M2+(F2/h)))*HSR
      # Male freq-dep fecundity of Female chicks
      A[1,no_stages]          <- ((k*F2)/(M2+(F2/h)))*HSR
      # Male freq-dep fecundity of Male chicks
      A[(no_stages/4)*3,no_stages]    <- ((k*F2)/(M2+(F2/h)))*HSR
      # define the new n (i.e., new stage distribution at time t)
      m <- A %*% m
    }
    # define rownames of stage matrix
    rownames(stage) <- rownames(A)
    # define colnames of stage matrix
    colnames(stage) <- 0:(niter - 1)
    # calculate the proportional stable stage distribution
    stage <- apply(stage, 2, function(x) x/sum(x))
    # define stable stage as the last stage
    stable.stage <- stage[, niter]
    # calc ASR as the proportion of the adult stable stage class that is male
    vr_pert_ASR[i, g] <- stable.stage[no_stages]/(stable.stage[no_stages/2] +
                                                    stable.stage[no_stages])
    # calc lambda as the pop change in the counts of the last two iterations
    vr_pert_lambda[i, g] <- pop[niter]/pop[niter - 1]
  }
  # get the spline function of ASR
  spl_ASR <- smooth.spline(vr_pert_ASR[,g] ~ rownames(vr_pert_ASR))
  # estimate the slope of the tangent of the spline at the vital rate
  ASR_pert_results[g, 2] <- predict(spl_ASR, x=vr[[g]], deriv=1)$y
  # re-scale sensitivity into elasticity
  ASR_pert_results[g, 3] <- vr[[g]]/ASR * ASR_pert_results[g, 2]
  # do the same steps but for lambda
  spl_lambda <- smooth.spline(vr_pert_lambda[,g] ~ rownames(vr_pert_lambda))
  lambda_pert_results[g, 2] <- predict(spl_lambda, x=vr[[g]], deriv=1)$y
  lambda_pert_results[g, 3] <- vr[[g]]/lambda * lambda_pert_results[g, 2]
}
# perturbation of the h index parameter
for (i in 1:length(h_nums)) # pick a perturbation level
{
  A <- matrix(sapply(matrix_str, eval, vr, NULL),
              nrow = sqrt(length(matrix_str)), byrow=TRUE,
              dimnames = list(stages, stages)) # build the matrix with the new value
  # reset the starting stage distribution for simulation (all with 10 individuals)
  m <- rep(10, no_stages)
  for (j in 1:niter) { # project the matrix through t iteration
    # stage distribution at time t
    stage[,j] <- m
    # population size at time t
    pop[j] <- sum(m)
    # number of male adults at time t
    M2 <- stage[4, j]
    # number of female adults at time t
    F2 <- stage[2, j]
    # Female freq-dep fecundity of Female chicks
    A[1,no_stages/2]        <- ((k*M2)/(M2+(F2/h_nums[i])))*HSR
```

```
    # Female freq-dep fecundity of Male chicks
    A[(no_stages/4)*3,no_stages/2] <- ((k*M2)/(M2+(F2/h_nums[i])))*HSR
    # Male freq-dep fecundity of Female chicks
    A[1,no_stages]          <- ((k*F2)/(M2+(F2/h_nums[i])))*HSR
    # Male freq-dep fecundity of Male chicks
    A[(no_stages/4)*3,no_stages] <- ((k*F2)/(M2+(F2/h_nums[i])))*HSR
    # define the new n (i.e., new stage distribution at time t)
    m <- A %*% m
  }
  # define rownames of stage matrix
  rownames(stage) <- rownames(A)
  # define colnames of stage matrix
  colnames(stage) <- 0:(niter - 1)
  # calculate the proportional stable stage distribution
  stage <- apply(stage, 2, function(x) x/sum(x))
  # define stable stage as the last stage
  stable.stage <- stage[, niter]
  # calc ASR as the proportion of the adult stable stage class that is male
  h_pert_ASR[i,] <- stable.stage[no_stages]/(stable.stage[no_stages/2] + stable.stage[no_stages])
  # calc lambda as the pop change in the counts of the last two iterations
  h_pert_lambda[i, ] <- pop[niter]/pop[niter - 1]
}
# get the spline function of ASR
spl_ASR <- smooth.spline(h_pert_ASR[, 1] ~ rownames(h_pert_ASR))
# estimate the slope of the tangent of the spline at the vital rate
ASR_pert_results[n+1, 2] <- predict(spl_ASR, x=h, deriv=1)$y
# re-scale sensitivity into elasticity
ASR_pert_results[n+1, 3] <- h/ASR * ASR_pert_results[n+1, 2]
# do the same steps but for lambda
spl_lambda <- smooth.spline(h_pert_lambda[,1] ~ rownames(h_pert_lambda))
lambda_pert_results[n+1, 2] <- predict(spl_lambda, x=h, deriv=1)$y
lambda_pert_results[n+1, 3] <- h/lambda * lambda_pert_results[n+1, 2]

# perturbation of k parameter
for (i in 1:length(k_nums)) # pick a perturbation level
{
  A <- matrix(sapply(matrix_str, eval, vr, NULL),
              nrow = sqrt(length(matrix_str)), byrow=TRUE,
              dimnames = list(stages, stages)) # build the matrix with the new value
  # reset the starting stage distribution for simulation (all with 10 individuals)
  m <- rep(10, no_stages)
  for (j in 1:niter) { # project the matrix through t iteration
    # stage distribution at time t
    stage[,j] <- m
    # population size at time t
    pop[j] <- sum(m)
    # number of male adults at time t
    M2 <- stage[4, j]
    # number of female adults at time t
    F2 <- stage[2, j]
    # Female freq-dep fecundity of Female chicks
    A[1,no_stages/2]        <- ((k_nums[i]*M2)/(M2+(F2/h)))*HSR
    # Female freq-dep fecundity of Male chicks
```

