R-code for "Consequences of stage- and sex-specific demography on the adult sex ratio of a polygamous bird population"

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

August 8, 2016

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` folder 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:

```r
# 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")
library(RMark)
library(stringr)
library(ggplot2)
library(dplyr)
library(gridExtra)
library(grid)
library(reshape2)
library(RColorBrewer)
library(Rmisc)
library(stats)
library(lme4)
```

Loading and wrangling data

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

- **chick_survival_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.
- **fledgling_adult_survival_data.txt** contains the mark-recapture field data of fledglings 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. *age* 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.
- **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.

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

fledgling_adult <-
  read.table("/Users/Luke/Dropbox/Luke/R_projects/Ceuta_ASR_Matrix_Modeling/data/fledgling_adult_surviva
             header = TRUE, colClasses = c("factor","character","factor","factor"))

breeding_data <-
  read.table("/Users/Luke/Dropbox/Luke/R_projects/Ceuta_ASR_Matrix_Modeling/data/breeding_data.txt",
             header = TRUE)
```

Sex-specific fecundity

The objective here was to determine the average per capita annual fecundity for males and females. These sex-specific vital rates were incorporated into the two-sex matrix model. The second objective was to evalutate if the distributions of male and female fecundity were different, which would provide evidence of the polygamous nature of the snowy plover mating system.

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.

```r
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

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

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

```r
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

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

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

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

group data according to Year, Sex, then Ring

```r
reproduction_df <- group_by(reproduction_df, year, Sex, Ring)
```

sum the total chicks produced per bird each year

```r
reproduction_df_sum <-
  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

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

group data according to Sex then Ring

```r
reproduction_df_sum <- group_by(reproduction_df_sum, Sex, Ring)
```

calculate avg total chicks produced per bird each year

```r
reproduction_df_sum_avg <-
  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

```r
fecundity_summary <-
  Rmisc::summarySE(reproduction_df_sum_avg,
                   measurevar = "avg_chicks_p_year", groupvars = c("Sex"))
```

Determine how many individuals were included in the analysis

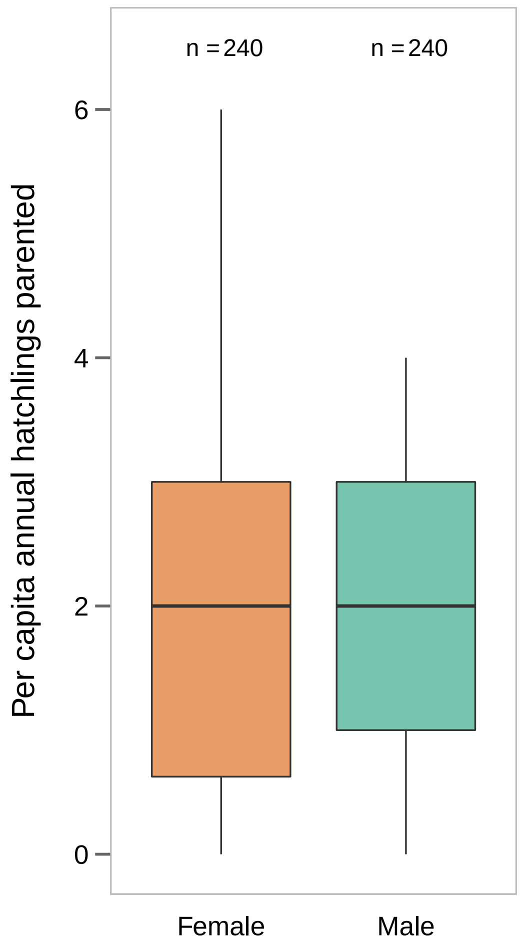
```r
sample_sizes_sex <-
  aggregate(Ring ~ Sex, data = reproduction_df_sum_avg, FUN = function(x){NROW(x)})
```

specify the color pallete to use for the plot

```r
cbPalette <- brewer.pal(8, "Dark2")[c(2,1)]
```

Step three: plot the sex-specific distributions

```r
ggplot() +
  geom_boxplot(aes(y = avg_chicks_p_year, x = Sex, fill = Sex),
               data = reproduction_df_sum_avg, size = .3, alpha = 0.6) +
  geom_text(data = sample_sizes_sex, size = 3,
            aes(y = c(6.5, 6.5), x = c(1.12, 2.12), label = Ring)) +
  annotate("text", x = c(0.92, 1.92), y = c(6.5, 6.5), label = "n = ",
           size = 3) +
  theme_bw() +
  theme(text=element_text(size=16),
        legend.position = "none",
        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.y = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        axis.ticks.x = element_blank(),
        panel.border = element_rect(linetype = "solid", colour = "grey")) +
  scale_fill_manual(values = cbPalette) +
  ylab("Per capita annual hatchlings parented") +
  scale_y_continuous(limits = c(0, 6.5))
```



Step four: statistical testing of sex-specific variance in fecundity

Run F-test to assess sex-specific variation in per capita fecundity

```
reproduction_df_sum_avg <- as.data.frame(reproduction_df_sum_avg)
var.test(reproduction_df_sum_avg[which(reproduction_df_sum_avg$Sex == "Female"),
                                 c("avg_chicks_p_year")],
         reproduction_df_sum_avg[which(reproduction_df_sum_avg$Sex == "Male"),
                                 c("avg_chicks_p_year")],
         alternative = "greater")
#>
#>  F test to compare two variances
#>
#> data:  reproduction_df_sum_avg[which(reproduction_df_sum_avg$Sex ==  and reproduction_df_sum_avg[whi
#> F = 1.4461, num df = 239, denom df = 239, p-value = 0.002255
#> alternative hypothesis: true ratio of variances is greater than 1
#> 95 percent confidence interval:
#>  1.168391      Inf
#> sample estimates:
#> ratio of variances
#>           1.446064
```

Assign the sex-specific values to constants that will be included in the matrix

```
RF <- fecundity_summary[1,3]
RM <- fecundity_summary[2,3]
```

Hatching sex ratio

Before running the matrix model, the hatching sex ratio needs to be calculated.

```
# subset the captures so that only chicks captured on the day of hatch are included
# (the "ch" refers to the capture history of an individual on each day of its life as
# a chick.  Thus, if the first charactoer 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)

# run 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
HSR <- mean(HSR_df_summary$prop_male)

# 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

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"
```

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, mating_function = TRUE)
  {
    if(mating_function)
    {
```

```
    # 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(0, NA, 0, NA,
               (demographic_rates$F_Chk_survl*demographic_rates$F_Fdg_survl),
               demographic_rates$F_Adt_survl,
               0, 0,
               0, NA, 0, NA,
               0, 0,
               (demographic_rates$M_Chk_survl*demographic_rates$M_Fdg_survl),
               demographic_rates$M_Adt_survl),
             nrow = 4, byrow = TRUE,
             dimnames = list(stages, stages))
    result
  }
  else
  {
    # 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(0, demographic_rates$RF * (1 - HSR), 0, demographic_rates$RM * (1 - HSR),
               (demographic_rates$F_Chk_survl*demographic_rates$F_Fdg_survl),
               demographic_rates$F_Adt_survl,
               0, 0,
               0, demographic_rates$RF * HSR, 0, demographic_rates$RM * HSR,
               0, 0,
               (demographic_rates$M_Chk_survl*demographic_rates$M_Fdg_survl),
               demographic_rates$M_Adt_survl),
             nrow = 4, byrow = TRUE,
             dimnames = list(stages, stages))
    result
  }
}
```

