R-code for "Demographic causes of adult sex ratio variation in birds and their consequences for parental cooperation"

Luke J. Eberhart-Phillips, María Cristina Carmona-Isunza, Orsolya Vincze, Clemens Küpper, Sama Zefania, Tom E. X. Miller, Tamás Székely, Joseph I. Hoffman, and Oliver Krüger

February 28, 2017

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

The structure of the code we present here roughly follows the analyses presented in the *Supplementary Materials: Methods* 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)
library(extrafont)
```

Loading data

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

- **data/juvenile_adult_mark-recapture_data.txt** contains the mark-recapture field data of juveniles and adults. Each row is a single uniquely marked individual identified by their *bird_ID*. 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. *population* describes the population in which the individual was sampled from.
- **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. *population* describes the population in which the individual was sampled from.
- **data/hatching_sex_ratio_data.txt** contains the sex and origin of each chick included in our analysis to assess hatching sex ratio. Each row is a chick uniquely identified by their *chick_ID*. The family of origin for each chick is shown in their *nest_ID*. *year* describes the year in which the chick hatched. A "1" in either the *male* or *female* column indicates the molecular sex-type of a given chick. *population* describes the population in which the chick was sampled.
- **data/parental_care_data.txt** contains the behavioral observations of the care-system for each family. Each row is an observation of a unique family (*brood_ID*) within a given *population*. *care_system* expresses the parental care recorded on a given observation (i.e. "male_care", "female_care", or "biparental_care". *hatch_date* indicates the date on which a given brood hatched. *date* indicates the date on which the observation was made.

```r
# change this to your working directory where you downloaded our repository
setwd("~/Dropbox/Luke/R_projects/Plover_ASR_Matrix_Modeling")

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

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

hatching_sex_ratio <-
  read.table("data/hatching_sex_ratio_data.txt",
             header = TRUE)

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

Parental sex roles

To put our estimate of ASR in the context of breeding behavior, we quantified sex bias in parental care based on behavioral observations from the field.

First, some data formatting. Define the observation data (*date*) and the hatch date (*hatch_date*) variables as a date (*as.Date*, year-month-day):

```
parental_care_data$date <- as.Date(parental_care_data$date, "%Y-%m-%d")
parental_care_data$hatch_date <- as.Date(parental_care_data$hatch_date, "%Y-%m-%d")
```

Next, calculate the age of each brood when it was observed. Subtract the observation *date* from the *hatch_date*. The assign the brood observation into a *brood_period* describing it's age (early: less than 10 days old, middle: between 10 and 20 days old, late: older than 20 days).

```
parental_care_data$brood_age <- parental_care_data$date-parental_care_data$hatch_date

parental_care_data$brood_period <- ifelse(parental_care_data$brood_age < 10, "early",
                                          ifelse(parental_care_data$brood_age > 20 , "late",
                                                 "middle"))
```

Transform the *care_system* variable into a numeric to aquire a ranking system of the care levels (i.e., biparental_care = 1, female_care = 2, male_care = 3).

```
parental_care_data$care_system_num <- as.numeric(as.factor(parental_care_data$care_system))
```

To account for surveyor oversight while recording tending parents (i.e., misidentifying a bi-parental familiy as uni-parental), bi-parental status always trumped female_care or male_care in broods where there were observations of different states within a single *brood_period*. This step summarizes all the observations a given brood and assigns the lowest level observed (i.e., biparental_care = 1, female_care = 2, male_care = 3) to that brood. We were comfortable with *female_care* trumping *male_care* because female uni-parental care is a rare in plovers, so it is a conservative approach.

```
parental_care_summary <-
  parental_care_data %>%
    dplyr::group_by(population, brood_period, brood_ID) %>%
    dplyr::summarise(care_system = care_system[which.min(as.numeric(care_system_num))])
```

Calculate the proportion of late families in each population that exhibit *biparental_care*, *female_care*, or *male_care*.

```
# summarize the absolute number of families for each popualtion that are in the "late" brood_period.
sample_sizes <- reshape::cast(filter(parental_care_summary, brood_period == "late") %>%
                              dplyr::group_by(population, care_system) %>%
                              dplyr::summarise(n = n_distinct(brood_ID)),
                              population ~ care_system, sum)

# create a matrix of the summary
care_summary <- as.matrix(sample_sizes[, -1])
rownames(care_summary) <- sample_sizes[,1]

# calculate the proportions for each popualtion
care_summary_prop <- as.data.frame(prop.table(care_summary,1))

care_summary <- as.data.frame(care_summary)
care_summary$totals <- rowSums(care_summary)
```

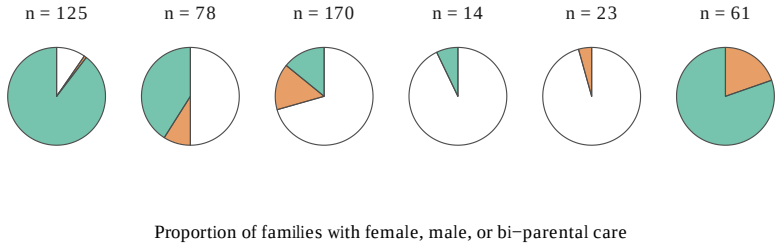
Table S2 in supplementary materials

```
cbind(care_summary_prop, care_summary)
#>     biparental_care female_care  male_care biparental_care female_care
#> KIP       0.0000000  0.19672131 0.80327869              0          12
#> KPM       0.7058824  0.15294118 0.14117647            120          26
#> KPT       0.5000000  0.08974359 0.41025641             39           7
#> MP        0.9285714  0.00000000 0.07142857             13           0
#> SP        0.0960000  0.00800000 0.89600000             12           1
#> WFP       0.9565217  0.04347826 0.00000000             22           1
#>     male_care totals
#> KIP        49     61
#> KPM        24    170
#> KPT        32     78
#> MP          1     14
#> SP        112    125
#> WFP         0     23
```

Figure 1c. Plot the pie charts illustrating the population-specific variation parental care

```
# define the levels of the population variable to determine the order of the plots in the figure.
parental_care_summary$population <- factor(parental_care_summary$population,
                                           levels = c("SP",
                                                      "KPT",
                                                      "KPM",
                                                      "MP",
                                                      "WFP",
                                                      "KIP"))

# specify the color palette (white = biparental_care, green = male_care, orange = female_care)
cbPalette <- c("white", RColorBrewer::brewer.pal(8, "Dark2")[c(2, 1)])

# calculate the sample sizes of families for each population
sample_sizes$n <- rowSums(sample_sizes[, -1])

# setup the sample size labels in the plot
n_labs_parents <- c(
  'SP'= paste("n = ", sample_sizes[sample_sizes$population == "SP", "n"], sep = ""),
  'KPT' = paste("n = ", sample_sizes[sample_sizes$population == "KPT", "n"], sep = ""),
  'KPM' = paste("n = ", sample_sizes[sample_sizes$population == "KPM", "n"], sep = ""),
  'MP' = paste("n = ", sample_sizes[sample_sizes$population == "MP", "n"], sep = ""),
  'WFP' = paste("n = ", sample_sizes[sample_sizes$population == "WFP", "n"], sep = ""),
  'KIP' = paste("n = ", sample_sizes[sample_sizes$population == "KIP", "n"], sep = "")
)

# draw the plot
Figure_1c <-
  ggplot(data = filter(parental_care_summary, brood_period == "late")) +
  geom_bar(mapping = aes(x = factor(1), fill = care_system), width = 1, position = "fill",
           size = 0.2, color = "#4d4d4d", alpha = 0.6) +
  coord_polar(theta = "y") +
  facet_grid(. ~ population, labeller = as_labeller(n_labs_parents)) +#, switch = "x") +
  theme_bw() +
  theme(text=element_text(family="Franklin Gothic Book"),
        legend.position="none",
        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=7, vjust=-0.5),
        axis.text.y  = element_blank(),
        axis.title.y = element_blank(),
        axis.text.x  = element_blank(),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_blank(),
        axis.ticks.length = unit(0.2, "cm"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        plot.margin = unit(c(0.4,0.2,0,1.2), "cm"),
        strip.background = element_blank(),
        strip.text = element_text(size=7),
        panel.margin = unit(0.3, "lines")) +
  scale_fill_manual(values = cbPalette) +
  ylab("\n\nProportion of families with female, male, or bi-parental care")
Figure_1c
```



Hatching sex ratio

The hatching sex ratio represents "rho" in the matrix model and is calculated from a dataset (*hatching_sex_ratio.txt*) that contains broods that met two criteria: 1) the brood size was the modal clutch size of the species, and 2) chicks were captured and sampled on the day of hatching. These criteria made sure to control for post-hatch brood mixing, which can occur in precocial species such as plovers.