```
    A[(no_stages/4)*3,no_stages/2] <- ((k_nums[i]*M2)/(M2+(F2/h)))*HSR
    # Male freq-dep fecundity of Female chicks
    A[1,no_stages]          <- ((k_nums[i]*F2)/(M2+(F2/h)))*HSR
    # Male freq-dep fecundity of Male chicks
    A[(no_stages/4)*3,no_stages] <- ((k_nums[i]*F2)/(M2+(F2/h)))*HSR
    # define the new n (i.e., new stage distribution at time t)
    m <- A %*% m
  }
  # define rownames of stage matrix
  rownames(stage) <- rownames(A)
  # define colnames of stage matrix
  colnames(stage) <- 0:(niter - 1)
  # calculate the proportional stable stage distribution
  stage <- apply(stage, 2, function(x) x/sum(x))
  # define stable stage as the last stage
  stable.stage <- stage[, niter]
  # calc ASR as the proportion of the adult stable stage class that is male
  k_pert_ASR[i,] <- stable.stage[no_stages]/(stable.stage[no_stages/2] +
                                               stable.stage[no_stages])
  # calc lambda as the pop change in the counts of the last two iterations
  k_pert_lambda[i, ] <- pop[niter]/pop[niter - 1]
}
# get the spline function of ASR
spl_ASR <- smooth.spline(k_pert_ASR[,1] ~ rownames(k_pert_ASR))
# estimate the slope of the tangent of the spline at the vital rate
ASR_pert_results[n+2, 2] <- predict(spl_ASR, x=k, deriv=1)$y
# re-scale sensitivity into elasticity
ASR_pert_results[n+2, 3] <- k/ASR * ASR_pert_results[n+2, 2]
# do the same steps but for lambda
spl_lambda <- smooth.spline(k_pert_lambda[,1] ~ rownames(k_pert_lambda))
lambda_pert_results[n+2, 2] <- predict(spl_lambda, x=k, deriv=1)$y
lambda_pert_results[n+2, 3] <- k/lambda * lambda_pert_results[n+2, 2]
# perturbation of HSR
for (i in 1:length(vr_nums)) # pick a perturbation level
{
  A <- matrix(sapply(matrix_str, eval, vr, NULL),
              nrow = sqrt(length(matrix_str)), byrow=TRUE,
              dimnames = list(stages, stages)) # build the matrix with the new value
  # reset the starting stage distribution for simulation (all with 10 individuals)
  m <- rep(10, no_stages)
  for (j in 1:niter) { # project the matrix through t iteration
    # stage distribution at time t
    stage[,j] <- m
    # population size at time t
    pop[j] <- sum(m)
    # number of male adults at time t
    M2 <- stage[4, j]
    # number of female adults at time t
    F2 <- stage[2, j]
    # Female freq-dep fecundity of Female chicks
    A[1,no_stages/2]        <- ((k*M2)/(M2+(F2/h)))*vr_nums[i]
    # Female freq-dep fecundity of Male chicks
    A[(no_stages/4)*3,no_stages/2] <- ((k*M2)/(M2+(F2/h)))*vr_nums[i]
```

```
      # Male freq-dep fecundity of Female chicks
      A[1,no_stages]          <- ((k*F2)/(M2+(F2/h)))*vr_nums[i]
      # Male freq-dep fecundity of Male chicks
      A[(no_stages/4)*3,no_stages] <- ((k*F2)/(M2+(F2/h)))*vr_nums[i]
      # define the new n (i.e., new stage distribution at time t)
      m <- A %*% m
    }
    # define rownames of stage matrix
    rownames(stage) <- rownames(A)
    # define colnames of stage matrix
    colnames(stage) <- 0:(niter - 1)
    # calculate the proportional stable stage distribution
    stage <- apply(stage, 2, function(x) x/sum(x))
    # define stable stage as the last stage
    stable.stage <- stage[, niter]
    # calc ASR as the proportion of the adult stable stage class that is male
    HSR_pert_ASR[i,] <- stable.stage[no_stages]/(stable.stage[no_stages/2] +
                                                   stable.stage[no_stages])
    # calc lambda as the pop change in the counts of the last two iterations
    HSR_pert_lambda[i, ] <- pop[niter]/pop[niter - 1]
  }
  # get the spline function of ASR
  spl_ASR <- smooth.spline(HSR_pert_ASR[,1] ~ rownames(HSR_pert_ASR))
  # estimate the slope of the tangent of the spline at the vital rate
  ASR_pert_results[n+3, 2] <- predict(spl_ASR, x=HSR, deriv=1)$y
  # re-scale sensitivity into elasticity
  ASR_pert_results[n+3, 3] <- HSR/ASR * ASR_pert_results[n+3, 2]
  # do the same steps but for lambda
  spl_lambda <- smooth.spline(HSR_pert_lambda[,1] ~ rownames(HSR_pert_lambda))
  lambda_pert_results[n+3, 2] <- predict(spl_lambda, x=HSR, deriv=1)$y
  lambda_pert_results[n+3, 3] <- HSR/lambda * lambda_pert_results[n+3, 2]
  # store all results into a list
  result <- list(ASR_pert_results = ASR_pert_results,
                 lambda_pert_results = lambda_pert_results)
}
```

LTRE_analysis() estimates the contribution that each vital rate has on ASR bias, given the sensitivities calculated in the previous function (see formula 8 on page 133 of Veran and Beissinger (2009))

```
LTRE_analysis <-
  function(Mprime_sens, matrix_str, vital_rates){

    # make empty dataframes to stroe LTRE results for ASR and lambda
    LTRE_ASR <-
      data.frame(parameter = c("Chick survival", "Juvenile survival",
                               "Adult survival", "Hatching sex ratio",
                               "Mating system"),
                 contribution = numeric(5))

    LTRE_lambda <-
      data.frame(parameter = c("Chick survival", "Juvenile survival",
                               "Adult survival", "Hatching sex ratio",
                               "Mating system"),
                 contribution = numeric(5))
```

