

# 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*

*October 6, 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` 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\_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.
- **data/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.
- **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"))

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

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

---

## Fecundity

The objective here was to determine the average per capita annual fecundity for females. This vital rate was then incorporated into the two-sex matrix model. The second objective was to evaluate 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 column, 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"))
```

Determine how many individuals were included in the analysis

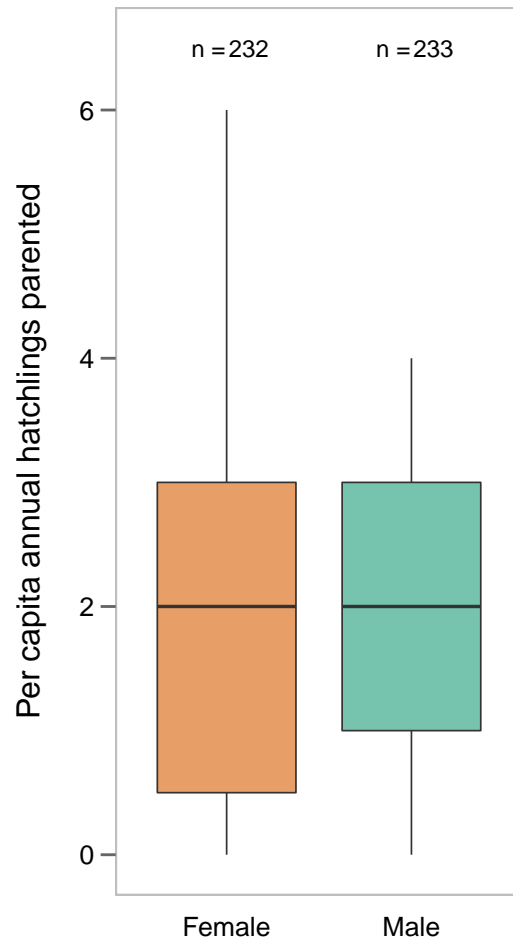
```
sample_sizes_sex <-
  aggregate(Ring ~ Sex, data = reproduction_df_sum_avg, FUN = function(x){NROW(x)})
```

specify the color palette to use for the plot

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

**Step three: plot the sex-specific distributions**

```
ggplot2::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$Sex == "Female"),
#> and reproduction_df_sum_avg[which(reproduction_df_sum_avg$Sex == "Male"),
#> F = 1.4651, num df = 231, denom df = 232, p-value = 0.0019
#> alternative hypothesis: true ratio of variances is greater than 1
#> 95 percent confidence interval:
#>  1.179765      Inf
#> sample estimates:
#> ratio of variances
#>      1.46513
```

Assign the value of female per capita annual fecundity to a constant that will be included in the matrix

```
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 procedure

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)
{
  # 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, demographic_rates$RF * (1 - HSR), 0, 0,
      # second row of matrix
      (demographic_rates$F_Chk_survl*demographic_rates$F_Fdg_survl),
      demographic_rates$F_Adt_survl,
      0, 0,
      # third row of matrix
      0, demographic_rates$RF * HSR, 0, 0,
      # fourth row of matrix
      0, 0,
      (demographic_rates$M_Chk_survl*demographic_rates$M_Fdg_survl),
      demographic_rates$M_Adt_survl),
    nrow = 4, byrow = TRUE,
    dimnames = list(stages, stages))
  result
}
```

**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 (A)
{
  # Number of stages in the matrix
  x <- ncol(A)
  {
    # determine the dominant eigen value and vector of the matrix
    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
}
```

## 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 *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) {

  # 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 <- fledgling_adult$bird_ID %in% chick_boot$bird_ID

  # extract these individuals from the adult data
  fledgling_adult_boot1 <- fledgling_adult[present, ]
}
```

```

# determine the left over adults
spare_fledgling_adult <- fledgling_adult[!present, ]

# randomly sample from these left over adults
fledgling_adult_boot2 <-
  spare_fledgling_adult[sample(1:nrow(spare_fledgling_adult),
                              size = nrow(fledgling_adult) -
                                nrow(fledgling_adult_boot1),
                              replace = TRUE), ]

# bind these two adult samples together
fledgling_adult_boot <- rbind(fledgling_adult_boot1, fledgling_adult_boot2)

# make a list of these two datasets, which will be used in the next function
out <- list(chick_boot = chick_boot, fledgling_adult_boot = fledgling_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"]]
  fledgling_adult <- plover_boot_list[["fledgling_adult_boot"]]

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

  # Create processed RMark data formatted as Cormack-Jolly-Seber with 2 groups
  # (sex and age initially 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 initially 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
# (uncomment the next three lines to assess)
# 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
# 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 fledgling and adult
# resight probability
fledgling_adult_survival = function()
{
  # sex- and stage-specific survival:
  Phi.agexsex = list(formula = ~ age * sex)