First, for each population, we tested for significant deviations in the hatching sex ratio from parity. We used a binomial mixed-effects model that included *nest_ID* as a random effect to control for the non-independence of siblings.

Kittlitz's plover:

```
KIP_mod <-
  lme4::glmer(cbind(male, female) ~ (1|nest_ID),
              data = hatching_sex_ratio[which(hatching_sex_ratio$population == "KIP"),],
              family = binomial)

# non-significant
summary(KIP_mod)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
```

```
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(male, female) ~ (1 | nest_ID)
#>    Data:
#> hatching_sex_ratio[which(hatching_sex_ratio$population == "KIP"),      ]
#>
#>      AIC      BIC   logLik deviance df.resid
#>    203.2    209.1    -99.6    199.2      142
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.0572 -1.0572  0.9459  0.9459  0.9459
#>
#> Random effects:
#>  Groups  Name        Variance Std.Dev.
#>  nest_ID (Intercept) 0        0
#> Number of obs: 144, groups:  nest_ID, 72
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   0.1112     0.1669   0.666    0.505
```

Tuzla population of Kentish plover:

```
KPT_mod <-
  lme4::glmer(cbind(male, female) ~ (1|nest_ID),
              data = hatching_sex_ratio[which(hatching_sex_ratio$population == "KPT"),],
              family = binomial)

# non-significant
summary(KPT_mod)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(male, female) ~ (1 | nest_ID)
#>    Data:
#> hatching_sex_ratio[which(hatching_sex_ratio$population == "KPT"),      ]
#>
#>      AIC      BIC   logLik deviance df.resid
#>    367.1    374.3   -181.6    363.1      260
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.0154 -1.0154  0.9849  0.9849  0.9849
#>
#> Random effects:
#>  Groups  Name        Variance  Std.Dev.
#>  nest_ID (Intercept) 6.462e-14 2.542e-07
#> Number of obs: 262, groups:  nest_ID, 102
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  0.03054    0.12357   0.247    0.805
```

Maio population of Kentish plover:

```
KPM_mod <-
  lme4::glmer(cbind(male, female) ~ (1|nest_ID),
              data = hatching_sex_ratio[which(hatching_sex_ratio$population == "KPM"),],
              family = binomial)

# non-significant
summary(KPM_mod)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(male, female) ~ (1 | nest_ID)
#>    Data:
#> hatching_sex_ratio[which(hatching_sex_ratio$population == "KPM"),      ]
#>
#>      AIC      BIC   logLik deviance df.resid
#>    274.4    280.9   -135.2    270.4      195
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.0725 -0.8424 -0.7075  0.9157  1.1460
#>
#> Random effects:
#>  Groups  Name        Variance Std.Dev.
#>  nest_ID (Intercept) 0.596    0.772
#> Number of obs: 197, groups:  nest_ID, 107
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)  -0.0956     0.1724  -0.555    0.579
```

Snowy plover:

```
SP_mod <-
  lme4::glmer(cbind(male, female) ~ (1|nest_ID),
              data = hatching_sex_ratio[which(hatching_sex_ratio$population == "SP"),],
              family = binomial)

# non-significant
summary(SP_mod)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(male, female) ~ (1 | nest_ID)
#>    Data: hatching_sex_ratio[which(hatching_sex_ratio$population == "SP"),
#>     ]
#>
#>      AIC      BIC   logLik deviance df.resid
#>    673.1    681.5   -334.6    669.1      482
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -0.9398 -0.9398 -0.9398  1.0640  1.0640
#>
```

```
#> Random effects:
#>  Groups  Name        Variance  Std.Dev.
#>  nest_ID (Intercept) 3.851e-14 1.963e-07
#> Number of obs: 484, groups:  nest_ID, 198
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) -0.12413    0.09108  -1.363    0.173
```

White-fronted plover:

```
WFP_mod <-
  lme4::glmer(cbind(male, female) ~ (1|nest_ID),
              data = hatching_sex_ratio[which(hatching_sex_ratio$population == "WFP"),],
              family = binomial)

# non-significant
summary(WFP_mod)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(male, female) ~ (1 | nest_ID)
#>    Data:
#> hatching_sex_ratio[which(hatching_sex_ratio$population == "WFP"),      ]
#>
#>      AIC      BIC   logLik deviance df.resid
#>     42.6     45.4    -19.3     38.6       28
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.1803 -0.6573  0.4785  0.5275  0.8472
#>
#> Random effects:
#>  Groups  Name        Variance Std.Dev.
#>  nest_ID (Intercept) 1.58     1.257
#> Number of obs: 30, groups:  nest_ID, 13
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept) 0.591068   0.004593   128.7   <2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#> convergence code: 0
#> Model failed to converge with max|grad| = 0.015371 (tol = 0.001, component 1)
```

Madagascar plover:

```
MP_mod <-
  lme4::glmer(cbind(male, female) ~ (1|nest_ID),
              data = hatching_sex_ratio[which(hatching_sex_ratio$population == "MP"),],
              family = binomial)

# non-significant
summary(MP_mod)
#> Generalized linear mixed model fit by maximum likelihood (Laplace
```

```
#>   Approximation) [glmerMod]
#>  Family: binomial  ( logit )
#> Formula: cbind(male, female) ~ (1 | nest_ID)
#>    Data: hatching_sex_ratio[which(hatching_sex_ratio$population == "MP"),
#>     ]
#>
#>      AIC      BIC   logLik deviance df.resid
#>     32.8     35.0    -14.4     28.8       20
#>
#> Scaled residuals:
#>     Min      1Q  Median      3Q     Max
#> -1.3229 -1.3229  0.7559  0.7559  0.7559
#>
#> Random effects:
#>  Groups  Name        Variance Std.Dev.
#>  nest_ID (Intercept) 0        0
#> Number of obs: 22, groups:  nest_ID, 11
#>
#> Fixed effects:
#>             Estimate Std. Error z value Pr(>|z|)
#> (Intercept)   0.5596     0.4432   1.263    0.207
```