```
  # run a for loop to extract the parameter contributions
  for(i in 1:nrow(LTRE_ASR))
  {
    LTRE_ASR[i, 2] <-
      ifelse(i < 4, (vital_rates[[i + 3]] - vital_rates[[i]]) *
               Mprime_sens$ASR_pert_results$sensitivities[i + 3],
             ifelse(i == 4, ((1-vital_rates[[9]]) - vital_rates[[9]]) *
                      Mprime_sens$ASR_pert_results$sensitivities[9],
                    (1 - vital_rates[[7]]) * Mprime_sens$ASR_pert_results$sensitivities[7]))
  }

  for(i in 1:nrow(LTRE_lambda))
  {
    LTRE_lambda[i, 2] <-
      ifelse(i < 4, (vital_rates[[i + 3]] - vital_rates[[i]]) *
               Mprime_sens$lambda_pert_results$sensitivities[i + 3],
             ifelse(i == 4, (vital_rates[[9]] - (1-vital_rates[[9]])) *
                      Mprime_sens$lambda_pert_results$sensitivities[9],
                    (vital_rates[[7]] - 1) * Mprime_sens$lambda_pert_results$sensitivities[7]))
  }

  LTRE_ASR$parameter <- factor(LTRE_ASR$parameter, levels = c("Adult survival",
                                                              "Juvenile survival",
                                                              "Chick survival",
                                                              "Hatching sex ratio",
                                                              "Mating system"))

  LTRE_lambda$parameter <- factor(LTRE_lambda$parameter, levels = c("Adult survival",
                                                                    "Juvenile survival",
                                                                    "Chick survival",
                                                                    "Hatching sex ratio",
                                                                    "Mating system"))

  LTRE_results <- list(LTRE_ASR = LTRE_ASR,
                       LTRE_lambda = LTRE_lambda)
}
```

define the iterations variable as a factor

```
survival_rates_boot$iter <- as.factor(survival_rates_boot$iter)
```

summarise the bootstrap stage- and sex-specific survival rates for the deterministic matrix

```
survival_rates_boot_summary <-
  Rmisc::summarySE(survival_rates_boot,
                   measurevar = "estimate",
                   groupvars = c("sex_stage"),
                   conf.interval = 0.95)
```

define deteriministic Ceuta vital rates estimated from mark-recapture analysis. This are the "treatment" rates observed in the field:

```
VR_treat <- list(F_Chk_survl = survival_rates_boot_summary[2,3],
                 F_Juv_survl = survival_rates_boot_summary[3,3],
                 F_Adt_survl = survival_rates_boot_summary[1,3],
```

```
                  M_Chk_survl = survival_rates_boot_summary[5,3],
                  M_Juv_survl = survival_rates_boot_summary[6,3],
                  M_Adt_survl = survival_rates_boot_summary[4,3],
                  # Define h (harem size, h = 1 is monogamy) and k (clutch size)
                  h = h,
                  k = 3,
                  # Define primary sex ratio
                  HSR = HSR)
```

Define vital rates of the M prime matrix (i.e., average between a "control matrix" and the "treatment matrix"). The control matrix is a matrix in which the female vital rates are set to the male vital rates, and the treatment matrix is the matrix containing the sex-specific values estimated from the field (see formula 8 on page 133 of Veran and Beissinger (2009)). The M-prime matrix is the average matrix of the treatment and control matricies:

```
VR_mprime <- list(F_Chk_survl = (survival_rates_boot_summary[2,3] +
                                   survival_rates_boot_summary[5,3])/2,
                  F_Juv_survl = (survival_rates_boot_summary[3,3] +
                                   survival_rates_boot_summary[6,3])/2,
                  F_Adt_survl = (survival_rates_boot_summary[1,3] +
                                   survival_rates_boot_summary[4,3])/2,
                  M_Chk_survl = (survival_rates_boot_summary[5,3] +
                                   survival_rates_boot_summary[5,3])/2,
                  M_Juv_survl = (survival_rates_boot_summary[6,3] +
                                   survival_rates_boot_summary[6,3])/2,
                  M_Adt_survl = (survival_rates_boot_summary[4,3] +
                                   survival_rates_boot_summary[4,3])/2,
                  # Define h (harem size, h = 1 is monogamy) and k (clutch size)
                  h = (h+1)/2,
                  k = 3,
                  # Define primary sex ratio
                  HSR = (HSR+0.5)/2)
```

specify the struture of the matrix (i.e. show the lower-level element functions)

```
matrix_structure <- expression(
                                # top row of matrix
                                0, NA, 0, NA,

                                # second row of matrix
                                (F_Chk_survl * F_Juv_survl), F_Adt_survl, 0, 0,

                                # third row of matrix
                                0, NA, 0, NA,

                                # fourth row of matrix
                                0, 0, (M_Chk_survl * M_Juv_survl), M_Adt_survl

                                )
```

build the treatment matrix

```
treatment_matrix <- plover_matrix(VR_treat)
```

build the M-prime matrix

```
M_prime_matrix <- plover_matrix(VR_mprime)
```

determine the ASR at the stable stage distribution

```
treatment_ASR_analysis <-
  matrix_ASR(M = treatment_matrix, h = h, HSR = VR_treat$HSR, iterations = 1000)
ASR_treat <- treatment_ASR_analysis$ASR
ASR_treat
#>     M_Adt
#> 0.6328887

M_prime_ASR_analysis <-
  matrix_ASR(M = M_prime_matrix, h = 1, HSR = VR_mprime$HSR, iterations = 1000)
ASR_mprime <- M_prime_ASR_analysis$ASR
ASR_mprime
#>     M_Adt
#> 0.5618169
```

specify the lambda of the matrices

```
lambda_treat <- treatment_ASR_analysis$lambda
lambda_treat
#> [1] 0.8511694

lambda_mprime <- M_prime_ASR_analysis$lambda
lambda_mprime
#> [1] 0.8686893
```

conduct a sensitivity analysis on the treatment matrix

```
treat_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = VR_treat,
                       matrix_str = matrix_structure,
                       h = VR_treat$h,
                       k = VR_treat$k,
                       HSR = VR_treat$HSR,
                       niter = 1000,
                       ASR = ASR_treat,
                       lambda = lambda_treat)
```

conduct a sensitivity analysis on the M-Prime matrix

```
Mprime_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = VR_mprime,
                       matrix_str = matrix_structure,
                       h = VR_mprime$h,
                       k = VR_mprime$k,
                       HSR = VR_mprime$HSR,
                       niter = 1000,
                       ASR = ASR_mprime,
                       lambda = lambda_mprime)
```

conduct the LTRE comparing the two matrices