freq_dep_SSD_ASR() calculates the ASR of the population based on the two-sex two-stage projection matrix built by the *plover_matrix()* function. It also incorporates the frequency-dependent harmonic mean mating 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) *h* is the harem size for the species. h > 1 is polgynous, h < 1 is polyandrous, h = 1 is monogamous (default) *k* is the clutch size for the species (default is 3) *HSR* is the hatching sex ratio (default is 0.5) *iterations* is the number of iterations to simulate the matrix until SSD is achieved (default is 30)

```
freq_dep_SSD_ASR <-
  function (A, n = rep(10, nrow(A)), h = 1, k = 3,
            iterations = 30, HSR = 0.5, mating_function = TRUE)
  {
    # Number of stages in matrix
    x <- length(n)
    if(mating_function)
    {
      # Number of time steps to simulate
      t <- iterations
      # an empty t by x matrix
      stage <- matrix(numeric(x * t), nrow = x)
      # for loop that goes through each of t time steps
      for (i in 1:t) {
        # stage distribution at time t
        stage[,i] <- 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
        A[1,x/2]        <- (k*M2)/(M2+(F2/h))*HSR
        # Female freq-dep fecundity of Male chicks
        A[(x/4)*3,x/2]  <- (k*M2)/(M2+(F2/h))*HSR
        # Male freq-dep fecundity of Female chicks
        A[1,x]          <- (k*F2)/(M2+(F2/h))*HSR
        # Male freq-dep fecundity of Male chicks
        A[(x/4)*3,x]    <- (k*F2)/(M2+(F2/h))*HSR
        # define the new n (i.e., new stage distribution at time t)
        n <- A %*% n
        # define rownames of stage matrix
        rownames(stage) <- rownames(A)
        # define colnames of stage matrix
        colnames(stage) <- 0:(t - 1)
        # define stable stage as the last stage
        w <- stage[, t]
      }
      # calculate the proportional stable stage distribution
      stable.stage <- w/sum(w)
      # 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])

      # make a list of results
      pop.proj <- list(ASR = ASR,
                       stable.stage = stable.stage,
                       stage.vectors = stage,
                       SSD_M2 = stable.stage[4],
                       SSD_F2 = stable.stage[2])
    }
    else
    {
      ev <- eigen(A)
      lmax <- which.max(Re(ev$values))
      W <- ev$vectors
      w <- abs(Re(W[, lmax]))
      names(w) <- colnames(A)
    # calculate the proportional stable stage distribution
    stable.stage <- w/sum(w)
    # 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])

    # make a list of results
    pop.proj <- list(ASR = ASR,
                       stable.stage = stable.stage,
                       SSD_M2 = stable.stage[4],
                       SSD_F2 = stable.stage[2])
    }
    # print the list as output to the function
    pop.proj
  }
```

This section runs our bootstrap of the mark-recapture datasets. Each iteration will do the following computational steps:

1) Load the following function **bootstrap_data()** to randomly sample with replacement from the *chick* and *fledgling_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 *fledgling_adult* data. Each bootstrapped sample has the same length as the original data.

```
bootstrap_data <- function(fledgling_adult, chick) {
  chick_boot <- chick[sample(1:nrow(chick),
                             size = nrow(chick),
                             replace = TRUE), ]
  present <- fledgling_adult$ring %in% chick_boot$ring
  fledgling_adult_boot1 <- fledgling_adult[present, ]
  spare_fledgling_adult <- fledgling_adult[!present, ]
  fledgling_adult_boot2 <-
    spare_fledgling_adult[sample(1:nrow(spare_fledgling_adult),
                                 size = nrow(fledgling_adult) -
                                   nrow(fledgling_adult_boot1),
                                 replace = TRUE), ]
  fledgling_adult_boot <- rbind(fledgling_adult_boot1, fledgling_adult_boot2)
  out <- list(chick_boot = chick_boot, fledgling_adult_boot = fledgling_adult_boot)
}
```

2) 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) {

  # First steps require some wrangling

  # specify the bootstrapped data samples
  chick <- plover_boot_list[["chick_boot"]]
  fledgling_adult <- plover_boot_list[["fledgling_adult_boot"]]

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

            # Remove capture histories that have no resights (i.e., all zeros in the ch)
            # chick <- chick[!which(str_detect(chick[,"ch"],"1") == TRUE),]

```

```r
# Create processed RMARK data format as Cormack-Jolly_Seber with 2 groups
# (sex and age initally ringed), starting at year 2006, two age groups
# (first-years and adults) in which the first-year stage only lasts for
# one year.
fledgling_adult.proc <- RMark::process.data(fledgling_adult, model = "CJS",
                                            groups = c("sex", "age"),
                                            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
fledgling_adult.ddl <- RMark::make.design.data(fledgling_adult.proc)
chick.ddl <- RMark::make.design.data(chick.proc)

# adds first-year / adult age field to design data in column "Age"
fledgling_adult.ddl <- RMark::add.design.data(data = fledgling_adult.proc,
                                              ddl = fledgling_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.
fledgling_adult.ddl$Phi$marked.as.adult = 0
fledgling_adult.ddl$Phi$marked.as.adult[fledgling_adult.ddl$Phi$initial.age.class == "A"] = 1
fledgling_adult.ddl$p$marked.as.adult = 0
fledgling_adult.ddl$p$marked.as.adult[fledgling_adult.ddl$p$initial.age.class == "A"] = 1

# # check parameter matrices to see if groups were binned correctly
# PIMS(mark(fledgling_adult.proc,fledgling_adult.ddl,
# model.parameters=list(Phi=list(formula=~age+sex)),output=F),"Phi")
#
# Create quadratic time variable so that it can be
# tested for temporal variation in fledgling and adult survival...
time <- c(0:(fledgling_adult.proc$nocc[1] - 1))
quadratic <- time^2
quad_time <- data.frame(time, quadratic)
quad_time$time <- c(2006:2012)
fledgling_adult.ddl$p <-
  merge_design.covariates(fledgling_adult.ddl$Phi,
                          quad_time, bygroup = FALSE, bytime = TRUE)
fledgling_adult.ddl$Phi <-
  merge_design.covariates(fledgling_adult.ddl$Phi,
                          quad_time, bygroup = FALSE, bytime = TRUE)

# ...and chicks
time <- c(0:(chick.proc$nocc[1] - 1))
```

```r
quadratic <- time^2
quad_time <- data.frame(time, quadratic)
chick.ddl$p <-
  merge_design.covariates(chick.ddl$Phi,
                          quad_time, bygroup = FALSE, bytime = TRUE)
chick.ddl$Phi <-
  merge_design.covariates(chick.ddl$Phi,
                          quad_time, bygroup = FALSE, bytime = TRUE)

# Second step is to create the candidate survival models for
# fledglings and adults as a function
fledgling_adult_survival = function()
{
  setwd("/external/spazedawgs/Luke/R_projects/Ceuta_ASR_Matrix_Modeling/output/temp/") # set wd so th
  # sex- and stage-specific survival:
  Phi.agexsex = list(formula = ~ age * sex)

  # Models exploring variation in encounter probability
  # sex-dependent:
  p.sex = list(formula =  ~ sex)
  # age-dependent:
  p.age = list(formula =  ~ age)
  # constant:
  p.dot = list(formula =  ~ 1)
  # linear variation across time:
  p.time = list(formula =  ~ time)
  # factorial variation across time:
  p.Time = list(formula =  ~ Time)
  # quadratic variation across time:
  p.quadratic = list(formula =  ~ quadratic)
  # interaction between sex and linear time:
  p.sexxtime = list(formula =  ~ sex * time)
  # interaction between age and time:
  p.agextime = list(formula =  ~ age * time)
  # interaction between sex and factorial time:
  p.sexxTime = list(formula =  ~ sex * Time)
  # interaction between age and factorial time:
  p.agexTime = list(formula =  ~ age * Time)
  # interaction between age and sex:
  p.agexsex = list(formula =  ~ age * sex)
  # interaction between quadratic time and sex:
  p.quadraticxsex = list(formula =  ~ quadratic * sex)
  # interaction between quadratic time and age:
  p.quadraticxage = list(formula =  ~ quadratic * age)
  # # interaction between quadratic time and sex:
  # p.quadraticxagexsex = list(formula =  ~ quadratic * age * sex)
  # p.Timexagexsex = list(formula =  ~ Time * age * sex)
  # p.timexagexsex = list(formula =  ~ time * age * sex)
  # additive effects of sex and linear time:
  p.sex_time = list(formula =  ~ sex + time)
  # additive effects of age and linear time:
  p.age_time = list(formula =  ~ age + time)
  # additive effects of sex and factorial time:
```