Summarize the hatching sex ratio data. First, calculate the proportion of each brood that is male

```
HSR_summary <-
  hatching_sex_ratio %>%
  dplyr::group_by(population, nest_ID) %>%
  dplyr::summarise(no_males = sum(male),
                   clutch_size = n_distinct(chick_ID))

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

Then calculate some summary statistics for each population. The population-specific *HSR* parameter is used as rho in the matrix later on.

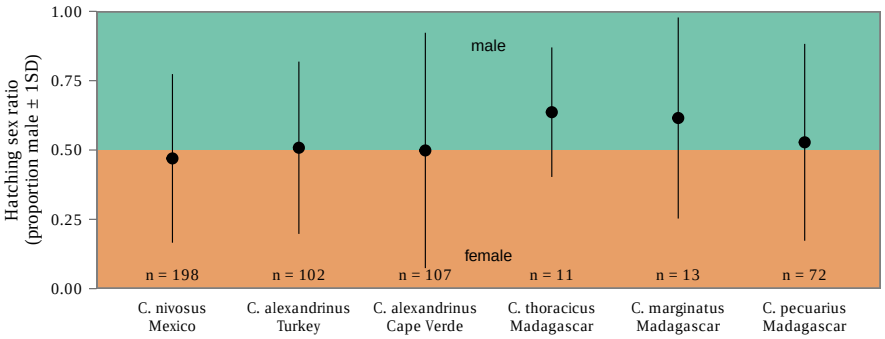
```
grand_HSR_summary <-
  HSR_summary %>%
  dplyr::group_by(population) %>%
  dplyr::summarise(HSR = mean(prop_male),
                   sd = sd(prop_male),
                   n_nests = n_distinct(nest_ID),
                   n_chicks = sum(clutch_size),
                   SE = sd(prop_male) / sqrt(length(prop_male)),
                   lower_CI =
                     mean(prop_male) - (sd(prop_male) / sqrt(length(prop_male)) * 1.96),
                   upper_CI =
                     mean(prop_male) + (sd(prop_male) / sqrt(length(prop_male)) * 1.96))

grand_HSR_summary
#> # A tibble: 6 × 8
#>   population       HSR        sd n_nests n_chicks         SE  lower_CI
#>       <fctr>     <dbl>     <dbl>   <int>    <int>      <dbl>     <dbl>
#> 1        KIP 0.5277778 0.3549339      72      144 0.04182936 0.4457922
#> 2        KPM 0.4984424 0.4242982     107      197 0.04101846 0.4180462
#> 3        KPT 0.5081699 0.3105918     102      262 0.03075317 0.4478937
```

```
#> 4         MP 0.6363636 0.2335497      11       22 0.07041788 0.4983446
#> 5         SP 0.4696970 0.3040093     198      484 0.02160500 0.4273512
#> 6        WFP 0.6153846 0.3625055      13       30 0.10054094 0.4183244
#> # ... with 1 more variables: upper_CI <dbl>
```

Figure S3. Plot the variation in hatching sex ratio across the 6 populations.

```
# create a column with the sample sizes used for each population (will be a label in the plot)
grand_HSR_summary$n_nests_sample_size <- paste("n = ", grand_HSR_summary$n_nests, sep = "")

# define the levels of the population variable to determine the order of the popualtions in the figure.
grand_HSR_summary$population <-
  factor(grand_HSR_summary$population ,
         levels = c("SP",
                    "KPT",
                    "KPM",
                    "MP",
                    "WFP",
                    "KIP"))

# specify the population names for labelling in the plot
population_names <- c(
  'SP'="C. nivosus\nMexico",
  'KPT'="C. alexandrinus\nTurkey",
  'MP'="C. thoracicus\nMadagascar",
  'KPM'="C. alexandrinus\nCape Verde",
  'WFP'="C. marginatus\nMadagascar",
  'KIP'="C. pecuarius\nMadagascar"
)

# draw the background plot that includes the male and female halves of color
Figure_S3_background <-
  ggplot2::ggplot(data = grand_HSR_summary, aes(y = HSR, x = population)) +
  theme_bw() +
  annotate("rect", xmin = 0, xmax = 6, ymin = 0, ymax = 0.5, alpha = 0.6,
           fill = RColorBrewer::brewer.pal(8, "Dark2")[c(2)]) +
  annotate("rect", xmin = 0, xmax = 6, ymin = 0.5, ymax = 1, alpha = 0.6,
           fill = RColorBrewer::brewer.pal(8, "Dark2")[c(1)]) +
  annotate("text", x = c(3), y = c(0.9),
           label = c("male"), size = 2,
           vjust = c(1), hjust = c(0.5)) +
  annotate("text", x = c(3), y = c(0.1),
           label = c("female"), size = 2,
           vjust = c(0), hjust = c(0.5)) +
  theme(text = element_text(family="Franklin Gothic Book",
                            colour = "white"),
        legend.position = "none",
        legend.background = element_rect(fill = NA),
        axis.title.y = element_text(size = 7),
        axis.text.y = element_text(size = 6),
        axis.title.x = element_blank(),
        axis.text.x = element_text(size = 6),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_line(size = 0.2, colour = "white"),
```

```
        axis.ticks.length = unit(0.1, "cm"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.margin = unit(0.75, "lines"),
        plot.margin = unit(c(0.2, 0.2, 0.43 ,0.2), "cm")) +
  scale_y_continuous(limits = c(0, 1), expand = c(0, 0)) +
  scale_x_continuous(limits=c(0, 6),breaks=c(0, 1, 2), expand = c(0, 0)) +
  ylab("Hatching sex ratio\n(proportion male ± 1SD)")


# draw the plot with the data
Figure_S3 <-
  ggplot2::ggplot() +
  theme_bw() +
  geom_pointrange(data = grand_HSR_summary, aes(y = HSR, x = population, ymin = HSR-sd,
                                                ymax = HSR+sd), size = 0.2) +
  geom_text(aes(y = 0.05, x = population, label = n_nests_sample_size,
                family = "Franklin Gothic Book"),
            data = grand_HSR_summary, size = 2) +
  theme(text = element_text(family="Franklin Gothic Book"),
        legend.position = "none",
        panel.background = element_rect(fill = "transparent",colour = NA),
        plot.background = element_rect(fill = "transparent",colour = NA),
        axis.title.y = element_text(size = 7),
        axis.text.y = element_text(size = 6),
        axis.title.x = element_blank(),
        axis.text.x = element_text(size = 6),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_line(size = 0.2, colour = "grey40"),
        axis.ticks.length = unit(0.1, "cm"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.margin = unit(0.75, "lines"),
        plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), "cm")) +
  scale_y_continuous(limits = c(0, 1), expand = c(0, 0)) +
  scale_x_discrete(labels = population_names) +
  ylab("Hatching sex ratio\n(proportion male ± 1SD)")

# overlay them on eachother to make the final plot
grid::pushViewport( grid::viewport(
  layout = grid::grid.layout(1, 1, widths = unit(1, "npc"))))
print(Figure_S3_background, newpage = FALSE)
print(Figure_S3, newpage = FALSE)
grid::popViewport()
```



Finally, specify the population-specific HSR parameters that will be used in the matrix model later

```
HSR_MP <- as.vector(unlist(grand_HSR_summary[which(grand_HSR_summary$population == "MP"), "HSR"]))
HSR_KIP <- as.vector(unlist(grand_HSR_summary[which(grand_HSR_summary$population == "KIP"), "HSR"]))
HSR_WFP <- as.vector(unlist(grand_HSR_summary[which(grand_HSR_summary$population == "WFP"), "HSR"]))
HSR_KPT <- as.vector(unlist(grand_HSR_summary[which(grand_HSR_summary$population == "KPT"), "HSR"]))
HSR_KPM <- as.vector(unlist(grand_HSR_summary[which(grand_HSR_summary$population == "KPM"), "HSR"]))
HSR_SP <- as.vector(unlist(grand_HSR_summary[which(grand_HSR_summary$population == "SP"), "HSR"]))
```

Quantifying mating system

To put our estimate of ASR in the context of breeding behavior, we quantified sex bias in mating system based on behavioral obersvations from the field. Females of *Charadrius* species are more likely to desert broods and seek serial mates than males. Thus, we expected that females would have more mates per year than males.

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

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

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