  # Models exploring variation in encounter probability
  # constant:
  p.dot = list(formula = ~ 1)
  # sex-dependent:
  p.sex = list(formula = ~ sex)
  # age-dependent:
  p.age = list(formula = ~ age)
  # factorial variation across year:
  p.Time = list(formula = ~ Time)
  # interaction between sex and factorial year:
  p.sexTime = list(formula = ~ sex * Time)
  # interaction between age and factorial year:
  p.agexTime = list(formula = ~ age * Time)
  # interaction between age and sex:
  p.agexsex = list(formula = ~ age * sex)
}

```

```

# additive effects of sex and factorial year:
p.sex_Time = list(formula = ~ sex + Time)
# additive effects of age and factorial year:
p.age_Time = list(formula = ~ age + Time)
# additive effects of age and sex:
p.age_sex = list(formula = ~ age + sex)
# additive effects of sex, age, factorial year:
p.Time_age_sex = list(formula = ~ Time + age + sex)
# additive effect of year and interaction between age and sex:
p.Time_age_x_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, delete unneeded output files, and
# run the models in Program MARK
model.list <- RMark::mark.wrapper(cml, data = fledgling_adult.proc,
                                ddl = fledgling_adult.ddl, delete = TRUE)

# output the model list and store 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$X2, "[AJ]"))
fledgling_adult_reals$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. 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 = ~ year * quadratic)
  # 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 = ~ year + quadratic)
  # additive effects of year, sex, and quadratic age
  p.year.quadratic.Sex = list(formula = ~ year + quadratic + sex)
  # additive effect of year and interaction between sex and quadratic age
  p.year.quadratic.x.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, 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 store 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")
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 hatching sex ratio
  HSR = HSR,
  # Define the fecundity of females (RF)
  RF = RF)

# 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(A = demographic_matrix)

# 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)
}

```

C) 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)
{
  # run the sampling function and specify the datasets
  bootstrap_data_list <- bootstrap_data(fledgling_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, fledgling_adult, chick)

```

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

AIC tables of chick survival for each iteration

```

# 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 fledgling and adult survival for each iteration

```

# 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)

```

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. *npars* 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\_fledgling\_adult\_boot\_out.txt** contains the bootstrap output for model selection of fledgling 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. *npars* 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\_age*) 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)

fldg_ad_AIC_tables <-
  read.table("output/bootstrap/AIC_table_fledgling_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 survival 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.

```
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)
```

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 distinguish first-year stages (i.e. chicks and fledglings) from adults

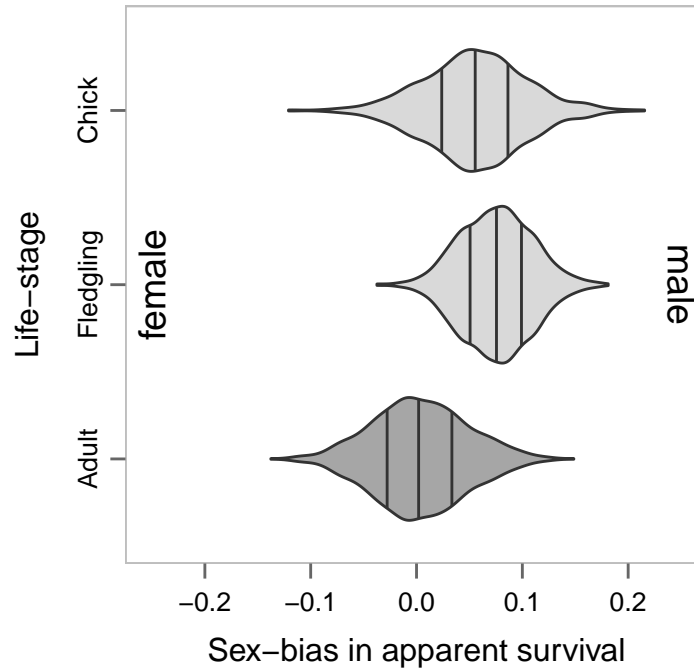
```
cbPalette <- c("#A6A6A6", "#D9D9D9", "#D9D9D9")
```

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

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")
```

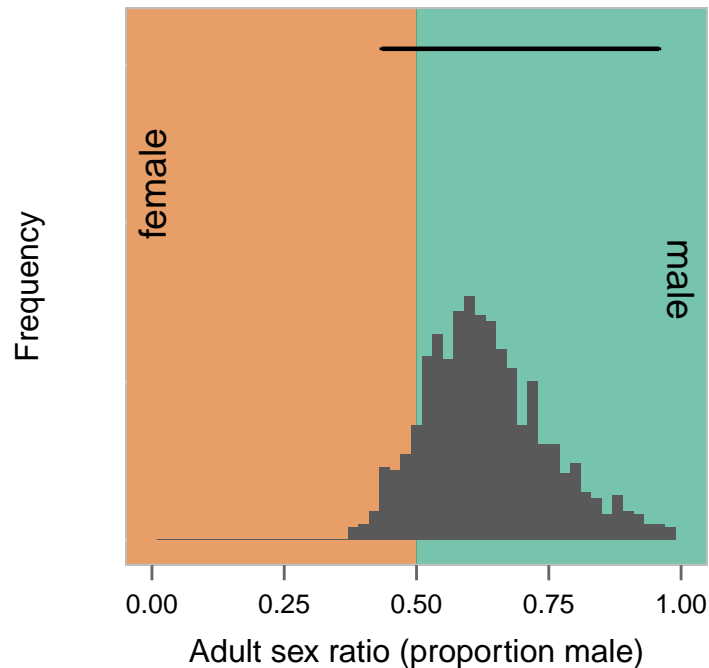
We visualized the bootstrapped results of adult sex ratio with a histogram. The horizontal black bar above the distribution illustrates the 95% confidence interval of the 1000 iterations.

```
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(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_boot)) +
  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))