```
LTRE_plover <-
  LTRE_analysis(Mprime_sens = Mprime_sensitivity_analysis,
                matrix_str = matrix_str,
                vital_rates = VR_treat)
```

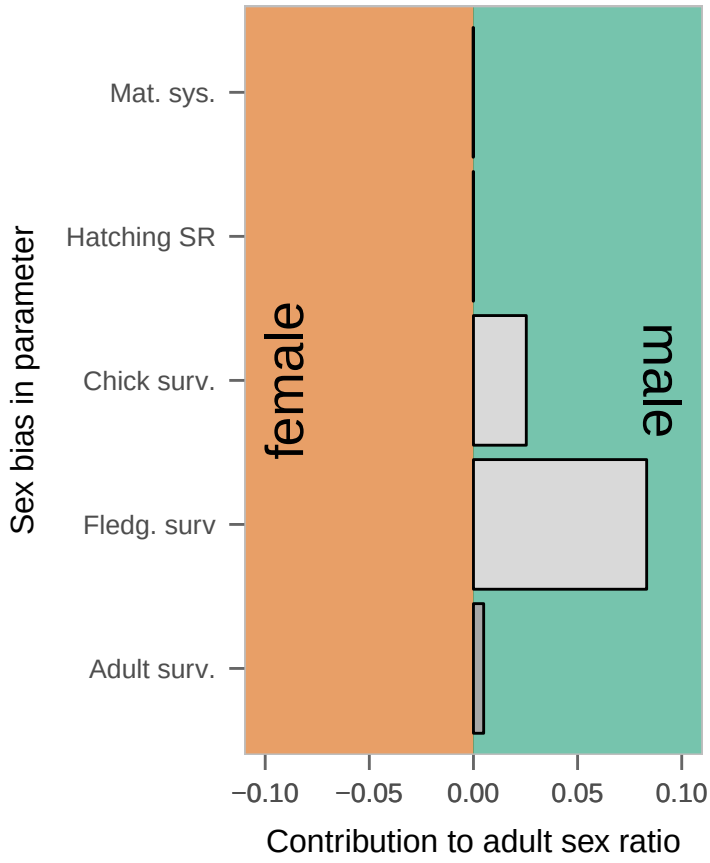
custom color palette for the plotting of Juvenile and Adult stats

```
cbPalette <- c("#A6A6A6", "#D9D9D9", "#D9D9D9", "#D9D9D9", "#A6A6A6")
```

Figure 2c: plot the comparative LTRE results

```
Background_LTRE_ASR <-
  ggplot2::ggplot(data = LTRE_plover$LTRE_ASR,
                  aes(x = parameter, y = contribution, fill = parameter)) +
  coord_flip() +
  theme_bw() +
  annotate("rect", xmin=-Inf, xmax=Inf, ymin=-Inf, ymax=0, alpha=0.6,
           fill=brewer.pal(8, "Dark2")[c(2)]) +
  annotate("rect", xmin=-Inf, xmax=Inf, ymin=0, ymax=Inf, alpha=0.6,
           fill=brewer.pal(8, "Dark2")[c(1)]) +
  annotate("text", x = c(2,2), y = c(-0.1, 0.1),
           label = c("female", "male"), size = 7,
           vjust = c(1,1), hjust = c(0.5,0.5), angle = c(90, 270)) +
  theme(text = element_text(color = "white"),
        legend.position = "none",
        axis.title.x = element_text(size=12, margin = margin(10, 0, 0, 0)),
        axis.text.x  = element_text(size=10, margin = margin(5, 0, 0, 0)),
        axis.title.y = element_text(size=12, margin = margin(0, 10, 0, 0)),
        axis.text.y  = element_text(size=10, angle = 0, hjust = 1,
                                    margin = margin(0, 5, 0, 0)),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_blank(),
        axis.ticks.x = element_line(size = 0.5, colour = "white"),
        axis.ticks.length = unit(0.2, "cm"),
        panel.border = element_blank(),
        plot.margin = unit(c(1,0.5,0.5,2.85), "cm"),
        panel.spacing = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  ylab("Contribution to adult sex ratio") +
  xlab("Sex bias in parameter") +
  scale_fill_manual(values = cbPalette) +
  scale_y_continuous(limits = c(-0.1, 0.1)) +
  scale_x_discrete(labels = c("Adult survival" = "Adult surv.",
                              "Juvenile survival" = "Fledg. surv",
                              "Chick survival" = "Chick surv.",
                              "Hatching sex ratio" = "Hatching SR",
                              "Mating system" = "Mat. sys."))

LTRE_ASR <-
  ggplot2::ggplot() +
  theme_bw() +
  coord_flip() +
  geom_bar(data = LTRE_plover$LTRE_ASR,
           aes(x = parameter, y = contribution, fill = parameter), color = "black", stat = "identity") +
  theme(legend.position = "none",
        panel.background = element_rect(fill = "transparent",colour = NA),
        plot.background = element_rect(fill = "transparent",colour = NA),
```

```
        axis.title.x = element_text(size=12, margin = margin(10, 0, 0, 0)),
        axis.text.x  = element_text(size=10, margin = margin(5, 0, 0, 0)),
        axis.title.y = element_text(size=12, margin = margin(0, 10, 0, 0)),
        axis.text.y  = element_text(size=10, angle = 0, hjust = 1,
                                    margin = margin(0, 5, 0, 0)),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        axis.ticks.x = element_line(size = 0.5, colour = "grey40"),
        panel.border = element_rect(linetype = "solid", colour = "grey"),
        plot.margin = unit(c(1,0.5,0.5,0.5), "cm"),
        panel.spacing = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  ylab("Contribution to adult sex ratio") +
  xlab("Sex bias in parameter") +
  scale_fill_manual(values = cbPalette) +
  scale_y_continuous(limits = c(-0.10, 0.10)) +
  scale_x_discrete(labels = c("Adult survival" = "Adult surv.",
                              "Juvenile survival" = "Fledg. surv",
                              "Chick survival" = "Chick surv.",
                              "Hatching sex ratio" = "Hatching SR",
                              "Mating system" = "Mat. sys."))