```
length(unique(mating_df$ID))
#> [1] 1668
```

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 + population ~ year)
males <- reshape2::dcast(mating_df, male + population ~ year)
```

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

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

join these two dataframes together and define as numeric

```
females <- dplyr::inner_join(females, number_males_p_female)
females[,c(3:16)] <-
```

```
  lapply(females[,c(3:16)], as.numeric)
males <- dplyr::inner_join(males, number_females_p_male)
males[,c(3:16)] <-
  lapply(males[,c(3:16)], as.numeric)
```

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

```
females$attempts <- rowSums(females[, c(3:16)])
males$attempts <- rowSums(males[, c(3:16)])
```

calculate the number of years breeding

```
females$years <- rowSums(females[, c(3:16)] > 0)
males$years <- rowSums(males[, c(3:16)] > 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$sex)
colnames(females_no_1)[c(1,17)] <- c("focal", "mate")
males_no_1$sex <- "Male"
males_no_1$sex <- as.factor(males_no_1$sex)
colnames(males_no_1)[c(1,17)] <- c("focal", "mate")
mating <- rbind(females_no_1, males_no_1)
```

calculate the number of mates per year

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

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

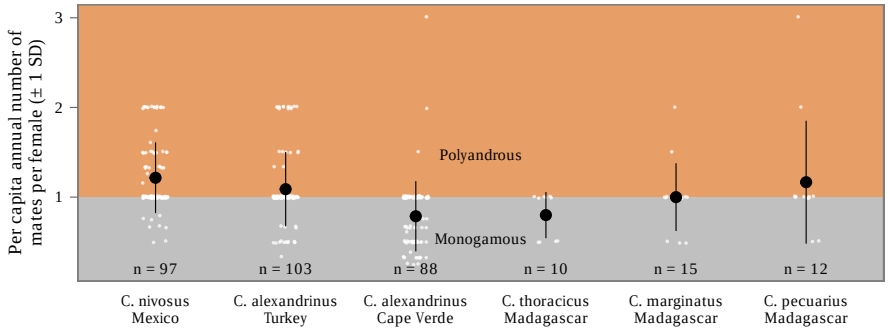
```
sex_specific_mating_system <-
  mating%>%
  dplyr::group_by(population, sex)%>%
  dplyr::summarise(mean_annual_no_mates = mean(no_mates_per_year),
                   var_annual_no_mates = var(no_mates_per_year),
                   median_annual_no_mates = median(no_mates_per_year),
                   sd_annual_no_mates = sd(no_mates_per_year),
                   n = n_distinct(focal))

sex_specific_mating_system$sample_size <- paste("n = ", sex_specific_mating_system$n, sep = "")

# When the mean annual number of mates is less than 1, the mating system is
# deemed monogamous and h = 1, but when females have more than 1 mates per year,
# the mating system is deemed polyandrous and h must be less than 1 (thus the
# inverse is calculated
KIP_h  <-
  ifelse(as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                       sex_specific_mating_system$population == "KIP"),
                                                 3]) < 1, 1,
         1/as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                         sex_specific_mating_system$population == "KIP"),
                                                   3]))
```

```
MP_h <-
  ifelse(as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                       sex_specific_mating_system$population == "MP"),
                                                     3]) < 1, 1,
         1/as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                         sex_specific_mating_system$population == "MP")
                                                       3]))

WFP_h <-
  ifelse(as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                       sex_specific_mating_system$population == "WFP"),
                                                     3]) < 1, 1,
         1/as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                         sex_specific_mating_system$population == "WFP")
                                                       3]))

KPT_h  <-
  ifelse(as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Male" &
                                                       sex_specific_mating_system$population == "KPT"),
                                                     3]) < 1, 1,
         1/as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Male" &
                                                         sex_specific_mating_system$population == "KPT")
                                                       3]))

KPM_h <-
  ifelse(as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                       sex_specific_mating_system$population == "KPM"),
                                                     3]) < 1, 1,
         1/as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                         sex_specific_mating_system$population == "KPM")
                                                       3]))

SP_h <-
  ifelse(as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                       sex_specific_mating_system$population == "SP"),
                                                     3]) < 1, 1,
         1/as.numeric(sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female" &
                                                         sex_specific_mating_system$population == "SP")
                                                       3]))

# display the h values for each population (these are used in the mating function of the matrix model)
KIP_h
#> [1] 0.8571429
MP_h
#> [1] 1
WFP_h
#> [1] 1
SP_h
#> [1] 0.8224986
KPM_h
#> [1] 1
KPT_h
#> [1] 0.8549422
```

Figure S4: plot the sex-specific distributions of mating system

```
# define the factor levels of the population variable so that the populations are in an
# order that reflects the ASR (male biased to female biased)
mating$population <-
  factor(mating$population ,
         levels = c("SP",
                    "KPT",
                    "KPM",
                    "MP",
                    "WFP",
                    "KIP"))

# draw the background plot with the orange and grey shades
Figure_S4_background <-
  ggplot2::ggplot(data = mating[which(mating$sex == "Female"),],
                  aes(y = 1/no_mates_per_year, x = population)) +
  theme_bw() +
  annotate("rect", xmin = 0, xmax = 6, ymin = 1.22, ymax = 4, alpha = 0.6,
           fill = RColorBrewer::brewer.pal(8, "Dark2")[c(2)]) +
  annotate("rect", xmin = 0, xmax = 6, ymin = 0, ymax = 1.22, alpha = 0.6,
           fill = RColorBrewer::brewer.pal(8, "Greys")[c(5)]) +
  annotate("text", x = c(3), y = c(0.7),
           label = c("Monogamous"), size = 2,
           family="Franklin Gothic Book", vjust = c(1), hjust = c(0.5)) +
  annotate("text", x = c(3), y = c(1.75),
           label = c("Polyandrous"), size = 2,
           family="Franklin Gothic Book", vjust = c(0), hjust = c(0.5)) +
  theme(text = element_text(family="Franklin Gothic Book", colour = "white"),
        legend.position = "none",
        legend.background = element_rect(fill = NA),
        axis.title.y = element_text(size = 7),
        axis.text.y = element_text(size = 6),
        axis.title.x = element_blank(),
        axis.text.x = element_text(size = 6),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_line(size = 0.2, colour = "white"),
        axis.ticks.length = unit(0.1, "cm"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.margin = unit(0.75, "lines"),
        plot.margin = unit(c(0.2, 0.2, 0.43,0.2), "cm")) +
  scale_y_continuous(expand = c(0, 0)) +
  scale_x_continuous(expand = c(0, 0)) +
  ylab("Per capita annual number of\nmates per female (± 1 SD)")

# draw the data
Figure_S4 <-
  ggplot2::ggplot() +
  theme_bw() +
  geom_jitter(aes(y = no_mates_per_year, x = population),
              data = mating[which(mating$sex == "Female"),], width = 0.25, alpha = 0.75,
              size = 0.3, fill = "white", color = "white", shape = 16) +
```

```
  geom_pointrange(data = sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female"),]
                  aes(y = mean_annual_no_mates, x = population,
                      ymin = (mean_annual_no_mates-sd_annual_no_mates),
                      ymax = (mean_annual_no_mates+sd_annual_no_mates)), size = 0.2) +
  geom_text(aes(y = 0.2, x = population, label = sample_size,
                family = "Franklin Gothic Book"),
            data = sex_specific_mating_system[which(sex_specific_mating_system$sex == "Female"),],
            size = 2) +
  theme(text = element_text(family="Franklin Gothic Book"),
        legend.position = "none",
        panel.background = element_rect(fill = "transparent",colour = NA),
        plot.background = element_rect(fill = "transparent",colour = NA),
        axis.title.y = element_text(size = 7),
        axis.text.y = element_text(size = 6),
        axis.title.x = element_blank(),
        axis.text.x = element_text(size = 6),
        axis.ticks.x = element_blank(),
        axis.ticks.y = element_line(size = 0.2, colour = "grey40"),
        axis.ticks.length = unit(0.1, "cm"),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.margin = unit(0.75, "lines"),
        plot.margin = unit(c(0.2, 0.2, 0.2, 0.2), "cm")) +
  scale_x_discrete(labels = population_names) +
  ylab("Per capita annual number of\nmates per female (± 1 SD)")

# overlay the two plots ontop of eachother to make the final plot
grid::pushViewport( grid::viewport(
  layout = grid::grid.layout(1, 1, widths = unit(1, "npc"))))
print(Figure_S4_background, newpage = FALSE)
print(Figure_S4, newpage = FALSE)
grid::popViewport()
```