```r
  p.sex_Time = list(formula =  ~ sex + Time)
  # additive effects of age and factorial time:
  p.age_Time = list(formula =  ~ age + Time)
  # additive effects of age and sex:
  p.age_sex = list(formula =  ~ age + sex)
  # additive effects of sex and quadratic time:
  p.quadratic_sex = list(formula =  ~ quadratic + sex)
  # additive effects of age and quadratic time:
  p.quadratic_age = list(formula =  ~ quadratic + age)
  # additive effects of sex, age, and quadratic time:
  p.quadratic_age_sex = list(formula =  ~ quadratic + age + sex)
  # additive effects of sex, age, factorial time:
  p.Time_age_sex = list(formula =  ~ Time + age + sex)
  # additive effects of sex, age, linear time:
  p.time_age_sex = list(formula =  ~ time + age + sex)

  # 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, the number of threads to use
  # (if using a server) and run the models in Program MARK
  model.list = RMark::mark.wrapper(cml, data = fledgling_adult.proc,
                                   ddl = fledgling_adult.ddl,
                                   threads = 20, brief = TRUE, delete = TRUE)

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

# Run the models on the bootstrapped data
fledgling_adult_survival_run <-
  fledgling_adult_survival()

# Extract the AIC model table from the model output
AIC_table_fledgling_adult <-
  fledgling_adult_survival_run$model.table

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

# extract and format survival rates from fledgling and adult model output
fledgling_adult_reals <-
  fledgling_adult_survival_run[[model_fledgling_adult_num]]$results$real

# format the output to tidy up the sex- and age-specific effects
Groups <- data.frame(str_split_fixed(rownames(fledgling_adult_reals), " ", n = 5))
fledgling_adult_reals <- cbind(Groups, fledgling_adult_reals)
fledgling_adult_reals <-
  fledgling_adult_reals[which(fledgling_adult_reals$X1 == "Phi"),]
fledgling_adult_reals$age <-
  unlist(str_extract_all(fledgling_adult_reals$X2,"[AJ]"))
```

```r
fledgling_adult_reals$age <-
  as.factor(ifelse(fledgling_adult_reals$age == "A","Adult","Fledgling"))
fledgling_adult_reals$sex <-
  unlist(str_extract_all(fledgling_adult_reals$X2,"[FM]"))
fledgling_adult_reals$sex <-
  as.factor(ifelse(fledgling_adult_reals$sex == "F","Female","Male"))
fledgling_adult_reals$sex_age <-
  paste(fledgling_adult_reals$sex,fledgling_adult_reals$age,sep = "_")
fledgling_adult_survival_real <-
  fledgling_adult_reals[,c("sex_age", "estimate")]
row.names(fledgling_adult_survival_real) <- NULL

# Do the same for chicks by creating the candidate survival models for
# chicks as a function
chick_survival = function()
{
  setwd("/external/spazedawgs/Luke/R_projects/Ceuta_ASR_Matrix_Modeling/output/temp/") # set wd so th
  # # sex- and linear age-specific survival:
  # Phi.Time.x.sex = list(formula = ~ Sex * Time)
  # sex- and quadratic age-specific survival:
  Phi.quadratic.x.sex = list(formula = ~ sex * quadratic)
  # # sex-specific survival:
  # Phi.Sex = list(formula = ~ Sex)

  # Models exploring variation in encounter probability
  # constant:
  p.dot = list(formula = ~ 1)
  # # linear across time
  # p.Time = list(formula = ~ Time)
  # quadratic across time
  p.quadratic = list(formula = ~ quadratic)
  # annual variation
  p.year = list(formula = ~ year)
  # sex-specific
  p.sex = list(formula = ~ sex)
  # # interaction between year and linear age
  # p.year.x.Time = list(formula = ~ Year * Time)
  # interaction between year and quadratic age
  p.year.x.quadratic = list(formula = ~ year * quadratic)
  # # interaction between sex and linear age
  # p.year.x.Time = list(formula = ~ Sex * Time)
  # interaction between year and quadratic age
  p.sex.x.quadratic = list(formula = ~ sex * quadratic)
  # # additive effects of sex and linear age
  # p.sex.Time = list(formula = ~ Sex + Time)
  # additive effects of sex and linear age
  p.sex.quadratic = list(formula = ~ sex + quadratic)
  # # additive effects of year and linear age
  # p.year.Time = list(formula = ~ Year + Time)
  # additive effects of year and quadratic age
  p.year.quadratic = list(formula = ~ year + quadratic)
  # # additive effects of year, sex, and linear age
  # p.year.Time.Sex = list(formula = ~ Year + Time + Sex)
  # additive effects of year, sex, and quadratic age
  p.year.quadratic.Sex = list(formula = ~ year + quadratic + sex)

  # 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, the number of threads to use
  # (if using a server) and run the models in Program MARK
  model.list = RMark::mark.wrapper(cml, data = chick.proc, ddl = chick.ddl,
                                   threads  =  20, brief = TRUE, 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.
# this can either be done by DCS^25 if there is no age effect:
if(nrow(chick_reals) == 2)
{
  plover_Survival_to_Fledge_F <-
    chick_reals[which(chick_reals$sex == "Female"),
                c("estimate")]^25
  plover_Survival_to_Fledge_M <-
    chick_reals[which(chick_reals$sex == "Male"),
                c("estimate")]^25
}
# or by calculating the product of all DCS estimates:
if(nrow(chick_reals) != 2){
  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")
  age <- c("Chick", "Chick")
  sex_age <- paste(sex, age, sep = "_")
  chick_survival_real <- data.frame(sex_age, estimate)

  # Bind the fledgling and adult dataframe with the chicks
  survival_rates <- rbind(fledgling_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_Fdg_survl = survival_rates[3,2],
                            F_Adt_survl = survival_rates[1,2],
                            M_Chk_survl = survival_rates[6,2],
                            M_Fdg_survl = survival_rates[4,2],
                            M_Adt_survl = survival_rates[2,2],
                            # # Define h (harem size, h < 1 is polyandry) and k (clutch size)
                            # h = 1,
                            # k = 3,
                            # Define hatching sex ratio
                            HSR = HSR,
                            # Define the fecundity of males (RM) and females (RF)
                            RF = RF,
                            RM = RM)

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

  # Determine the ASR at the stable stage distribution
  #ASR_SSD <- freq_dep_SSD_ASR(A = demographic_matrix, h = 1, k = 3, HSR = HSR)
  ASR_SSD <- freq_dep_SSD_ASR(A = demographic_matrix, mating_function = FALSE)

  # 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_fledgling_adult,
         survival_rates,
         ASR_estimate)
}
```

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

```
run_bootstrap_survival_ASR <- function(num_boot, fledgling_adult, chick)
  {
  bootstrap_data_list <- bootstrap_data(fledgling_adult, chick)
  result <- bootstrap_survival_ASR(bootstrap_data_list, num_boot)
}
```

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

```
niter <- 1000
# to run the bootstrap survival analysis, uncomment to following line (could take multiple days to run)
#survival_ASR_bootstrap_result <- sapply(1:niter, run_bootstrap_survival_ASR, fledgling_adult, chick)

# here is the output of the bootstrap
survival_rates_boot <-
  read.table("/Users/Luke/Dropbox/Luke/R_projects/Ceuta_ASR_Matrix_Modeling/data/bootstrap_surv_results
             header = TRUE, colClasses = c("factor", "numeric","factor"))