# draw the background and the LTRE on top of eachother for the final plot
grid.newpage()
pushViewport(viewport(layout = grid.layout(1, 1, widths = unit(1, "npc"))))
print(Background_LTRE_ASR + theme(legend.position="none"),
      vp = viewport(layout.pos.row = 1 ,layout.pos.col = 1))
print(LTRE_ASR + theme(legend.position="none"),
      vp = viewport(layout.pos.row = 1, layout.pos.col = 1))
```



Determine how much larger the contribution of each vital rates is compared to juvenile survival juvenile vs chick:

```
LTRE_plover$LTRE_ASR[2,2]/LTRE_plover$LTRE_ASR[1,2]
#> [1] 3.278018
```

juvenile vs adult:

```
LTRE_plover$LTRE_ASR[2,2]/LTRE_plover$LTRE_ASR[3,2]
#> [1] 16.9537
```

chick vs adult:

```
LTRE_plover$LTRE_ASR[1,2]/LTRE_plover$LTRE_ASR[3,2]
#> [1] 5.171937
```

Sensitivity analysis of population growth

Biased ASR and polygamous mating systems can restrict the reproductive potential of a population due to a scarcity of the limiting sex (Caswell & Weeks 1986). Thus, population viability can be indirectly affected by ASR and mating system via the sex-specific effects that vital rates have on population growth under a biased ASR or a polygamous mating system, or both (Haridas et al. 2014). To investigate the relative influence that a biased ASR or a polygamous mating system has on population growth, we conducted a sensitivity analysis of all sex-specific parameters using four scenarios of the two-sex model: (i) polyandrous and male-biased ASR (i.e. the observed scenario), (ii) polyandrous and unbiased ASR, (iii) monogamous and male-biased ASR, and (iv) monogamous and unbiased ASR.

Here are the four scenarios and the structure of their vital rates:

```
VR_equal_mono <- list(F_Chk_survl = survival_rates_boot_summary[5,3],
                      F_Juv_survl = survival_rates_boot_summary[6,3],
                      F_Adt_survl = survival_rates_boot_summary[4,3],
                      M_Chk_survl = survival_rates_boot_summary[5,3],
                      M_Juv_survl = survival_rates_boot_summary[6,3],
                      M_Adt_survl = survival_rates_boot_summary[4,3],
                      # Define h (harem size, h = 1 is monogamy) and k (clutch size)
                      h = 1,
                      k = 3,
                      # Define primary sex ratio
                      HSR = 0.5)

VR_equal_poly <- list(F_Chk_survl = survival_rates_boot_summary[5,3],
                      F_Juv_survl = survival_rates_boot_summary[6,3],
                      F_Adt_survl = survival_rates_boot_summary[4,3],
                      M_Chk_survl = survival_rates_boot_summary[5,3],
                      M_Juv_survl = survival_rates_boot_summary[6,3],
                      M_Adt_survl = survival_rates_boot_summary[4,3],
                      # Define h (harem size, h = 1 is monogamy) and k (clutch size)
                      h = h,
                      k = 3,
                      # Define primary sex ratio
                      HSR = 0.5)

VR_diff_mono <- list(F_Chk_survl = survival_rates_boot_summary[2,3],
                     F_Juv_survl = survival_rates_boot_summary[3,3],
                     F_Adt_survl = survival_rates_boot_summary[1,3],
                     M_Chk_survl = survival_rates_boot_summary[5,3],
                     M_Juv_survl = survival_rates_boot_summary[6,3],
                     M_Adt_survl = survival_rates_boot_summary[4,3],
                     # Define h (harem size, h = 1 is monogamy) and k (clutch size)
                     h = 1,
                     k = 3,
                     # Define primary sex ratio
                     HSR = 0.5)
```

Convert the vital rates lists into the two-sex ptrojection matrix using the **plover_matrix()** function

```
equal_mono_mat <- plover_matrix(VR_equal_mono)
equal_poly_mat <- plover_matrix(VR_equal_poly)
diff_mono_mat <- plover_matrix(VR_diff_mono)
```

Derive equilibrium statistics (i.e., ASR and lambda) of each scenario using the **matrix_ASR()** function

```
equal_mono_ASR_analysis <-
  matrix_ASR(M = equal_mono_mat, h = 1, HSR = 0.5, iterations = 1000)
ASR_equal_mono <- equal_mono_ASR_analysis$ASR
ASR_equal_mono
#> M_Adt
#>   0.5

equal_poly_ASR_analysis <-
  matrix_ASR(M = equal_poly_mat, h = h, HSR = 0.5, iterations = 1000)
ASR_equal_poly <- equal_poly_ASR_analysis$ASR
```

```
ASR_equal_poly
#> M_Adt
#>   0.5

diff_mono_ASR_analysis <-
  matrix_ASR(M = diff_mono_mat, h = 1, HSR = 0.5, iterations = 1000)
ASR_diff_mono <- diff_mono_ASR_analysis$ASR
ASR_diff_mono
#>     M_Adt
#> 0.6330502
```

Specify the lambda of each scenario

```
lambda_equal_mono <- equal_mono_ASR_analysis$lambda
lambda_equal_mono
#> [1] 0.8914422

lambda_equal_poly <- equal_poly_ASR_analysis$lambda
lambda_equal_poly
#> [1] 0.9071447

lambda_diff_mono <- diff_mono_ASR_analysis$lambda
lambda_diff_mono
#> [1] 0.8460346
```

Run the sensitivity analysis on each scenario

```
equal_mono_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = VR_equal_mono,
                       matrix_str = matrix_structure,
                       h = VR_equal_mono$h,
                       k = VR_equal_mono$k,
                       HSR = VR_equal_mono$HSR,
                       niter = 1000,
                       ASR = ASR_equal_mono,
                       lambda = lambda_equal_mono)

equal_poly_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = VR_equal_poly,
                       matrix_str = matrix_structure,
                       h = VR_equal_poly$h,
                       k = VR_equal_poly$k,
                       HSR = VR_equal_poly$HSR,
                       niter = 1000,
                       ASR = ASR_equal_poly,
                       lambda = lambda_equal_poly)

diff_mono_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = VR_diff_mono,
                       matrix_str = matrix_structure,
                       h = VR_diff_mono$h,
                       k = VR_diff_mono$k,
                       HSR = VR_diff_mono$HSR,
                       niter = 1000,
                       ASR = ASR_diff_mono,
```