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

```
# significantly female-biased
wilcox.test(no_mates_per_year ~ sex, data = filter(mating, population == "SP"))
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  no_mates_per_year by sex
#> W = 6533, p-value = 2.994e-06
```

```
#> alternative hypothesis: true location shift is not equal to 0

# non-significant
wilcox.test(no_mates_per_year ~ sex, data = filter(mating, population == "KPT"))
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  no_mates_per_year by sex
#> W = 5238.5, p-value = 0.2309
#> alternative hypothesis: true location shift is not equal to 0

# non-significant
wilcox.test(no_mates_per_year ~ sex, data = filter(mating, population == "KPM"))
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  no_mates_per_year by sex
#> W = 3780.5, p-value = 0.4757
#> alternative hypothesis: true location shift is not equal to 0

# non-significant
wilcox.test(no_mates_per_year ~ sex, data = filter(mating, population == "MP"))
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  no_mates_per_year by sex
#> W = 53, p-value = 0.9002
#> alternative hypothesis: true location shift is not equal to 0

# non-significant
wilcox.test(no_mates_per_year ~ sex, data = filter(mating, population == "WFP"))
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  no_mates_per_year by sex
#> W = 118.5, p-value = 0.9589
#> alternative hypothesis: true location shift is not equal to 0

# non-significant
wilcox.test(no_mates_per_year ~ sex, data = filter(mating, population == "KIP"))
#>
#>  Wilcoxon rank sum test with continuity correction
#>
#> data:  no_mates_per_year by sex
#> W = 66.5, p-value = 1
#> alternative hypothesis: true location shift is not equal to 0
```

Bootstrapping proceedure of stage- and sex specific survival

Specify where RMark should look on your computer for Program MARK. This may vary based on your operating system (e.g., Windows, Linux, Mac OS X, etc.). This website provides a nice workflow for installing

Program MARK and linking it to your R interface based on which operating system you have.

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

Step one: Assign functions

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

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

```
plover_matrix <-
  function(demographic_rates){
      # Define plover life-stages of the two-sex plover matrix model
      stages <- c("F_1st_yr",  "F_Adt",  "M_1st_yr",  "M_Adt")
      # Build the 4x4 matrix
      result <-
        matrix(c(
                  # top row of matrix
                  0, NA, 0, NA,
                  # second row of matrix
                  demographic_rates$F_Juv_survl,
                  demographic_rates$F_Adt_survl,
                  0, 0,
                  # third row of matrix
                  0, NA, 0, NA,
                  # fourth row of matrix
                  0, 0,
                  demographic_rates$M_Juv_survl,
                  demographic_rates$M_Adt_survl),
                nrow = length(stages), byrow = TRUE,
                dimnames = list(stages, stages))
      result
  }
```

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

```
matrix_ASR <-
  function(M, n = rep(10, nrow(M)), h = 1, k = 3,
           iterations = 1000, HSR = 0.5, plot = FALSE){
    # Number of stages in matrix
    x <- length(n)
    # Number of time steps to simulate
    t <- iterations
    # an empty t by x matrix to store the stage distributions
    stage <- matrix(numeric(x * t), nrow = x)
    # an empty t vector to store the population sizes
    pop <- numeric(t)
    # for loop that goes through each of t time steps
    for (i in 1:t) {
      # stage distribution at time t
      stage[,i] <- n
```

```
    # population size at time t
    pop[i] <- sum(n)
    # number of male adults at time t
    M2 <- stage[4, i]
    # number of female adults at time t
    F2 <- stage[2, i]
    # Female freq-dep fecundity of Female chicks
    M[1,x/2]        <- (k*M2)/(M2+(F2*h))*HSR
    # Female freq-dep fecundity of Male chicks
    M[(x/4)*3,x/2]  <- (k*M2)/(M2+(F2*h))*HSR
    # Male freq-dep fecundity of Female chicks
    M[1,x]          <- (k*F2)/(M2+(F2*h))*HSR
    # Male freq-dep fecundity of Male chicks
    M[(x/4)*3,x]    <- (k*F2)/(M2+(F2*h))*HSR
    # define the new n (i.e., new stage distribution at time t)
    n <- M %*% n
    # define rownames of stage matrix
    rownames(stage) <- rownames(M)
    # define colnames of stage matrix
    colnames(stage) <- 0:(t - 1)
    # calculate the proportional stable stage distribution
    stage <- apply(stage, 2, function(x) x/sum(x))
    # define stable stage as the last stage
    stable.stage <- stage[, t]
  }
  # calc ASR as the proportion of the adult stable stage class that is male
  ASR <- stable.stage[x]/(stable.stage[x/2] + stable.stage[x])

  if(plot)
  {
    # plot distrubution to assure that it is not chaotic
    matplot(rownames(t(stage)), t(stage), type='l', lwd=2, las=1)
  }
  # make a list of results
  pop.proj <- list(ASR = ASR,
                   stable.stage = stable.stage,
                   stage.vectors = stage,
                   SSD_M2 = stable.stage[4],
                   SSD_F2 = stable.stage[2])
  # print the list as output to the function
  pop.proj
}
```

Step two: running the bootstrap

Each iteration will do the following computational steps:

A) Load the following function **bootstrap_data()** to randomly sample with replacement *juvenile_adult* dataset. Each bootstrapped sample has the same length as the original data.

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

  # subset the data to include only the population of interest
  juvenile_adult2 <- dplyr::filter(juvenile_adult, population == pop_name)
```

```
  # sample a new chick mark-recapture dataset the same size as the original,
  # with replacement
  juvenile_adult_boot <- juvenile_adult2[sample(1:nrow(juvenile_adult2),
                                                size = nrow(juvenile_adult2),
                                                replace = TRUE), ]
}
```

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, *juvenile_adult_boot* is the output dataframe from **bootstrap_data()** and *num_boot* is the bootstrap number in the loop (leave unspecified).