ASR_boot <-
  read.table("/Users/Luke/Dropbox/Luke/R_projects/Ceuta_ASR_Matrix_Modeling/data/bootstrap_ASR_results.
             header = TRUE, colClasses = c("numeric", "factor"))
```

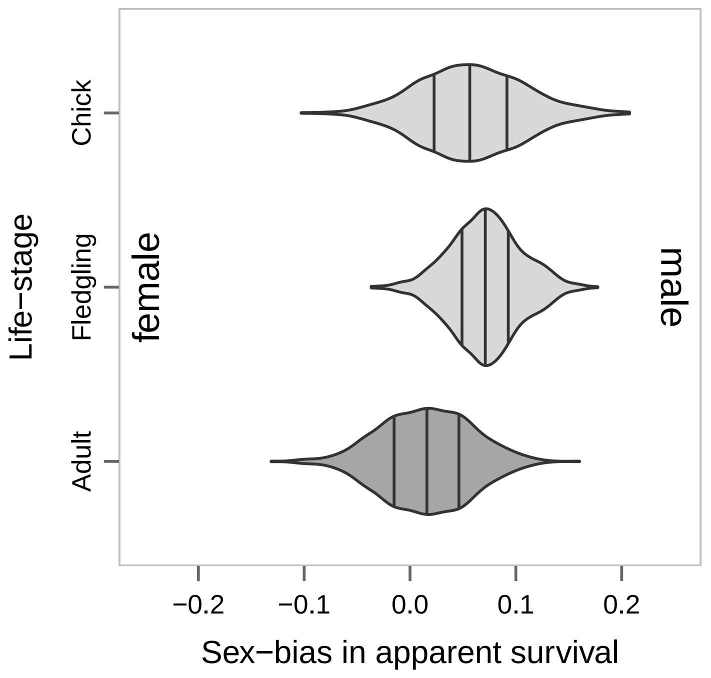
5) Extract the sex- and stage-specific survival rates from the output

```
## extract AIC_table_chick
# 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)

## extract AIC_table_fledgling_adult
# AIC_table_fledgling_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_fledgling_adult_boot)/niter
# AIC_table_fledgling_adult_boot$iter <- rep(1:niter, each = num_mods)

## extract Survival_rates
# 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)

## extract ASR
# 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)
```

Visualizations of bootstrap results

Sex-biases in survial across chicks, fledglings, 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.

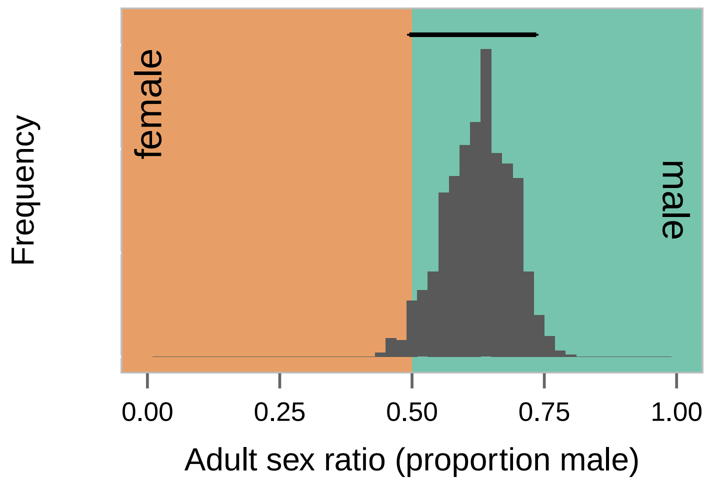
```r
sex_diff_survival <- function(survival_rates_boot) {
  # make an empty datarame to store the results
  sex_diff_surv_output <- data.frame(Adult = numeric(niter),
                                     Fledgling = numeric(niter),
                                     Chick = numeric(niter))
  # for 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]
    Fledgling <-
      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] <- Fledgling
    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)
#> No id variables; using all as measure variables

# 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
# fledglings) 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", "Fledgling", "Chick"))

# plot the sex-biases in survival across the three stages
ggplot2::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)) +
                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(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 = 90, hjust = 0.5,
                                                  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.margin = 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")
```



Adult sex ratio distribution

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.

```r
# specify the confidence interval bounds
CI <- 0.95

# Calculate the confidence interval, mean, and median of the ASR bootstraps
ASR_boot_95CI <- stats::quantile(ASR_boot$ASR, c((1 - CI)/2, 1 - (1 - CI)/2), na.rm = TRUE)
ASR_boot_mean <- mean(ASR_boot$ASR)
ASR_boot_median <- median(ASR_boot$ASR)

# 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")

# plot the ASR histogram
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(95, 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)) +
  geom_errorbarh(data = ASR_boot_summary, aes(y = 155, x = lcl, xmin = lcl, xmax = ucl),
                 color = "black", size = 0.8, linetype = "solid") +
  theme_bw() +
  theme(legend.position="none",
        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, 15, 0, 0)),
        axis.text.y  = element_text(size=10, angle = 90, hjust = 0.5,
                                    margin = margin(0, 5, 0, 0), color = "white"),
        axis.ticks.y = element_line(size = 0.5, colour = "white"),
        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(1,0.5,0.5,0.5), "cm"),
        strip.background = element_blank(),
        strip.text = element_blank(),
        panel.margin = 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, 160))
```

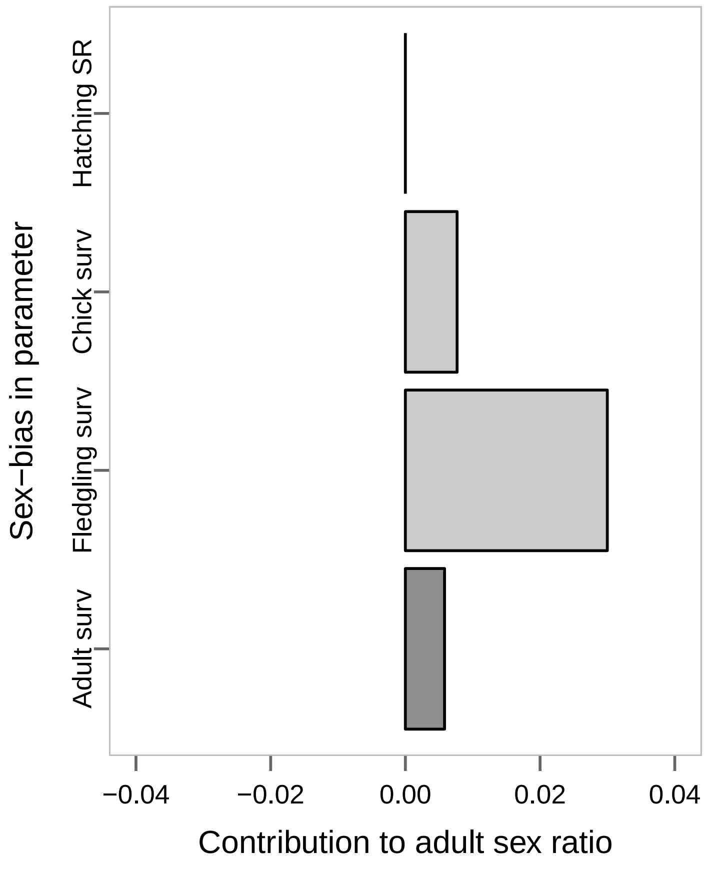


Life table response experiment

```
ASR_analysis <-
  function (A, zero = TRUE)
    {
  ev <- eigen(A) # makes list of the eigen values and eigen vectors of A
  lmax <- which.max(Re(ev$values)) # index of dominant eigen value
  W <- ev$vectors # Eigen vectors
  w <- abs(Re(W[, lmax])) # dominant eigen vector
  stable.stage = w / sum(w) # stable stage distribution
  ASR <- stable.stage[4] / (stable.stage[2] + stable.stage[4]) # SSD ASR
  V <- try(Conj(solve(W)), silent = TRUE) # check if possible to proceed
  if (class(V) == "try-error") {
    ASR.analysis <- list(ASR = ASR, stable.stage = stable.stage,
                         sensitivities = A * NA, elasticities = A * NA)
  }
  else {
    v <- abs(Re(V[lmax, ])) # solve matrix
    s <- v %o% w # outer product of v and w
    if (zero) {
      s[A == 0] <- 0
    }
    e <- s * A/ASR # calculate elasticities
    x <- dimnames(A) # get vital rate names
    dimnames(s) <- x # assign vital rate names to s
    names(w) <- x[[1]]
    names(v) <- x[[1]]
    ASR.analysis <- list(ASR = ASR, stable.stage = stable.stage,
                         sensitivities = s, elasticities = e)
  }
  ASR.analysis
}