```



AIC model selection summary (panels in Supplementary Material Figure 1)

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

First, wrangle the bootstrap AIC table output

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

fldg_ad_AIC_tables$model_number <- as.numeric(fldg_ad_AIC_tables$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))

fldg_ad_AIC_tables_summary <-
  fldg_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)

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

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

fldg_ad_AIC_tables <-
  dplyr::left_join(fldg_ad_AIC_tables_summary, fldg_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))

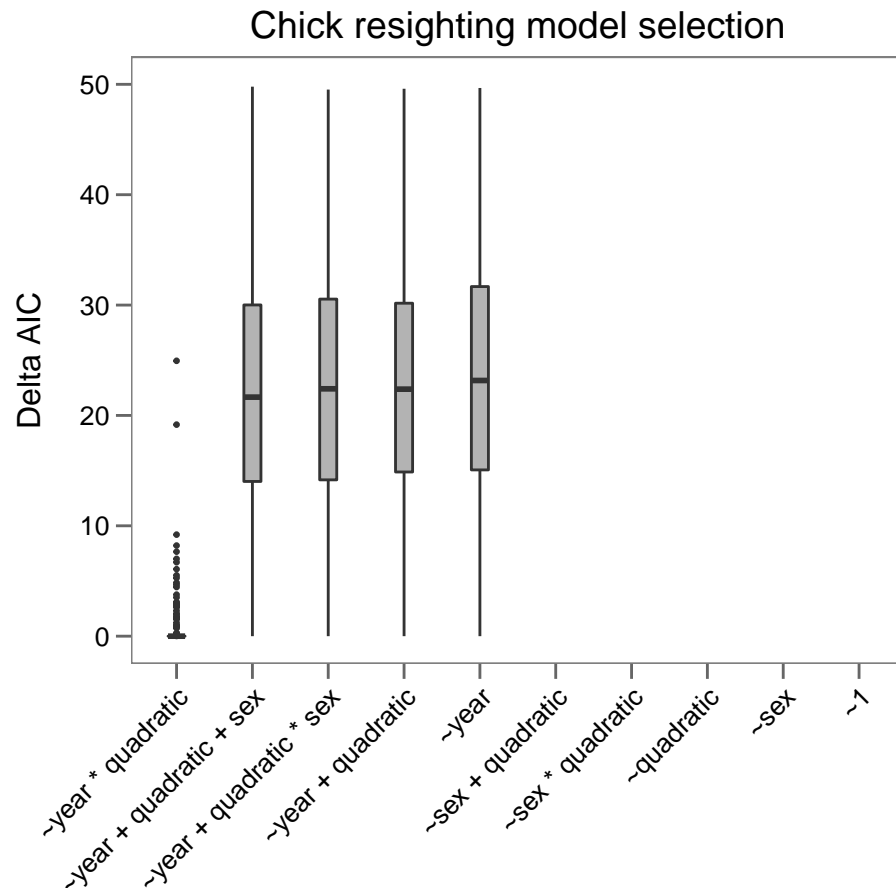
fldg_ad_AIC_tables$p <-
  factor(fldg_ad_AIC_tables$p,
         levels = str_sub(as.character(fldg_ad_AIC_tables_summary$model),
                           start = 18, end = str_length(fldg_ad_AIC_tables_summary$model)-1))
```

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