```
bootstrap_survival_ASR <- function(juvenile_adult_boot, num_boot, start_year,
                                   num_years, h, HSR, k) {

  # specify the bootstrapped data sample (from the previous function)
  juvenile_adult <- juvenile_adult_boot

  # remove bird_ID column
  juvenile_adult <- juvenile_adult[,-1]

  # Create processed RMark data formatted 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.
  juvenile_adult.proc <- RMark::process.data(juvenile_adult, model = "CJS",
                                             groups = c("sex", "age"),
                                             begin.time = start_year, age.var = 2,
                                             initial.age = c(1, 0))

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

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

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

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

```
#           output = F), "Phi")

# create the function that specifies the candidate models of juvenile and adult
# resight probability
juvenile_adult_survival = function()
{
  # sex- and stage-specific survival:
  Phi.agexsex = list(formula = ~ 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)
  # linear variation across time:
  p.time = list(formula =  ~ time)
  # Quadratic variation across time:
  p.Quadratic = list(formula =  ~ Quadratic)
  # interaction between sex and factorial year:
  p.sexxTime = list(formula =  ~ sex * Time)
  # interaction between age and factorial year:
  p.agexTime = list(formula =  ~ age * Time)
  # interaction between sex and linear year:
  p.sexxtime = list(formula =  ~ sex * time)
  # interaction between age and linear year:
  p.agextime = list(formula =  ~ age * time)
  # interaction between sex and Quadratic year:
  p.sexxQuadratic = list(formula =  ~ sex * Quadratic)
  # interaction between age and Quadratic year:
  p.agexQuadratic = list(formula =  ~ age * Quadratic)
  # 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 sex and linear year:
  p.sex_time = list(formula =  ~ sex + time)
  # additive effects of age and linear year:
  p.age_time = list(formula =  ~ age + time)
  # additive effects of sex and Quadratic year:
  p.Quadratic_sex = list(formula =  ~ Quadratic + sex)
  # additive effects of age and Quadratic year:
  p.Quadratic_age = list(formula =  ~ Quadratic + age)
  # 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 effects of sex, age, linear year:
```

```
  p.time_age_sex = list(formula =  ~ time + age + sex)
  # additive effects of sex, age, Quadratic year:
  p.Quadratic_age_sex = list(formula =  ~ Quadratic + age + sex)
  # interaction between factorial year, age and sex:
  p.Time_age_x_sex = list(formula =  ~ Time * age * sex)
  # interaction between linear year, age and sex:
  p.timexagexsex = list(formula =  ~ time * age * sex)
  # interaction between Quadratic year, age and sex:
  p.Quadraticxagexsex = list(formula =  ~ Quadratic * 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 = juvenile_adult.proc,
                                     ddl = juvenile_adult.ddl, delete = TRUE)

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

# Run the models on the bootstrapped data
juvenile_adult_survival_run <-
  juvenile_adult_survival()

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

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

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

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

```
    paste(juvenile_adult_reals$sex,juvenile_adult_reals$age,sep = "_")
  survival_rates <-
    juvenile_adult_reals[,c("sex_age", "estimate")]
  row.names(survival_rates) <- NULL

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

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

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

  # Extract ASR
  ASR_estimate <- ASR_SSD$ASR

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

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

```
run_bootstrap_survival_ASR <- function(num_boot, juvenile_adult, pop_name,
                                       start_year, num_years, k, h, HSR) {
  # run the sampling function and specify the datasets
  bootstrap_dataframe <- bootstrap_data(juvenile_adult, pop_name)

  # run the survival analysis and ASR deduction on the sampled data
  result <- bootstrap_survival_ASR(bootstrap_dataframe, num_boot, start_year, num_years, k, h, HSR)
}
```

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 section to run the bootstrap. To bypass this, load the bootstrap output datasets in the next section to continue the analysis.

```
# KiP_survival_ASR_bootstrap_result <-
#   sapply(1:niter, run_bootstrap_survival_ASR, juvenile_adult, pop_name = "KIP",
```

```
#          start_year = 2009, num_years = 7, k = 2, HSR = HSR_KIP, h = KIP_h)
# MP_survival_ASR_bootstrap_result <-
#   sapply(1:niter, run_bootstrap_survival_ASR, juvenile_adult, pop_name = "MP",
#          start_year = 2009, num_years = 7, k = 2, HSR = HSR_MP, h = MP_h)
# WfP_survival_ASR_bootstrap_result <-
#   sapply(1:niter, run_bootstrap_survival_ASR, juvenile_adult, pop_name = "WFP",
#          start_year = 2009, num_years = 7, k = 3, HSR = HSR_WFP, h = WFP_h)
# KPT_survival_ASR_bootstrap_result <-
#   sapply(1:niter, run_bootstrap_survival_ASR, juvenile_adult, pop_name = "KPT",
#          start_year = 1996, num_years = 6, k = 3, HSR = HSR_KPT, h = KPT_h)
# KPM_survival_ASR_bootstrap_result <-
#   sapply(1:niter, run_bootstrap_survival_ASR, juvenile_adult, pop_name = "KPM",
#          start_year = 2007, num_years = 9, k = 3, HSR = HSR_KPM, h = KPM_h)
# SP_survival_ASR_bootstrap_result <-
#   sapply(1:niter, run_bootstrap_survival_ASR, juvenile_adult, pop_name = "SP",
#          start_year = 2006, num_years = 7, k = 3, HSR = HSR_SP, h = SP_h)
```

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

AIC tables of juvenile and adult survival for each interation

```
# # Kittlitz's plover
# KiP_AIC_table_juvenile_adult_boot <-
# do.call(rbind, lapply(seq(from = 1, to = niter * 3, by = 3),
#                       function(x) KiP_survival_ASR_bootstrap_result[[x]]))
# num_mods <- nrow(KiP_AIC_table_juvenile_adult_boot)/niter
# KiP_AIC_table_juvenile_adult_boot$iter <- rep(1:niter, each = num_mods)
# KiP_AIC_table_juvenile_adult_boot$population <- "KIP"
#
# # Madagascar plover
# MP_AIC_table_juvenile_adult_boot <-
# do.call(rbind, lapply(seq(from = 1, to = niter * 3, by = 3),
#                       function(x) MP_survival_ASR_bootstrap_result[[x]]))
# num_mods <- nrow(MP_AIC_table_juvenile_adult_boot)/niter
# MP_AIC_table_juvenile_adult_boot$iter <- rep(1:niter, each = num_mods)
# MP_AIC_table_juvenile_adult_boot$population <- "MP"
#
# # white-fronted plover
# WfP_AIC_table_juvenile_adult_boot <-
# do.call(rbind, lapply(seq(from = 1, to = niter * 3, by = 3),
#                       function(x) WfP_survival_ASR_bootstrap_result[[x]]))
# num_mods <- nrow(WfP_AIC_table_juvenile_adult_boot)/niter
# WfP_AIC_table_juvenile_adult_boot$iter <- rep(1:niter, each = num_mods)
# WfP_AIC_table_juvenile_adult_boot$population <- "WfP"
#
# # Tuzla Kentish plover
# KPT_AIC_table_juvenile_adult_boot <-
# do.call(rbind, lapply(seq(from = 1, to = niter * 3, by = 3),
#                       function(x) KPT_survival_ASR_bootstrap_result[[x]]))
# num_mods <- nrow(KPT_AIC_table_juvenile_adult_boot)/niter
# KPT_AIC_table_juvenile_adult_boot$iter <- rep(1:niter, each = num_mods)
# KPT_AIC_table_juvenile_adult_boot$population <- "KPT"
```

```
#
# # Maio Kentish plover
# KPM_AIC_table_juvenile_adult_boot <-
# do.call(rbind, lapply(seq(from = 1, to = niter * 3, by = 3),
#                       function(x) KPM_survival_ASR_bootstrap_result[[x]]))
# num_mods <- nrow(KPM_AIC_table_juvenile_adult_boot)/niter
# KPM_AIC_table_juvenile_adult_boot$iter <- rep(1:niter, each = num_mods)
# KPM_AIC_table_juvenile_adult_boot$population <- "KPM"
#
# # snowy plover
# SP_AIC_table_juvenile_adult_boot <-
# do.call(rbind, lapply(seq(from = 1, to = niter * 3, by = 3),
#                       function(x) SP_survival_ASR_bootstrap_result[[x]]))
# num_mods <- nrow(SP_AIC_table_juvenile_adult_boot)/niter
# SP_AIC_table_juvenile_adult_boot$iter <- rep(1:niter, each = num_mods)
# SP_AIC_table_juvenile_adult_boot$population <- "SP"
#
# AIC_table_juvenile_adult_boot_out <- rbind(KiP_AIC_table_juvenile_adult_boot,
#                                            WfP_AIC_table_juvenile_adult_boot,
#                                            MP_AIC_table_juvenile_adult_boot,
#                                            KPT_AIC_table_juvenile_adult_boot,
#                                            KPM_AIC_table_juvenile_adult_boot,
#                                            SP_AIC_table_juvenile_adult_boot)
```