```
                        lambda = lambda_diff_mono)
```

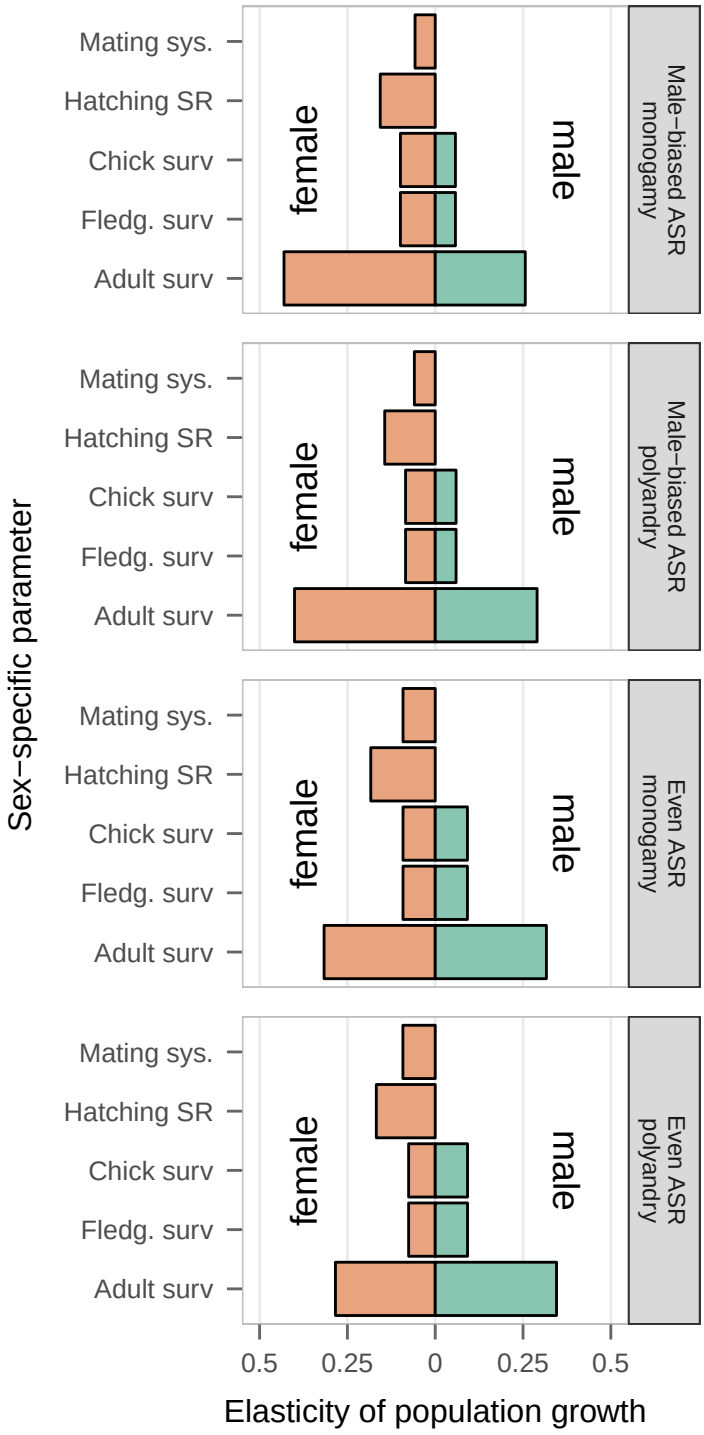
Tidy up the dataframe for plotting

```
# name each scenario
equal_mono_sensitivity_analysis$lambda_pert_results$model <- "equal_mono"
equal_poly_sensitivity_analysis$lambda_pert_results$model <- "equal_poly"
diff_mono_sensitivity_analysis$lambda_pert_results$model <- "diff_mono"
treat_sensitivity_analysis$lambda_pert_results$model <- "diff_poly"

# bind them together
lambda_elasticities <- rbind(equal_mono_sensitivity_analysis$lambda_pert_results[-8,-2],
                             equal_poly_sensitivity_analysis$lambda_pert_results[-8,-2],
                             diff_mono_sensitivity_analysis$lambda_pert_results[-8,-2],
                             treat_sensitivity_analysis$lambda_pert_results[-8,-2])

# define the vital rates for males and females
lambda_elasticities$vital_rate <-
  as.factor(c("Chick_survival", "Juvenile_survival", "Adult_survival", "Chick_survival",
              "Juvenile_survival", "Adult_survival", "Mating_system", "Hatching_sex_ratio"))

# classify the levels of the vital rates
lambda_elasticities$vital_rate <- factor(lambda_elasticities$vital_rate,
                                    levels = c("Adult_survival", "Juvenile_survival",
                                               "Chick_survival", "Hatching_sex_ratio",
                                               "Mating_system"))

# define the sex that each vital rate is representing
lambda_elasticities$sex <- as.factor(c("Female", "Female",
                                       "Female", "Male",
                                       "Male", "Male",
                                       "Female", "Female"))

# create a unique name for each scenario and vital rate combination
lambda_elasticities$model_vital_rate <- as.factor(paste(lambda_elasticities$model,
                                                        lambda_elasticities$vital_rate, sep = "_"))

# classify the levels of the scenarios
lambda_elasticities$model <- factor(lambda_elasticities$model,
                                    levels = c("diff_mono", "diff_poly", "equal_mono", "equal_poly"))
```

Figure 3: plot the elasticities of lambda to perturbations of each parameter under the four scenarios

```
# define the color palette to use in plotting
cbPalette <- c("#e8a47e", "#88c6b2")