```

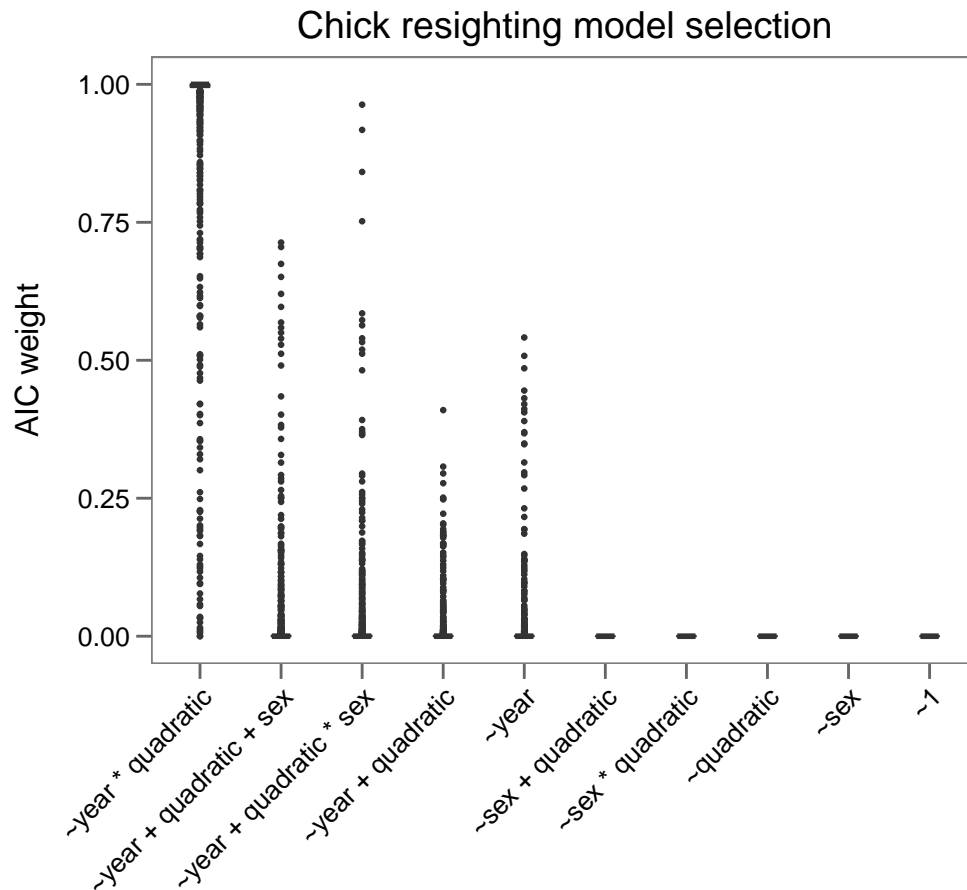
ggplot2::ggplot(aes(y = DeltaAICc, x = p), data = chick_AIC_tables) +
  theme_bw() +
  geom_boxplot(width = 0.3, fill = "grey70", outlier.size = 0.5) +
  theme(legend.position = "none",
        axis.title.x = element_blank(),
        axis.text.x = element_text(size=10, angle = 45, hjust = 1),
        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(0.5,0.5,0.5,3.0), "cm"),
        panel.margin = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  scale_y_continuous(limits=c(0,50)) +
  xlab("Model") +
  ylab("Delta AIC") +
  ggtitle("Chick resighting model selection")

```



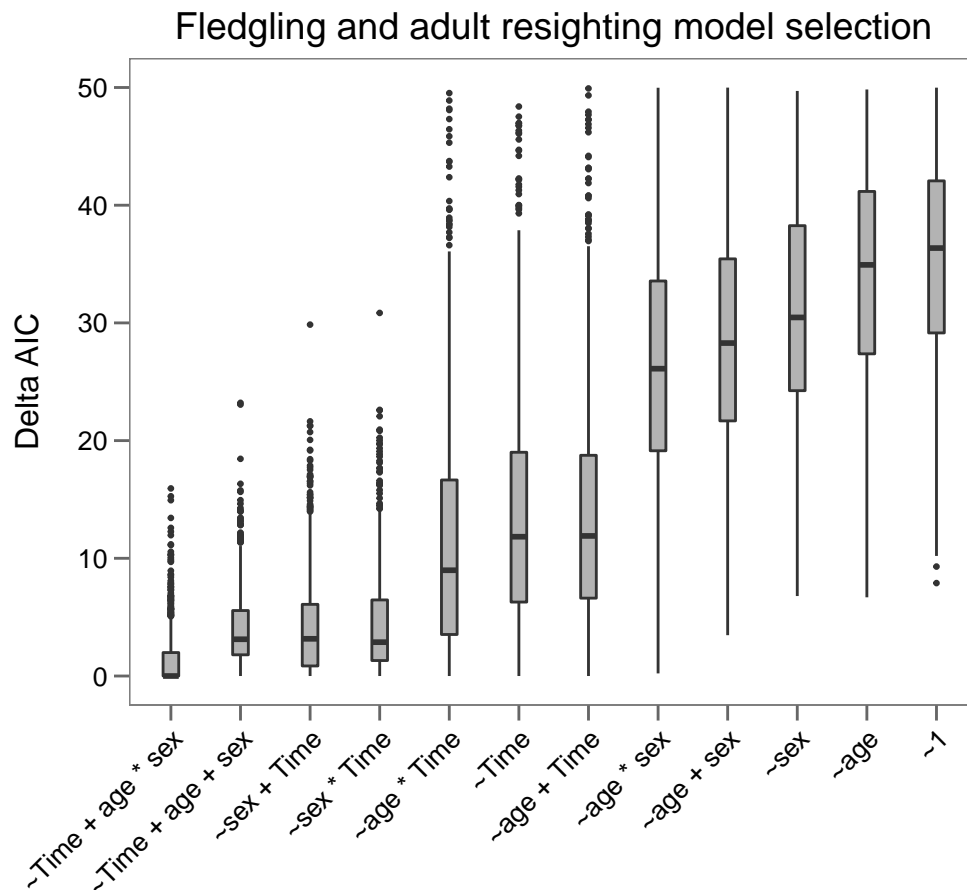
plot the overall model ranks of the chick survival analysis based on AIC weight