Survival rates for each iteration

(uncomment these sections if you ran the bootstrap)

```
# # Kittlitz's plover
# KiP_survival_rates_boot <-
# do.call(rbind, lapply(seq(from = 2, to = niter * 3, by = 3),
#                       function(x) KiP_survival_ASR_bootstrap_result[[x]]))
# KiP_survival_rates_boot$iter <- rep(1:niter, each = 4)
# KiP_survival_rates_boot$population <- "KIP"
#
# # Madagascar plover
# MP_survival_rates_boot <-
# do.call(rbind, lapply(seq(from = 2, to = niter * 3, by = 3),
#                       function(x) MP_survival_ASR_bootstrap_result[[x]]))
# MP_survival_rates_boot$iter <- rep(1:niter, each = 4)
# MP_survival_rates_boot$population <- "MP"
#
# # white-fronted plover
# WfP_survival_rates_boot <-
# do.call(rbind, lapply(seq(from = 2, to = niter * 3, by = 3),
#                       function(x) WfP_survival_ASR_bootstrap_result[[x]]))
# WfP_survival_rates_boot$iter <- rep(1:niter, each = 4)
# WfP_survival_rates_boot$population <- "WFP"
#
# # Tuzla Kentish plover
# KPT_survival_rates_boot <-
# do.call(rbind, lapply(seq(from = 2, to = niter * 3, by = 3),
#                       function(x) KPT_survival_ASR_bootstrap_result[[x]]))
```

```
# KPT_survival_rates_boot$iter <- rep(1:niter, each = 4)
# KPT_survival_rates_boot$population <-"KPT"
#
# # Maio Kentish plover
# KPM_survival_rates_boot <-
# do.call(rbind, lapply(seq(from = 2, to = niter * 3, by = 3),
#                       function(x) KPM_survival_ASR_bootstrap_result[[x]]))
# KPM_survival_rates_boot$iter <- rep(1:niter, each = 4)
# KPM_survival_rates_boot$population <- "KPM"
#
# # snowy plover
# SP_survival_rates_boot <-
# do.call(rbind, lapply(seq(from = 2, to = niter * 3, by = 3),
#                       function(x) SP_survival_ASR_bootstrap_result[[x]]))
# SP_survival_rates_boot$iter <- rep(1:niter, each = 4)
# SP_survival_rates_boot$population <- "SP"
#
# survival_rates_boot_out <- rbind(SP_survival_rates_boot,
#                                  KiP_survival_rates_boot,
#                                  KPT_survival_rates_boot,
#                                  KPM_survival_rates_boot,
#                                  MP_survival_rates_boot,
#                                  WfP_survival_rates_boot)
```

ASR estimate for each iteration

(uncomment these sections if you ran the bootstrap)

```
# # Kittlitz'z plover
# KiP_ASR_boot <-
# sapply(seq(from = 3, to = niter * 3, by = 3),
#        function(x) KiP_survival_ASR_bootstrap_result[[x]])
# KiP_ASR_boot <- data.frame(ASR_boot = unname(KiP_ASR_boot), iter = 1:niter, population = "KIP")
#
# # Madagascar plover
# MP_ASR_boot <-
# sapply(seq(from = 3, to = niter * 3, by = 3),
#        function(x) MP_survival_ASR_bootstrap_result[[x]])
# MP_ASR_boot <- data.frame(ASR_boot = unname(MP_ASR_boot), iter = 1:niter, population = "MP")
#
# # white-fronted plover
# WfP_ASR_boot <-
# sapply(seq(from = 3, to = niter * 3, by = 3),
#        function(x) WfP_survival_ASR_bootstrap_result[[x]])
# WfP_ASR_boot <- data.frame(ASR_boot = unname(WfP_ASR_boot), iter = 1:niter, population = "WFP")
#
# # Tuzla Kentish plover
# KPT_ASR_boot <-
# sapply(seq(from = 3, to = niter * 3, by = 3),
#        function(x) KPT_survival_ASR_bootstrap_result[[x]])
# KPT_ASR_boot <- data.frame(ASR_boot = unname(KPT_ASR_boot), iter = 1:niter, population = "KPT")
#
# # Maio Kentish plover
```

```
# KPM_ASR_boot <-
# sapply(seq(from = 3, to = niter * 3, by = 3),
#        function(x) KPM_survival_ASR_bootstrap_result[[x]])
# KPM_ASR_boot <- data.frame(ASR_boot = unname(KPM_ASR_boot), iter = 1:niter, population = "KPM")
#
# # snowy plover
# SP_ASR_boot <-
# sapply(seq(from = 3, to = niter * 3, by = 3),
#        function(x) SP_survival_ASR_bootstrap_result[[x]])
# SP_ASR_boot <- data.frame(ASR_boot = unname(SP_ASR_boot), iter = 1:niter, population = "SP")
#
# ASR_boot_out <- rbind(SP_ASR_boot,
#                       KiP_ASR_boot,
#                       KPT_ASR_boot,
#                       KPM_ASR_boot,
#                       MP_ASR_boot,
#                       WfP_ASR_boot)
```

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

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

```
setwd("~/Dropbox/Luke/R_projects/Plover_ASR_Matrix_Modeling")

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

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

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

Visualizations of bootstrap results

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

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

```
sex_diff_survival <- function(survival_rates_boot, pop_name) {

  # subset the data to include only the population of interest
  survival_rates_boot2 <- dplyr::filter(survival_rates_boot, population == pop_name)

  # make an empty datarame to store the results
  sex_diff_surv_output <- data.frame(Adult = numeric(niter),
                                     Juvenile = 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_boot2[which(survival_rates_boot2$iter == i), 2][2] -
      survival_rates_boot2[which(survival_rates_boot2$iter == i), 2][1]
    Juvenile <-
      survival_rates_boot2[which(survival_rates_boot2$iter == i), 2][4] -
      survival_rates_boot2[which(survival_rates_boot2$iter == i), 2][3]

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

  # 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")
  sex_diff_surv_output$population <- pop_name

  # return the output
  sex_diff_surv_output
}
```

run the function on the bootstrap list from above

```
KIP_sex_diff_survival_output <- sex_diff_survival(survival_rates_boot, pop_name = "KIP")
WFP_sex_diff_survival_output <- sex_diff_survival(survival_rates_boot, pop_name = "WFP")
MP_sex_diff_survival_output <- sex_diff_survival(survival_rates_boot, pop_name = "MP")
KPT_sex_diff_survival_output <- sex_diff_survival(survival_rates_boot, pop_name = "KPT")
KPM_sex_diff_survival_output <- sex_diff_survival(survival_rates_boot, pop_name = "KPM")
SP_sex_diff_survival_output <- sex_diff_survival(survival_rates_boot, pop_name = "SP")