# specify the facet labels
scenario_names <- c(
                    `diff_mono` = "Male-biased ASR\nmonogamy",
                    `diff_poly` = "Male-biased ASR\npolyandry",
                    `equal_mono` = "Even ASR\nmonogamy",
                    `equal_poly` = "Even ASR\npolyandry"
                    )

lambda_elasticity <-
```

```
 ggplot2::ggplot() +
        theme_bw() +
        coord_flip() +
        geom_bar(data = filter(lambda_elasticities, sex == "Female"),
                 aes(x = vital_rate, y = elasticities*-1, fill = sex),
                 color = "black", stat = "identity", alpha = 1) +
        geom_bar(data = filter(lambda_elasticities, sex == "Male"),
                 aes(x = vital_rate, y = elasticities, fill = sex),
                 color = "black", stat = "identity", alpha = 1) +
        annotate("text", x = c(3), y = c(-0.375),
                 label = c("female"),
                 vjust = c(0.5), angle = 90, size = 5) +
        annotate("text", x = c(3), y = c(0.375),
                 label = c("male"),
                 vjust = c(0.5), angle = 270, size = 5) +
        facet_grid(model ~ ., labeller = as_labeller(scenario_names)) +
        theme(legend.position = "none",
              panel.background = element_rect(fill = "transparent", colour = NA),
              plot.background = element_rect(fill = "transparent", colour = NA),
              axis.title.x = element_text(size=12, margin = margin(10, 0, 0, 0)),
              axis.text.x  = element_text(size=10, margin = margin(5, 0, 0, 0)),
              axis.title.y = element_text(size=12, margin = margin(0, 10, 0, 0)),
              axis.text.y  = element_text(size=10, angle = 0, hjust = 1,
                                          margin = margin(0, 5, 0, 0)),
              panel.grid.major.y = element_blank(),
              panel.grid.minor = element_blank(),
              axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
              axis.ticks.length = unit(0.2, "cm"),
              axis.ticks.x = element_line(size = 0.5, colour = "grey40"),
              panel.border = element_rect(linetype = "solid", colour = "grey"),
              plot.margin = unit(c(1,0.5,0.5,0.5), "cm"),
              panel.spacing = unit(0.75, "lines")) +
        ylab("Elasticity of population growth") +
        xlab("Sex-specific parameter") +
        scale_fill_manual(values = cbPalette) +
        scale_y_continuous(limits = c(-0.5, 0.5), breaks = c(-0.5, -0.25, 0, 0.25, 0.5),
                           labels = c("0.5", "0.25", "0", "0.25", "0.5")) +
        scale_x_discrete(labels = c("Adult_survival" = "Adult surv",
                                    "Juvenile_survival" = "Fledg. surv",
                                    "Chick_survival" = "Chick surv",
                                    "Hatching_sex_ratio" = "Hatching SR",
                                    "Mating_system" = "Mating sys."))
lambda_elasticity
```



Predicitive accuracy of two-sex and one-sex matrix models

this function extracts the deterministic lambda from each bootstrap iteration using a two-sex matrix and a one-sex matrix that has sex-averaged vital rates and constant fecundity

```r
lambda_extract <-
  function(survival_rates, niter = 1000){
    # make an empty datarame to store the results
    output <- data.frame(two_sex_lambda = numeric(niter),
                         one_sex_lambda = numeric(niter))
    # for loop to go through each iteration and calculate the differece between female and male
    # survival rates for each stage.
    for(i in 1:niter){
      # Create a list of demographic rates from the survival analyses above
      VR_two_sex_boot <- list(F_Chk_survl = survival_rates[which(survival_rates$iter == i), 2][5],
                              F_Juv_survl = survival_rates[which(survival_rates$iter == i), 2][3],
                              F_Adt_survl = survival_rates[which(survival_rates$iter == i), 2][1],
                              M_Chk_survl = survival_rates[which(survival_rates$iter == i), 2][6],
                              M_Juv_survl = survival_rates[which(survival_rates$iter == i), 2][4],
                              M_Adt_survl = survival_rates[which(survival_rates$iter == i), 2][2],
                              # Define h (harem size, h < 1 is polyandry) and k (clutch size)
                              h = h,
                              k = 3,
                              # Define primary sex ratio (assumed to be 0.5)
                              HSR = HSR)

      VR_one_sex_boot <- list(Chk_survl = mean(survival_rates[which(survival_rates$iter == i), 2][6],
                                               survival_rates[which(survival_rates$iter == i), 2][5]),
                              Juv_survl = mean(survival_rates[which(survival_rates$iter == i), 2][4],
                                               survival_rates[which(survival_rates$iter == i), 2][3]),
                              Adt_survl = mean(survival_rates[which(survival_rates$iter == i), 2][2],
                                               survival_rates[which(survival_rates$iter == i), 2][1]),
                              # female fecundity
                              RF = RF)

      # Build matrix based on rates specified in the list above
      two_sex_matrix_boot <- plover_matrix(VR_two_sex_boot, two_sex = TRUE)
      one_sex_matrix_boot <- plover_matrix(VR_one_sex_boot, two_sex = FALSE)

      # Determine the ASR at the stable stage distribution
      two_sex_lambda <-
        matrix_ASR(M = two_sex_matrix_boot, h = VR_two_sex_boot$h,
                   HSR = VR_two_sex_boot$HSR, iterations = 75)
      one_sex_lambda <-
        Re(eigen(one_sex_matrix_boot)$values)[1]

      # Extract ASR
      output[i, 1] <- two_sex_lambda$lambda
      output[i, 2] <- one_sex_lambda
    }
    # restructure the output and lable columns
    output <- suppressMessages(reshape2::melt(data = output))
    colnames(output) <- c("parameter", "estimate")
    # return the output
    output
  }
```

Apply the function to the bootstrap output

```
lambda_boot <- lambda_extract(survival_rates = survival_rates_boot, niter = 1000)
```

calculate the 95% confidence intervals and means of the two-sex and one-sex versions of the matrix model