```
ggplot2::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(0.5,0.5,0.5,2.0), "cm"),
        panel.margin = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  scale_y_continuous(limits=c(0,1)) +
  xlab("Model") +
  ylab("AIC weight") +
  ggtitle("Chick resighting model selection")
```



plot the overall model ranks of the fledgling and adult survival analysis based on Delta AIC

```
ggplot2::ggplot(aes(y = DeltaAICc, x = p), data = fldg_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, 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(0.5,0.5,0.5,2.0), "cm"),
        panel.margin = unit(0.75, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  scale_y_continuous(limits=c(0,50)) +
  xlab("Model") +
  ylab("Delta AIC") +
  ggtitle("Fledgling and adult resighting model selection")
```



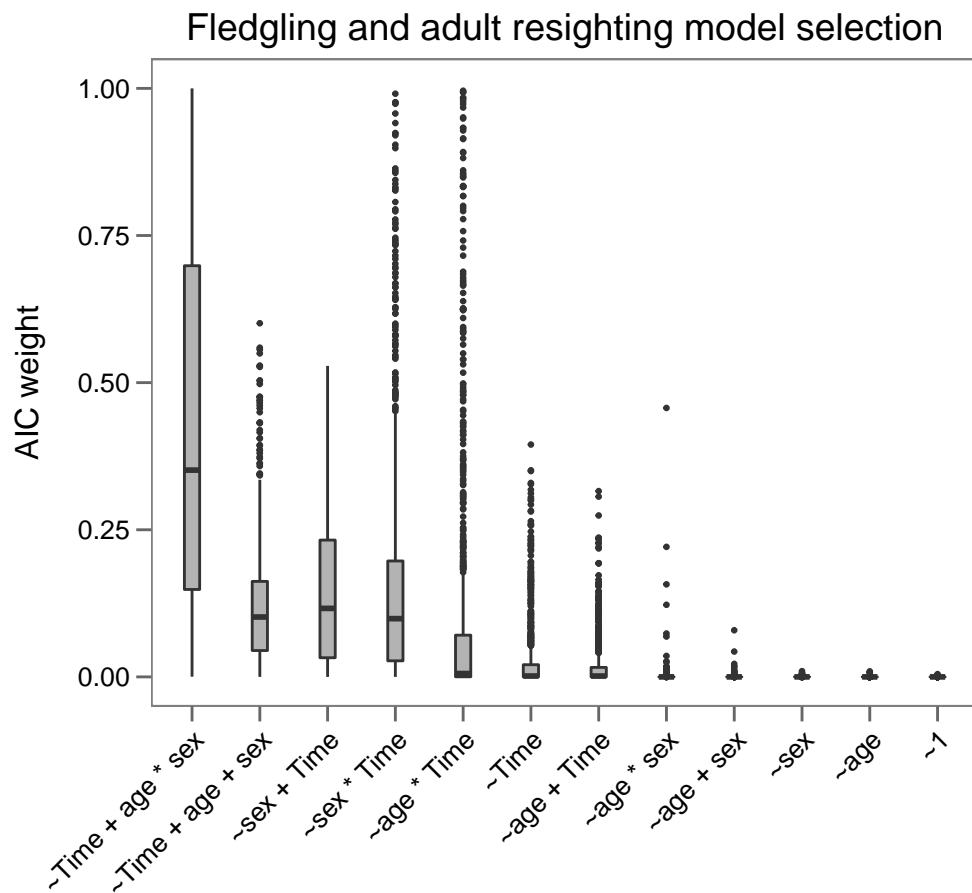
plot the overall model ranks of the fledgling and adult survival analysis based on AIC weight



```

ggplot2::ggplot(aes(y = weight, x = p), data = fldg_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, 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(0.5,0.5,0.5,2.0), "cm"),
        panel.margin = 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") +
  ggtitle("Fledgling and adult resighting model selection")

```



## Life table response experiment

Perturbation analyses provide important information about the relative contribution that each component of a matrix model has on the response, in our case ASR. To assess how influential a sex-bias in each of the three life-stages is on ASR dynamics, we employed a life-table response experiment (LTRE). In essence, the LTRE compares the response of a “control” matrix to that of a “treatment” matrix and assesses the relative contribution that model components have on the effect size of the treatment.

The following two functions need to be specified first. Note: these were modified from the **popbio** R package by Chris Stubben, Brook Milligan, and Patrick Nantel. Modifications were made so that the response was ASR rather than lambda.

**ASR\_analysis()** determines the stable-stage distribution from a matrix (A) and then calculates the ASR as the proportion of the adults in that distribution that are male.

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

```

e <- s * A/ASR

# get vital rate names
x <- dimnames(A)

# assign vital rate names to s
dimnames(s) <- x
names(w) <- x[[1]]
names(v) <- x[[1]]

# output a list containing the ASR, SSD, sensitivities, and elasticities
ASR.analysis <- list(ASR = ASR, stable.stage = stable.stage,
                    sensitivities = s, elasticities = e)
}
ASR.analysis
}

```

**ASR\_perturbation()** determines the lower-level sensitivities of each matrix element then runs the LTRE analysis.