lower_level_sens_analysis <-
  function(freq_dep_ASR, VR_list, mating_function = TRUE)
  {
    if(mating_function)
    {
      # make a list of all parameters
      vr <- list(
        F_Chk_survl = VR_list$F_Chk_survl,
        F_Fdg_survl = VR_list$F_Fdg_survl,
        F_Adt_survl = VR_list$F_Adt_survl,
        M_Chk_survl = VR_list$M_Chk_survl,
        M_Fdg_survl = VR_list$M_Fdg_survl,
        M_Adt_survl = VR_list$M_Adt_survl,
        h = VR_list$h,
        k = VR_list$k,
        HSR = VR_list$HSR,
        M2 = unname(freq_dep_ASR$SSD_M2),
        F2 = unname(freq_dep_ASR$SSD_F2))
      # make a matrix of the elements
      el <- expression(0, ((k * M2) / (M2 + (F2 / h))) * (1 - HSR),
                       0, ((k * F2) / (M2 + (F2 / h))) * (1 - HSR),
                       (F_Chk_survl * F_Fdg_survl), F_Adt_survl, 0, 0,
                       0, ((k * M2) / (M2 + (F2 / h))) * HSR, 0,
                       ((k * F2) / (M2 + (F2 / h))) * HSR,
                       0, 0, (M_Chk_survl * M_Fdg_survl), M_Adt_survl)
      # calculate the effect of proportional changes in vital rates
      n <- length(vr)
```

```
vr_nums <- seq(0, 2, 0.01) # proportional changes in increments of 0.01 between 0 and 2
# First calculate sensitivites
vrsen <- matrix(numeric(n * length(vr_nums)),
                ncol = n, dimnames = list(vr_nums, names(vr)))
for (h in 1:n)
{
  vr2 <- vr
  for (i in 1:length(vr_nums))
  {
    vr2[[h]] <- vr_nums[i]
    A <-
      matrix(sapply(el, eval, vr2, NULL), nrow = sqrt(length(el)), byrow=TRUE)
    vrsen[i, h] <-
      eigen(A)$vectors[4, 1] / (eigen(A)$vectors[2, 1] + eigen(A)$vectors[4, 1])
  }
}
# Next calculate rescaled elasticities
vrelas <- matrix(numeric(n * length(vr_nums)),
                 ncol = n, dimnames = list(vr_nums, names(vr)))
for (h in 1:n)
{
  for (i in 1:length(vr_nums))
  {
    vr2 <- vr
    vr2[[h]] <- vr_nums[i] * vr2[[h]]
    A <- matrix(sapply(el, eval, vr2 , NULL), nrow = sqrt(length(el)), byrow = TRUE)
    vrelas[i, h] <-
      (eigen(A)$vectors[4, 1] /
         (eigen(A)$vectors[2, 1] + eigen(A)$vectors[4, 1])) /
      unname(freq_dep_ASR$ASR)
  }
}
# tidy up and label results
colnames(vrsen) <- c("Female chick survival",
                     "Female fledgling survival",
                     "Female adult survival",
                     "Male chick survival",
                     "Male fledgling survival",
                     "Male adult survival",
                     "Mating system index (h)",
                     "Clutch size",
                     "Hatching sex ratio",
                     "Breeding males",
                     "Breeding females")
colnames(vrelas) <- c("Female chick survival",
                      "Female fledgling survival",
                      "Female adult survival",
                      "Male chick survival",
                      "Male fledgling survival",
                      "Male adult survival",
                      "Mating system (h)",
                      "Clutch size",
                      "Hatching sex ratio",
                             "Breeding males",
                             "Breeding females")
    }
    else
    {
      # make a list of all parameters
      vr <- list(
        F_Chk_survl = VR_list$F_Chk_survl,
        F_Fdg_survl = VR_list$F_Fdg_survl,
        F_Adt_survl = VR_list$F_Adt_survl,
        M_Chk_survl = VR_list$M_Chk_survl,
        M_Fdg_survl = VR_list$M_Fdg_survl,
        M_Adt_survl = VR_list$M_Adt_survl,
        RF = VR_list$RF,
        RM = VR_list$RM,
        HSR = VR_list$HSR)
      # make a matrix of the elements
      el <- expression(0, RF * (1 - HSR),
                       0, RM * (1 - HSR),
                       (F_Chk_survl * F_Fdg_survl), F_Adt_survl, 0, 0,
                       0, RF * HSR,
                       0, RM * HSR,
                       0, 0, (M_Chk_survl * M_Fdg_survl), M_Adt_survl)
      # calculate the effect of proportional changes in vital rates
      n <- length(vr)
      vr_nums <- seq(0, 2, 0.01) # proportional changes in increments of 0.01 between 0 and 2
      # First calculate sensitivites
      vrsen <- matrix(numeric(n * length(vr_nums)),
                      ncol = n, dimnames = list(vr_nums, names(vr)))
      for (h in 1:n)
      {
        vr2 <- vr
        for (i in 1:length(vr_nums))
        {
          vr2[[h]] <- vr_nums[i]
          A <-
            matrix(sapply(el, eval, vr2, NULL), nrow = sqrt(length(el)), byrow=TRUE)
          vrsen[i, h] <-
            eigen(A)$vectors[4, 1] / (eigen(A)$vectors[2, 1] + eigen(A)$vectors[4, 1])
        }
      }
      # Next calculate rescaled elasticities
      vrelas <- matrix(numeric(n * length(vr_nums)),
                       ncol = n, dimnames = list(vr_nums, names(vr)))
      for (h in 1:n)
      {
        for (i in 1:length(vr_nums))
        {
          vr2 <- vr
          vr2[[h]] <- vr_nums[i] * vr2[[h]]
          A <- matrix(sapply(el, eval, vr2 , NULL), nrow = sqrt(length(el)), byrow = TRUE)
          vrelas[i, h] <-
            (eigen(A)$vectors[4, 1] /
               (eigen(A)$vectors[2, 1] + eigen(A)$vectors[4, 1])) /
            unname(freq_dep_ASR$ASR)
        }
      }
      # tidy up and label results
      colnames(vrsen) <- c("Female chick survival",
                           "Female fledgling survival",
                           "Female adult survival",
                           "Male chick survival",
                           "Male fledgling survival",
                           "Male adult survival",
                           "Female Fecundity",
                           "Male Fecundity",
                           "Hatching sex ratio")
      colnames(vrelas) <- c("Female chick survival",
                            "Female fledgling survival",
                            "Female adult survival",
                            "Male chick survival",
                            "Male fledgling survival",
                            "Male adult survival",
                            "Female Fecundity",
                            "Male Fecundity",
                            "Hatching sex ratio")
    }
    Sensitivities <- melt(vrsen)
    colnames(Sensitivities) <- c("Perturbation", "Vitalrate", "Sensitivity")
    Elasticities <- melt(vrelas)
    colnames(Elasticities) <- c("Perturbation", "Vitalrate", "Elasticity")
    results <- list(Sensitivities = Sensitivities,
                    Elasticities = Elasticities,
                    Element_expression = el)
    results
  }