```
CI <- 0.95
lambda_boot_summary <-
  lambda_boot%>%
  dplyr::group_by(parameter)%>%
  dplyr::summarise(mean = mean(estimate),
                   lcl = quantile(estimate, (1 - CI)/2),
                   ucl = quantile(estimate, 1 - (1 - CI)/2))
```

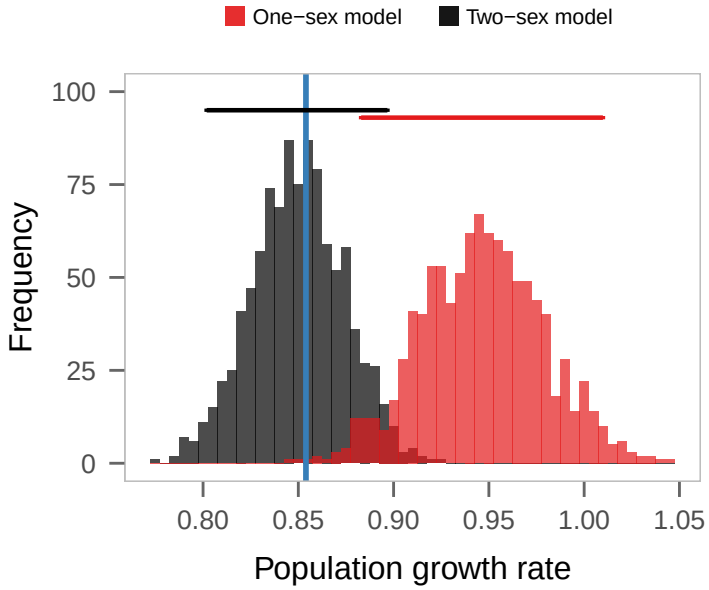
Store the actual annual population size estimate from the field and calculate the mean of the popualtion growth rate observed over the seven-year period

```
Actual <- c(198, 184, 98, 102, 131, 96, 58)

Actual_lambda <- c(Actual[2]/Actual[1],
                     Actual[3]/Actual[2],
                     Actual[4]/Actual[3],
                     Actual[5]/Actual[4],
                     Actual[6]/Actual[5],
                     Actual[7]/Actual[6])

Actual_lambda_mean <- mean(Actual_lambda)
```

Figure 4: plot the predicted lambda distributions of the two-sex and one-sex model simulations between 2006 and 2012, including 95% CI and the observed average population growth.

```
cbPalette <- c(brewer.pal(8, "Set1")[c(1)], brewer.pal(9, "Greys")[c(9)])
lambda_histogram <-
ggplot() +
  geom_histogram(binwidth = 0.005, data = filter(lambda_boot, parameter == "two_sex_lambda"),
                 aes(x = estimate, fill = parameter), alpha = 0.7) +
  geom_histogram(binwidth = 0.005, data = filter(lambda_boot, parameter == "one_sex_lambda"),
                 aes(x = estimate, fill = parameter), alpha = 0.7) +
  geom_vline(xintercept = Actual_lambda_mean, color = brewer.pal(8, "Set1")[c(2)], size=1) +
  geom_errorbarh(data = filter(lambda_boot_summary, parameter == "two_sex_lambda"),
                 aes(y = 95, x = lcl, xmin = lcl, xmax = ucl),
                 color = "black", size = 0.8, linetype = "solid") +
  geom_errorbarh(data = filter(lambda_boot_summary, parameter == "one_sex_lambda"),
                 aes(y = 93, x = lcl, xmin = lcl, xmax = ucl),
                 color = brewer.pal(8, "Set1")[c(1)], size = 0.8, linetype = "solid") +
  theme_bw() +
  theme(legend.text = element_text(size = 8),
        legend.title = element_blank(),
        legend.position = "top",
        legend.key.height=unit(0.6,"line"),
        legend.key.width=unit(0.6,"line"),
        axis.title.x = element_text(size=12, margin = margin(10, 0, 0, 0)),
        axis.text.x  = element_text(size=10, margin = margin(5, 0, 0, 0)),
        axis.title.y = element_text(size=12, margin = margin(0, 5, 0, 0)),
        axis.text.y  = element_text(size=10, angle = 0, hjust = 1,
                                    margin = margin(0, 5, 0, 0)),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
```

```
        axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        axis.ticks.x = element_line(size = 0.5, colour = "grey40"),
        panel.border = element_rect(linetype = "solid", colour = "grey"),
        plot.margin = unit(c(0.5,0.5,0.5,0.5), "cm"),
        panel.spacing = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  ylab("Frequency") +
  xlab("Population growth rate") +
  scale_y_continuous(limits = c(0, 100)) +
  scale_fill_manual(values = cbPalette, labels = c("One-sex model     ", "Two-sex model"))
lambda_histogram
```



R session information

```
sessionInfo()
#> R version 3.4.0 (2017-04-21)
#> Platform: x86_64-apple-darwin15.6.0 (64-bit)
#> Running under: macOS Sierra 10.12.4
#>
#> Matrix products: default
#> BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
#> LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
#>
#> locale:
#> [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
#>
#> attached base packages:
#> [1] grid      stats     graphics  grDevices utils     datasets  methods
#> [8] base
```

```
#>
#> other attached packages:
#>  [1] magrittr_1.5       lme4_1.1-13        Matrix_1.2-10
#>  [4] Rmisc_1.5          plyr_1.8.4         lattice_0.20-35
#>  [7] RColorBrewer_1.1-2 reshape2_1.4.2     gridExtra_2.2.1
#> [10] dplyr_0.5.0        ggplot2_2.2.1      stringr_1.2.0
#> [13] RMark_2.2.2
#>
#> loaded via a namespace (and not attached):
#>  [1] Rcpp_0.12.11     nloptr_1.0.4     compiler_3.4.0   tools_3.4.0
#>  [5] digest_0.6.12    nlme_3.1-131     evaluate_0.10    tibble_1.3.1
#>  [9] gtable_0.2.0     rlang_0.1.1      DBI_0.6-1        yaml_2.1.14
#> [13] parallel_3.4.0   mvtnorm_1.0-6    expm_0.999-2     coda_0.19-1
#> [17] knitr_1.16       rprojroot_1.2    R6_2.2.1         survival_2.41-3
#> [21] rmarkdown_1.5    minqa_1.2.4      MASS_7.3-47      backports_1.1.0
#> [25] scales_0.4.1     htmltools_0.3.6  matrixcalc_1.0-3 splines_3.4.0
#> [29] assertthat_0.2.0 colorspace_1.3-2 labeling_0.3     stringi_1.1.5
#> [33] lazyeval_0.2.0   munsell_0.4.3    msm_1.6.4
```