```

ASR_perturbation <-
function (elements, VR_list, freq_dep_ASR)
{

# list of parameters in the treatment matrix
# this contains the observed parameters 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$RF,
  HSR = VR_list$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,
  HSR = 0.5)

# list of parameters in the M-prime matrix
# this contains the average difference between each parameter
M_prime_matrix <- list(
  F_Chk_survl = (treatment_matrix$F_Chk_survl + control_matrix$F_Chk_survl)/2,
  F_Fdg_survl = (treatment_matrix$F_Fdg_survl + control_matrix$F_Fdg_survl)/2,

```

```

F_Adt_surv1 = (treatment_matrix$F_Adt_surv1 + control_matrix$F_Adt_surv1)/2,
M_Chk_surv1 = (treatment_matrix$M_Chk_surv1 + control_matrix$M_Chk_surv1)/2,
M_Fdg_surv1 = (treatment_matrix$M_Fdg_surv1 + control_matrix$M_Chk_surv1)/2,
M_Adt_surv1 = (treatment_matrix$M_Adt_surv1 + control_matrix$M_Chk_surv1)/2,
RF = (treatment_matrix$RF + control_matrix$RF)/2,
HSR = (treatment_matrix$HSR + control_matrix$HSR)/2)

# 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 lower-level functions to the matrix
vrs_treatment <- try(sapply(elements, eval, treatment_matrix, NULL), silent = TRUE)
vrs_control <- try(sapply(elements, eval, control_matrix, NULL), silent = TRUE)
vrs_M_prime <- try(sapply(elements, eval, M_prime_matrix, NULL), silent = TRUE)

# check if its okay to proceed
if (class(vrs_treatment) == "try-error") {
  vrs_treatment <- sub("Error in eval\\(\\(expr, envir, enclos\\) : ",
    "", vrs_treatment[1])
  stop(paste("Cannot evaluate element expression using given vital rates:",
    vrs_treatment))
}
if (class(vrs_control) == "try-error") {
  vrs_control <- sub("Error in eval\\(\\(expr, envir, enclos\\) : ",
    "", vrs_control[1])
  stop(paste("Cannot evaluate element expression using given vital rates:",
    vrs_control))
}
if (class(vrs_M_prime) == "try-error") {
  vrs_M_prime <- sub("Error in eval\\(\\(expr, envir, enclos\\) : ",
    "", vrs_M_prime[1])
}

```

```

    stop(paste("Cannot evaluate element expression using given vital rates:",
              vrs_M_prime))
  }

  # 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_treatment <- matrix(vrs_treatment, nrow = n, byrow = TRUE)

  # build the control matrix
  A_control <- matrix(vrs_control, nrow = n, byrow = TRUE)

  # build the control matrix
  M_prime <- matrix(vrs_M_prime, nrow = n, byrow = TRUE)

  # run sensitivity analyses on both matrices
  ASR_M_prime <- ASR_analysis(M_prime)
  ASR_treatment <- ASR_analysis(A_treatment)

  # calculate derivatives of lower-level matrix elements
  deriv.funcs <- sapply(elements, deriv, namevec = names(treatment_matrix),
                        function.arg = TRUE)
  devs_treatment <- lapply(deriv.funcs, function(x) do.call(x, treatment_matrix))
  devs_m_prime <- lapply(deriv.funcs, function(x) do.call(x, M_prime_matrix))

  # run for loop to go through each parameter and estimate elasticity,
  # sensitivity, and LTRE
  for (i in 1:length(treatment_matrix)) {
    # first extract the
    derivs_treatment <-
      matrix(as.numeric(lapply(devs_treatment, function(x) attr(x, "gradient")[i])),
            nrow = n, byrow = TRUE)
    derivs_m_prime <-
      matrix(as.numeric(lapply(devs_m_prime, function(x) attr(x, "gradient")[i])),
            nrow = n, byrow = TRUE)

    res[i, 2] <-
      sum(derivs_treatment * ASR_treatment$sensitivities)
    res[i, 3] <-
      treatment_matrix[[i]] / ASR_treatment$ASR *
      sum(derivs_treatment * ASR_treatment$sensitivities)