# LTRE analysis of ASR
vitalsens_ASR <-
  function (elements, VR_list, freq_dep_ASR, mating_function = TRUE)
  {
    if(mating_function)
    {
      # list of parameters in the treatment matrix
      # this contains the observed paramters based on previous analyses
      treatment_matrix <- list(
        F_Chk_survl = VR_list$F_Chk_survl,
        F_Fdg_survl = VR_list$F_Fdg_survl,
        F_Adt_survl = VR_list$F_Adt_survl,
        M_Chk_survl = VR_list$M_Chk_survl,
        M_Fdg_survl = VR_list$M_Fdg_survl,
        M_Adt_survl = VR_list$M_Adt_survl,
        h = VR_list$h,
        k = VR_list$k,
        HSR = VR_list$HSR,
        M2 = unname(freq_dep_ASR$SSD_M2),
        F2 = unname(freq_dep_ASR$SSD_F2))
      # list of parameters in the control matrix
      # this contains parameters with no sex-differences
      control_matrix <- list(
        F_Chk_survl = VR_list$M_Chk_survl,
        F_Fdg_survl = VR_list$M_Fdg_survl,
        F_Adt_survl = VR_list$M_Adt_survl,
        M_Chk_survl = VR_list$M_Chk_survl,
        M_Fdg_survl = VR_list$M_Fdg_survl,
        M_Adt_survl = VR_list$M_Adt_survl,
        h = VR_list$h,
        k = VR_list$k,
        HSR = 0.5,
        M2 = unname(freq_dep_ASR$SSD_M2),
        F2 = unname(freq_dep_ASR$SSD_F2))
      # check if everything is correctly structured before proceeding
      if (is.vector(treatment_matrix)) {
        treatment_matrix <- as.list(treatment_matrix)
      }
      if (!is.list(treatment_matrix)) {
        stop("Vital rates should be a vector or list")
      }
      if (class(elements) != "expression") {
        stop("Matrix elements should be an expression")
      }
      if (is.vector(control_matrix)) {
        control_matrix <- as.list(control_matrix)
      }
      if (!is.list(control_matrix)) {
        stop("Vital rates should be a vector or list")
      }
      # find the number of stage and sex specific parameters
      n <- sqrt(length(elements))
      if (n%%1 != 0) {
        stop(paste("Length of element expression is", length(elements),
                   "- Expecting power of 2 like 4, 9, 16 to form a square matrix"))
      }
      # add the mating function parameters to the matrices
      vrs <- try(sapply(elements, eval, treatment_matrix, NULL), silent = TRUE)
      vrs_LTRE <- try(sapply(elements, eval, control_matrix, NULL), silent = TRUE)
      if (class(vrs) == "try-error") {
        vrs <- sub("Error in eval\\(expr, envir, enclos\\) :",
                   "", vrs[1])
        stop(paste("Cannot evaluate element expression using given vital rates:",
                   vrs))
      }
      if (class(vrs_LTRE) == "try-error") {
        vrs_LTRE <- sub("Error in eval\\(expr, envir, enclos\\) :",
                        "", vrs_LTRE[1])
        stop(paste("Cannot evaluate element expression using given vital rates:",
                   vrs_LTRE))
      }
      # make an empty dataframe where all the perturbation stats will go
      res <- data.frame(estimate = unlist(treatment_matrix), sensitivity = 0,
                        elasticity = 0, LTRE = 0)
      # build the treatment matrix
      A <- matrix(vrs, nrow = n, byrow = TRUE)
      # build the control matrix
      A_LTRE <- matrix(vrs_LTRE, nrow = n, byrow = TRUE)
      # transform the matrix to M-prime (see formula in manuscript)
      Ac <- (A + A_LTRE) / 2
      # run sensitivity analyses on both matrices
      SAc <- ASR_analysis(Ac)
      ASR <- ASR_analysis(A)
      # calculate derivatives of lower-level matrix elements
      deriv.funcs <- sapply(elements, deriv, namevec = names(treatment_matrix),
                            function.arg = TRUE)
      devs <- lapply(deriv.funcs, function(x) do.call(x, treatment_matrix))
      # run for loop to go through each parameter and estimate elasticity,
      # sensitivity, and LTRE
      for (i in 1:length(treatment_matrix)) {
        derivs <- matrix(as.numeric(lapply(devs, function(x) attr(x, "gradient")[i])),
                         nrow = n, byrow = TRUE)
        res[i, 2] <- sum(derivs * ASR$sensitivities)
        res[i, 3] <- treatment_matrix[[i]] / ASR$ASR * sum(derivs * ASR$sensitivities)
        # only do LTRE calculations on survival parameters RELATIVE to one sex
        # i.e., don't calculate LTRE on mating system components
        res[i, 4] <- ifelse(i > 3 & i < 6, NA,
                            ifelse(i < 4, (treatment_matrix[[i + 3]] - treatment_matrix[[i]]) *
                                     sum(derivs * SAc$sensitivities),
                                   ifelse(i == 9, (control_matrix[[i]] - treatment_matrix[[i]]) *
                                            sum(derivs * SAc$sensitivities),
                                          NA)))
      }
      # consolidate results
      y <- res
      y$Vital_rate <- as.factor(rownames(y))
      colnames(y) <- c("Estimate", "Sensitivity", "Elasticity", "LTRE", "Vital_rate")
      y_melt <- suppressMessages(melt(y[,c(2:5)]))
      y_melt$parameter <-
        as.factor(ifelse(str_detect(y_melt$Vital_rate,"Chk"), "Chick survival",
                    ifelse(str_detect(y_melt$Vital_rate,"Fdg"), "Fledgling survival",
                      ifelse(str_detect(y_melt$Vital_rate,"Adt"), "Adult survival",
                        ifelse(str_detect(y_melt$Vital_rate,"HSR"), "Hatching sex ratio",
                          ifelse(str_detect(y_melt$Vital_rate,"h"), "Mating System",
                            ifelse(str_detect(y_melt$Vital_rate,"k"), "Clutch size",
                                                            "No. breeding adults")))))))
      y_melt$parameter <- factor(y_melt$parameter, levels = c("Adult survival",
                                                              "Fledgling survival",
                                                              "Chick survival",
                                                              " ",
                                                              "No. breeding adults",
                                                              "Mating System",
                                                              "Clutch size",
                                                              "Hatching sex ratio"))
    }
```