    # only do LTRE calculations on survival and HSR parameters RELATIVE to one sex
    # i.e., don't calculate LTRE on fecundity
    res[i, 4] <- ifelse(i > 3 & i < 6, NA,
                      # calculate the sex differences of each survival rate, then multiply it
                      # by the sensitivity of that parameter in the M prime matrix
                      ifelse(i < 4, (treatment_matrix[[i + 3]] - treatment_matrix[[i]]) *
                                sum(derivs_m_prime * ASR_M_prime$sensitivities),
                              # calculate the difference in HSR from 0.5 and multiply it by its
                              # sensitivity

```

```

        ifelse(i == 8, (control_matrix[[i]] - treatment_matrix[[i]]) *
                  sum(derivs_m_prime * ASR_M_prime$sensitivities),
                  NA)))
    }

    # consolidate results
    y <- res
    y$Vital_rate <- as.factor(rownames(y))
    colnames(y) <- c("Estimate", "Sensitivity", "Elasticity", "LTRE", "Vital_rate")
    y_melt <- suppressMessages(reshape2::melt(y[,c(2:5)]))
    y_melt$parameter <-
      as.factor(ifelse(stringr::str_detect(y_melt$Vital_rate, "Chk"), "Chick survival",
        ifelse(stringr::str_detect(y_melt$Vital_rate, "Fdg"), "Fledgling survival",
          ifelse(stringr::str_detect(y_melt$Vital_rate, "Adt"), "Adult survival",
            "Hatching sex ratio"))))
    y_melt$parameter <- factor(y_melt$parameter, levels = c("Adult survival",
      "Fledgling survival",
      "Chick survival",
      "Hatching sex ratio"))
    y_melt$Sex <- as.factor(ifelse(stringr::str_detect(y_melt$Vital_rate, "F_") &
      y_melt$variable != "LTRE", "Female",
      ifelse(stringr::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),
    lcl = stats::quantile(estimate, (1 - CI)/2, na.rm = TRUE),
    ucl = stats::quantile(estimate, 1 - (1 - CI)/2, na.rm = TRUE))
survival_rates_boot_summary <- as.data.frame(survival_rates_boot_summary)
```

define deterministic 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],
  HSR = HSR,
  RF = RF)
```

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

```
matrix_structure <- expression(
  # top row of matrix
  0, RF * (1 - HSR), 0, 0,

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

  # third row of matrix
  0, RF * HSR, 0, 0,

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

)
```

assign deterministic vital rates to the matrix

```
deterministic_matrix <- plover_matrix(deterministic_list)
```

determine the ASR at the stable stage distribution

```
deterministic_ASR <-
  matrix_ASR(A = deterministic_matrix)
deterministic_ASR$ASR
#>      M_Adt
#> 0.6231446
```

run the life table response experiment

```
deterministic_LTRE <-  
  ASR_perturbation(elements = matrix_structure,  
                    VR_list = deterministic_list,  
                    freq_dep_ASR = deterministic_ASR)
```

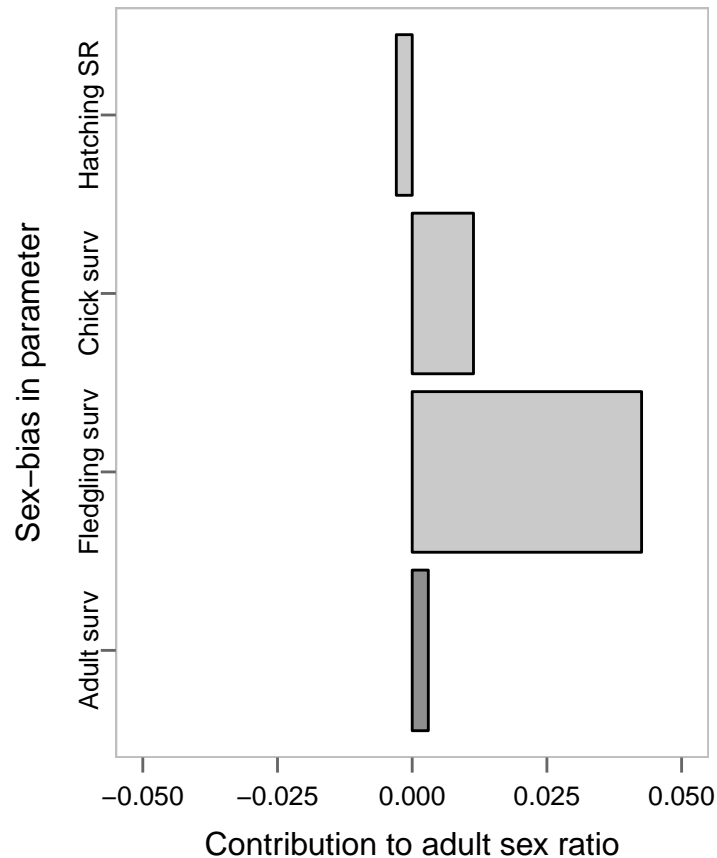
custom color palette for the plotting of Fledgling and Adult stats

```
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(-0.05, 0.05)) +  
  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:

```
deterministic_LTRE$LTRE[2,5]/deterministic_LTRE$LTRE[1,5]
#> [1] 3.743087
```

fledgling vs adult:

```
deterministic_LTRE$LTRE[2,5]/deterministic_LTRE$LTRE[3,5]
#> [1] 14.25118
```

chick vs adult:

```
deterministic_LTRE$LTRE[1,5]/deterministic_LTRE$LTRE[3,5]
#> [1] 3.807333
```

---

## Quantifying mating system

To put our estimate of ASR in the context of breeding behavior, we quantified sex-specific mating strategies and the reproductive success of snowy plovers at our study site. Females of this species desert broods to seek serial mates (Page et al. 2009). Thus, we expected that a larger number of females would exhibit within-season polygamy 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)),]
```

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 population

```
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)
```

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 = 15.05, df = 1, p-value = 0.0001047
```

run chi-squared test of the sex-differences in monogamy rates

```
chisq.test(table(mating$sex, mating$status)[,c(1,2)])
#>
#> Pearson's Chi-squared test with Yates' continuity correction
#>
#> data:  table(mating$sex, mating$status)[, c(1, 2)]
#> X-squared = 2.0102, df = 1, p-value = 0.1562
```

run chi-squared test of the sex-differences in mating behaviour rates

```
chisq.test(table(mating$sex, mating$status))
#>
#> Pearson's Chi-squared test
#>
#> data:  table(mating$sex, mating$status)
#> X-squared = 19.979, df = 3, p-value = 0.0001714
```

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

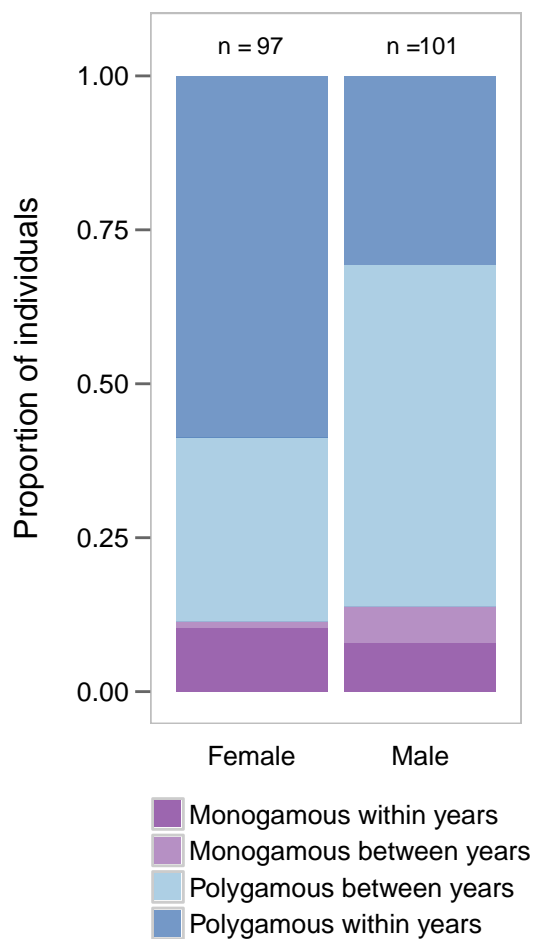
```
sample_sizes_sex <-  
  stats::aggregate(focal ~ sex, data = mating, FUN = function(x){NROW(x)})
```

define the color palatte to use in the plot

```
custom_pal <- c("#7b3294", "#9E6BB1", "#91bfdb", "#4575b4")
```

plot the sex-differences in mating behaviour

```
ggplot2::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(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) +  
  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] grid      stats      graphics  grDevices  utils      datasets  methods
#> [8] base
#>
#> other attached packages:
#> [1] magrittr_1.5      lme4_1.1-12      Matrix_1.2-7.1
#> [4] Rmisc_1.5         plyr_1.8.4       lattice_0.20-34
#> [7] RColorBrewer_1.1-2 reshape2_1.4.1   gridExtra_2.2.1
#> [10] dplyr_0.5.0       ggplot2_2.1.0    stringr_1.1.0
```

```

#> [13] RMark_2.2.0
#>
#> loaded via a namespace (and not attached):
#> [1] Rcpp_0.12.7      formatR_1.4      nloptr_1.0.4     tools_3.3.0
#> [5] digest_0.6.10    evaluate_0.9      tibble_1.2       gtable_0.2.0
#> [9] nlme_3.1-128     DBI_0.5-1        yaml_2.1.13      parallel_3.3.0
#> [13] mvtnorm_1.0-5    expm_0.999-0     coda_0.18-1      knitr_1.14
#> [17] R6_2.1.3         survival_2.39-5  rmarkdown_1.0    minqa_1.2.4
#> [21] scales_0.4.0     htmltools_0.3.5  matrixcalc_1.0-3 splines_3.3.0
#> [25] MASS_7.3-45      assertthat_0.1   colorspace_1.2-6 labeling_0.3
#> [29] stringi_1.1.1    lazyeval_0.2.0   munsell_0.4.3    msm_1.6.1

```