```
  else
  {
    # list of parameters in the treatment matrix
    # this contains the observed paramters based on previous analyses
    treatment_matrix <- list(
      F_Chk_survl = VR_list$F_Chk_survl,
      F_Fdg_survl = VR_list$F_Fdg_survl,
      F_Adt_survl = VR_list$F_Adt_survl,
      M_Chk_survl = VR_list$M_Chk_survl,
      M_Fdg_survl = VR_list$M_Fdg_survl,
      M_Adt_survl = VR_list$M_Adt_survl,
      RF = VR_list$RF,
      RM = VR_list$RM,
      HSR = VR_list$HSR)
    # list of parameters in the control matrix
    # this contains parameters with no sex-differences
    control_matrix <- list(
      F_Chk_survl = VR_list$M_Chk_survl,
      F_Fdg_survl = VR_list$M_Fdg_survl,
      F_Adt_survl = VR_list$M_Adt_survl,
      M_Chk_survl = VR_list$M_Chk_survl,
      M_Fdg_survl = VR_list$M_Fdg_survl,
      M_Adt_survl = VR_list$M_Adt_survl,
      RF = VR_list$RF,
      RM = VR_list$RM,
      HSR = VR_list$HSR)
    # check if everything is correctly structured before proceeding
    if (is.vector(treatment_matrix)) {
      treatment_matrix <- as.list(treatment_matrix)
    }
    if (!is.list(treatment_matrix)) {
      stop("Vital rates should be a vector or list")
    }
    if (class(elements) != "expression") {
      stop("Matrix elements should be an expression")
    }
    if (is.vector(control_matrix)) {
      control_matrix <- as.list(control_matrix)
    }
    if (!is.list(control_matrix)) {
      stop("Vital rates should be a vector or list")
    }
    # find the number of stage and sex specific parameters
    n <- sqrt(length(elements))
    if (n%%1 != 0) {
      stop(paste("Length of element expression is", length(elements),
                 "- Expecting power of 2 like 4, 9, 16 to form a square matrix"))
    }
    # add the mating function parameters to the matrices
    vrs <- try(sapply(elements, eval, treatment_matrix, NULL), silent = TRUE)
    vrs_LTRE <- try(sapply(elements, eval, control_matrix, NULL), silent = TRUE)
    if (class(vrs) == "try-error") {
      vrs <- sub("Error in eval\\(expr, envir, enclos\\) :",
              "", vrs[1])
      stop(paste("Cannot evaluate element expression using given vital rates:",
                 vrs))
    }
    if (class(vrs_LTRE) == "try-error") {
      vrs_LTRE <- sub("Error in eval\\(expr, envir, enclos\\) :",
                      "", vrs_LTRE[1])
      stop(paste("Cannot evaluate element expression using given vital rates:",
                 vrs_LTRE))
    }
    # make an empty dataframe where all the perturbation stats will go
    res <- data.frame(estimate = unlist(treatment_matrix), sensitivity = 0,
                      elasticity = 0, LTRE = 0)
    # build the treatment matrix
    A <- matrix(vrs, nrow = n, byrow = TRUE)
    # build the control matrix
    A_LTRE <- matrix(vrs_LTRE, nrow = n, byrow = TRUE)
    # transform the matrix to M-prime (see formula in manuscript)
    Ac <- (A + A_LTRE) / 2
    # run sensitivity analyses on both matrices
    SAc <- ASR_analysis(Ac)
    ASR <- ASR_analysis(A)
    # calculate derivatives of lower-level matrix elements
    deriv.funcs <- sapply(elements, deriv, namevec = names(treatment_matrix),
                          function.arg = TRUE)
    devs <- lapply(deriv.funcs, function(x) do.call(x, treatment_matrix))
    # run for loop to go through each parameter and estimate elasticity,
    # sensitivity, and LTRE
    for (i in 1:length(treatment_matrix)) {
      derivs <- matrix(as.numeric(lapply(devs, function(x) attr(x, "gradient")[i])),
                       nrow = n, byrow = TRUE)
      res[i, 2] <- sum(derivs * ASR$sensitivities)
      res[i, 3] <- treatment_matrix[[i]] / ASR$ASR * sum(derivs * ASR$sensitivities)
      # only do LTRE calculations on survival parameters RELATIVE to one sex
      # i.e., don't calculate LTRE on mating system components
      res[i, 4] <- ifelse(i > 3 & i < 6, NA,
                          ifelse(i < 4, (treatment_matrix[[i + 3]] - treatment_matrix[[i]]) *
                                   sum(derivs * SAc$sensitivities),
                                 ifelse(i == 9, (control_matrix[[i]] - treatment_matrix[[i]]) *
                                          sum(derivs * SAc$sensitivities),
                                        NA)))
    }
    # consolidate results
    y <- res
    y$Vital_rate <- as.factor(rownames(y))
    colnames(y) <- c("Estimate", "Sensitivity", "Elasticity", "LTRE", "Vital_rate")
    y_melt <- suppressMessages(melt(y[,c(2:5)]))
    y_melt$parameter <-
      as.factor(ifelse(str_detect(y_melt$Vital_rate,"Chk"), "Chick survival",
                ifelse(str_detect(y_melt$Vital_rate,"Fdg"), "Fledgling survival",
                  ifelse(str_detect(y_melt$Vital_rate,"Adt"), "Adult survival",
                    ifelse(str_detect(y_melt$Vital_rate,"HSR"), "Hatching sex ratio",
                      ifelse(str_detect(y_melt$Vital_rate,"RF"), "Female Fecundity",
                                "Male Fecundity"))))))
      y_melt$parameter <- factor(y_melt$parameter, levels = c("Adult survival",
                                                             "Fledgling survival",
                                                             "Chick survival",
                                                             "Female Fecundity",
                                                             "Male Fecundity",
                                                             "Hatching sex ratio"))
    }
    y_melt$Sex <- as.factor(ifelse(str_detect(y_melt$Vital_rate,"F_") &
                                    y_melt$variable != "LTRE", "Female",
                                  ifelse(str_detect(y_melt$Vital_rate,"M_") &
                                           y_melt$variable != "LTRE","Male", "Other")))
    y_melt$Sex <-
      factor(y_melt$Sex,
             levels = c("Female","Male", "Other"))
    y_melt$value_trans <- ifelse(y_melt$Sex == "Female",
                                abs(y_melt$value)*-1, y_melt$value)
    y_melt <- y_melt[,-c(1)]
    results <- list(Sensitivity = subset(y_melt, (variable == "Sensitivity")),
                    Elasticity = subset(y_melt, (variable == "Elasticity")),
                    LTRE = subset(y_melt, (variable == "LTRE" & !is.na(value))))
    results$LTRE$parameter <-
      factor(results$LTRE$parameter,
             levels = c("Adult survival",
                        "Fledgling survival",
                        "Chick survival",
                        "Hatching sex ratio"))
    row.names(results$Sensitivity) <- NULL
    row.names(results$Elasticity) <- NULL
    row.names(results$LTRE) <- NULL
    results$Sensitivity$value_trans <-
      as.numeric(results$Sensitivity$value_trans)
    results$Elasticity$value_trans <-
      as.numeric(results$Elasticity$value_trans)
    results$LTRE$value_trans <-
      as.numeric(results$LTRE$value_trans)
    results
  }

# 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 <-
  survival_rates_boot %>%
  dplyr::group_by(sex_age) %>%
  dplyr::summarise(Avg = mean(estimate))

survival_rates_boot_summary <- as.data.frame(survival_rates_boot_summary)

# Define Ceuta vital rates estimated from mark-recapture analysis:
deterministic_list <- list(F_Chk_survl = survival_rates_boot_summary[2,2],
                           F_Fdg_survl = survival_rates_boot_summary[3,2],
                           F_Adt_survl = survival_rates_boot_summary[1,2],
                           M_Chk_survl = survival_rates_boot_summary[5,2],
                           M_Fdg_survl = survival_rates_boot_summary[6,2],
                           M_Adt_survl = survival_rates_boot_summary[4,2],
# # Define h (harem size, h = 1 is monogamy) and k (clutch size)
#                            h = 1,
#                            k = 3,
# Define primary sex ratio (assumed to be 0.5)
                           HSR = HSR,
                           # Define the fecundity of males (RM) and females (RF)
                           RF = RF,
                           RM = RM)

# Ceuta matrix:
deterministic_matrix <- plover_matrix(deterministic_list, mating_function = FALSE)

# Determine the ASR at the stable stage distribution
deterministic_ASR <-
  freq_dep_SSD_ASR(A = deterministic_matrix, HSR = HSR, mating_function = FALSE)
deterministic_ASR$ASR
#>     M_Adt
#> 0.6263899

# Lower-level vital rate sensitivity analysis
deterministic_LLSA <-
  lower_level_sens_analysis(freq_dep_ASR = deterministic_ASR,
                            VR_list = deterministic_list,
                            mating_function = FALSE)

# Calculate vital rate sensitivities and elasticities
deterministic_LTRE <-
  vitalsens_ASR(elements = deterministic_LLSA$Element_expression,
                VR_list = deterministic_list, freq_dep_ASR = deterministic_ASR,
                mating_function = FALSE)

# Custom color palette for the plotting of Fledgling and Adult stats
cbPalette <- c("#737373", "#BDBDBD", "#BDBDBD", "#BDBDBD")

# plot the comparative LTRE results
ggplot2::ggplot() +
  theme_bw() +
  coord_flip() +
  geom_bar(data = deterministic_LTRE$LTRE,
           aes(x = parameter, y = value, fill = parameter),
           color = "black", stat = "identity", alpha = 0.8) +
  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 = 90, hjust = 0.5,
                                    margin = margin(0, 1, 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.margin = 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(-0.04, 0.04)) +
  scale_x_discrete(labels = c("Adult survival" = "Adult surv",
                              "Fledgling survival" = "Fledgling surv",
                              "Chick survival" = "Chick surv",
                              "Hatching sex ratio" = "Hatching SR"))
```



```
# Determine how much larger the contribution of each vital rates is compared to
# fledgling survival

# fledgling vs chick:
```

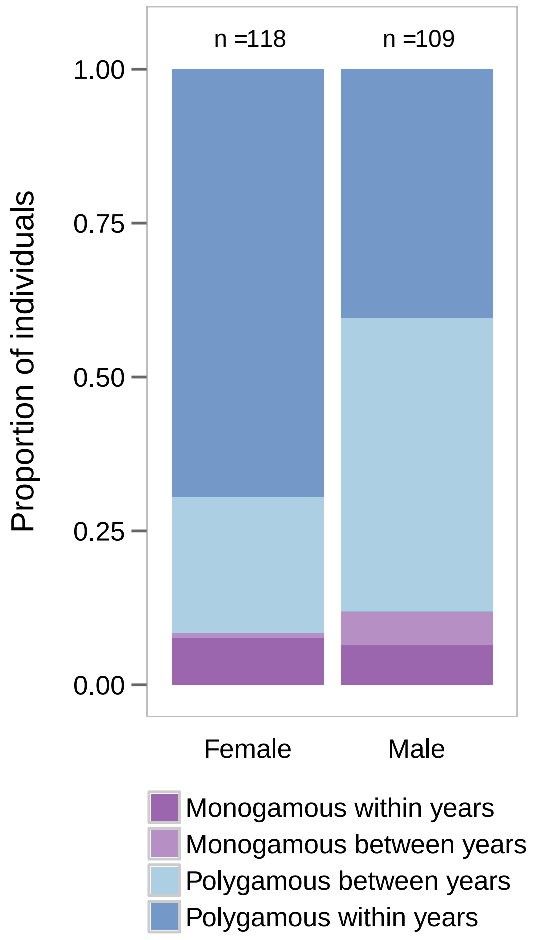
```r
deterministic_LTRE$LTRE[2,5]/deterministic_LTRE$LTRE[1,5]
#> [1] 3.903691

# fledgling vs adult:
deterministic_LTRE$LTRE[2,5]/deterministic_LTRE$LTRE[3,5]
#> [1] 5.155423
```

Quantifying the mating system

```r
# 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)),]

# 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 <- dcast(mating_df, female  ~ year)
#> Using pair as value column: use value.var to override.
#> Aggregation function missing: defaulting to length
males <- dcast(mating_df, male  ~ year)
#> Using pair as value column: use value.var to override.
#> Aggregation function missing: defaulting to length

# determine how many different mates each individual had over their lifetime in the popualtion
number_males_p_female <- aggregate(male ~ female, mating_df, function(x) length(unique(x)))
number_females_p_male <- aggregate(female ~ male, mating_df, function(x) length(unique(x)))

# Join these two dataframes together and define as numeric
females <- inner_join(females, number_males_p_female)
#> Joining by: "female"
females[,c(2:8)] <-
  lapply(females[,c(2:8)], as.numeric)
males <- inner_join(males, number_females_p_male)
#> Joining by: "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 <- filter(females, male  != 1 | years != 1 | attempts != 1)
males_no_1 <- filter(males, female  != 1 | years != 1 | attempts != 1)

# tidy up dataframes then bind them together
females_no_1$sex <- "Female"
```

```r
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)

# Determine if an individual was either:
# a) monogamous between years (i.e. only 1 mate in lifetime, with the number of attempts
# equaling the number years mating)
# b) monogamous within years (i.e. only 1 mate in lifetime, with the number of attempts
# greater the number years mating)
# c) polygamous between years (i.e. more than one mate in lifetime, with the number of
# attempts equaling the number years mating)
# d) polygamous within years (i.e. more than one mate in lifetime, with the number of
# attempts greater the number years mating)
mating$status <- ifelse(mating$mate == 1 & mating$years == mating$attempts,
                        "Monogamous between years",
                    ifelse(mating$mate == 1 & mating$years < mating$attempts,
                               "Monogamous within years",
                            ifelse(mating$mate > 1 & mating$years == mating$attempts,
                                      "Polygamous between years",
                                  ifelse(mating$mate > 1 & mating$years < mating$attempts,
                                             "Polygamous within years", "XXX"))))

# Calculate the number of mates per year
mating$no_mates_per_year <- mating$mate/mating$years

# Run chi-squared test of the sex-differences in polygamy rates
chisq.test(table(mating$sex, mating$status)[,c(3,4)])
#>
#>  Pearson's Chi-squared test with Yates' continuity correction
#>
#> data:  table(mating$sex, mating$status)[, c(3, 4)]
#> X-squared = 18.235, df = 1, p-value = 1.952e-05

# Run chi-squared test of the sex-differences in monogamy rates
chisq.test(table(mating$sex, mating$status)[,c(1,2)])
#> Warning in chisq.test(table(mating$sex, mating$status)[, c(1, 2)]): Chi-
#> squared approximation may be incorrect
#>
#>  Pearson's Chi-squared test with Yates' continuity correction
#>
#> data:  table(mating$sex, mating$status)[, c(1, 2)]
#> X-squared = 1.9908, df = 1, p-value = 0.1583

# Run chi-squared test of the sex-differences in mating behaviour rates
chisq.test(table(mating$sex, mating$status))
#> Warning in chisq.test(table(mating$sex, mating$status)): Chi-squared
#> approximation may be incorrect
#>
#>  Pearson's Chi-squared test
#>
```

```r
#> data:  table(mating$sex, mating$status)
#> X-squared = 23.629, df = 3, p-value = 2.986e-05

# Set the factor levels for plotting
mating$status <- factor(mating$status,
                        levels = c("Monogamous within years",
                                   "Monogamous between years",
                                   "Polygamous between years",
                                   "Polygamous within years"))

# Determine the number of males and females used in the analysis
sample_sizes_sex <- aggregate(focal ~ sex, data = mating, FUN = function(x){NROW(x)})

# Define the color palatte to use in the plot
custom_pal <- c("#7b3294", "#9E6BB1", "#91bfdb", "#4575b4")

# plot the sex-differences in mating behaviour
ggplot() +
  geom_bar(position = "fill", alpha = 0.75, data = mating, aes(x = sex, fill = status)) +
  geom_text(data = sample_sizes_sex, size = 3,
            aes(y = c(1.05, 1.05), x = c(1.11, 2.11), label = focal)) +
  annotate("text", x = c(0.92, 1.92), y = c(1.05, 1.05), label = "n = ", size = 3) +
  theme_bw() +
  theme(#text = element_text(family = "Arial"),
        legend.text = element_text(size = 10),
        legend.title = element_blank(),
        legend.position = "bottom",
        legend.key.height=unit(0.8,"line"),
        legend.key.width=unit(0.8,"line"),
        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(),
        strip.text.x = element_text(size=12),
        strip.background = element_blank(),
        strip.text = element_text(vjust = -10),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_line(size = 0.5, colour = "grey40"),
        axis.ticks.length = unit(0.2, "cm"),
        panel.border = element_rect(linetype = "solid", colour = "grey"),
        plot.margin = unit(c(0.2,0.2,-0.2,0.2), "cm")) +
  ylab("Proportion of individuals") +
  scale_fill_manual(values = custom_pal) +
  #facet_grid(. ~ sex) +
  scale_y_continuous(limits = c(0, 1.05)) +
  guides(fill = guide_legend(ncol = 1, byrow = TRUE))
```



R session information

```
sessionInfo()
#> R version 3.3.0 (2016-05-03)
#> Platform: x86_64-apple-darwin13.4.0 (64-bit)
#> Running under: OS X 10.11.6 (El Capitan)
#>
#> 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] stats     graphics  grDevices utils     datasets  methods   base
#>
#> other attached packages:
#>  [1] Rmisc_1.5          plyr_1.8.3         lattice_0.20-33
#>  [4] lme4_1.1-12        Matrix_1.2-6       extrafont_0.17
#>  [7] RColorBrewer_1.1-2 reshape2_1.4.1     gridExtra_2.2.1
#> [10] dplyr_0.4.3        ggplot2_2.1.0      stringr_1.0.0
#> [13] RMark_2.1.14       snowfall_1.84-6.1  snow_0.4-1
#>
#> loaded via a namespace (and not attached):
#>  [1] Rcpp_0.12.5      nloptr_1.0.4     formatR_1.4      tools_3.3.0
#>  [5] digest_0.6.9      nlme_3.1-128      evaluate_0.9      gtable_0.2.0
#>  [9] DBI_0.4-1         yaml_2.1.13       parallel_3.3.0    mvtnorm_1.0-5
#> [13] expm_0.999-0      coda_0.18-1       Rttf2pt1_1.3.4    knitr_1.13
#> [17] grid_3.3.0        R6_2.1.2          survival_2.39-4   rmarkdown_0.9.6
#> [21] minqa_1.2.4       extrafontdb_1.0   magrittr_1.5      codetools_0.2-14
#> [25] MASS_7.3-45       scales_0.4.0      htmltools_0.3.5   matrixcalc_1.0-3
#> [29] splines_3.3.0     assertthat_0.1    colorspace_1.2-6  labeling_0.3
#> [33] stringi_1.0-1     lazyeval_0.1.10   munsell_0.4.3     msm_1.6.1
```