# stack the results into one dataframe and tidy column names
All_pops_sex_diff <- rbind(KIP_sex_diff_survival_output,
                           WFP_sex_diff_survival_output,
                           MP_sex_diff_survival_output,
                           KPT_sex_diff_survival_output,
                           KPM_sex_diff_survival_output,
                           SP_sex_diff_survival_output)

colnames(All_pops_sex_diff) <- c("Stage", "Difference", "Population")
```

```
# define the factor levels of the population variable so that the populations
# are in an order that reflects the ASR (male biased to female biased)
All_pops_sex_diff$Population <-
  factor(All_pops_sex_diff$Population ,
         levels = c("SP",
                    "KPT",
                    "KPM",
                    "MP",
                    "WFP",
                    "KIP"))
```

calculate some summary statistics

```
sex_diff_survival_summary <-
    All_pops_sex_diff %>%
    dplyr::group_by(Population, Stage) %>%
    dplyr::summarise(avg = mean(Difference),
                     median = median(Difference),
                     var = var(Difference))
sex_diff_survival_summary
#> Source: local data frame [12 x 5]
#> Groups: Population [?]
#>
#>    Population    Stage          avg      median          var
#>        <fctr>   <fctr>        <dbl>       <dbl>        <dbl>
#> 1          SP    Adult  0.015806634  0.01652850 0.0018048170
#> 2          SP Juvenile  0.071583451  0.07104185 0.0011055333
#> 3         KPT    Adult -0.014021108 -0.01489860 0.0007521734
#> 4         KPT Juvenile  0.179036816  0.18184110 0.0046802390
#> 5         KPM    Adult -0.003097242 -0.00246450 0.0003771494
#> 6         KPM Juvenile -0.019810875 -0.02020295 0.0006045011
#> 7          MP    Adult -0.026299420 -0.02472355 0.0026500049
#> 8          MP Juvenile -0.030316496 -0.02654495 0.0048493676
#> 9         WFP    Adult  0.012488142  0.01147375 0.0010597985
#> 10        WFP Juvenile -0.146879608 -0.14925290 0.0053930743
#> 11        KIP    Adult -0.040634251 -0.03833140 0.0025202083
#> 12        KIP Juvenile -0.132648250 -0.13643930 0.0039854666

sex_age_sample_size_survival <-
  juvenile_adult %>%
  dplyr::group_by(population, sex, age) %>%
  dplyr::summarise(n = n_distinct(bird_ID))
sex_age_sample_size_survival
#> Source: local data frame [24 x 4]
#> Groups: population, sex [?]
#>
#>    population    sex    age     n
#>        <fctr> <fctr> <fctr> <int>
#> 1         KIP      F      A   382
#> 2         KIP      F      J   274
#> 3         KIP      M      A   416
#> 4         KIP      M      J   286
#> 5         KPM      F      A   254
#> 6         KPM      F      J   377
```

```
#> 7         KPM      M      A   213
#> 8         KPM      M      J   383
#> 9         KPT      F      A   557
#> 10        KPT      F      J   310
#> # ... with 14 more rows

total_sample_size_survival <-
  juvenile_adult %>%
  dplyr::group_by(population) %>%
  dplyr::summarise(n = n_distinct(bird_ID))
total_sample_size_survival
#> # A tibble: 6 × 2
#>   population     n
#>       <fctr> <int>
#> 1        KIP  1358
#> 2        KPM  1227
#> 3        KPT  1664
#> 4         MP   245
#> 5         SP  1259
#> 6        WFP   366
```

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

```
cbPalette <- c("#BDBDBD", "#737373")
```

reorder the levels of the stage factors

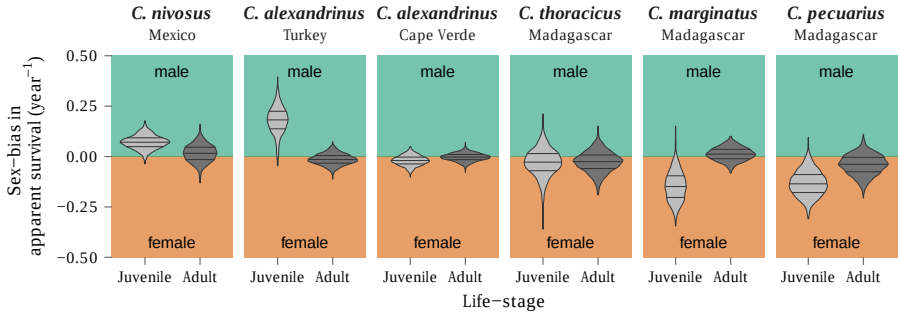
```
All_pops_sex_diff$Stage <-
  factor(All_pops_sex_diff$Stage, levels = c("Adult", "Juvenile"))

population_names <- c(
  'SP'="Snowy",
  'KPT'="Kentish (Tuzla)",
  'MP'="Madagascar",
  'KPM'="Kentish (Maio)",
  'WFP'="White-fronted",
  'KIP'="Kittlitz's"
)
```

Figure 1a: plot the sex-biases in survival across the Juvenile and Adult stages for each population

```
All_pops_sex_diff$Stage <-
  factor(All_pops_sex_diff$Stage, levels = c("Juvenile", "Adult"))

population_names <- c(
  'SP'="\nMexico",
  'KPT'="\nTurkey",
  'MP'="\nMadagascar",
  'KPM'="\nCape Verde",
  'WFP'="\nMadagascar",
  'KIP'="\nMadagascar"
)

species_names <- c(
  'SP'="C. nivosus\n",
  'KPT'="C. alexandrinus\n",
```

```
  'MP'="C. thoracicus\n",
  'KPM'="C. alexandrinus\n",
  'WFP'="C. marginatus\n",
  'KIP'="C. pecuarius\n"
)

Figure_1a_background <-
  ggplot(aes(y = Difference, x = Stage, fill = Stage), data = All_pops_sex_diff) +
  theme_bw() +
  annotate("rect", xmin=0, xmax=4, ymin=-0.5, ymax=0, alpha=0.6,
           fill=brewer.pal(8, "Dark2")[c(2)]) +
  annotate("rect", xmin=0, xmax=4, ymin=0, ymax=0.5, alpha=0.6,
           fill=brewer.pal(8, "Dark2")[c(1)]) +
  annotate("text", x = c(2), y = c(-0.45),
           label = c("female"), size = 2,
           vjust = c(0), hjust = c(0.5)) +
  annotate("text", x = c(2), y = c(0.45),
           label = c("male"), size = 2,
           vjust = c(1), hjust = c(0.5)) +
  facet_grid(. ~ Population, labeller = as_labeller(species_names)) +
  theme(text = element_text(family="Franklin Gothic Book", colour = "white"),
        legend.position = "none",
        axis.title.x = element_text(size=7, vjust=-0.1),
        axis.text.x  = element_text(size=6, angle = 0, hjust = 0.5),
        axis.title.y = element_text(size=7, hjust=0.5, vjust = 3.5),
        axis.text.y  = element_text(size=6),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_blank(),
        axis.ticks.x = element_line(size = 0.2, colour = "white"),
        axis.ticks.length = unit(0.1, "cm"),
        panel.border = element_blank(),
        plot.margin = unit(c(0.145,0.2,0.2,0.2), "cm"),
        panel.margin = unit(0.3, "lines"),
        strip.background = element_blank(),
        strip.text = element_text(size=7, face = "italic")) +
  scale_x_continuous(limits=c(0,4),breaks=c(0,1), labels=c("Juvenile", "Adult"),
                     expand = c(0, 0)) +
  scale_y_continuous(limits=c(-0.5,0.5), expand = c(0, 0)) +
  xlab("Life-stage") +
  ylab(expression(paste("             Sex-bias in\napparent survival (year" ^-1, ")",
                        sep = "")))

Figure_1a <-
  ggplot(aes(y = Difference, x = Stage, fill = Stage), data = All_pops_sex_diff) +
  theme_bw() +
  geom_violin(draw_quantiles = c(0.25, 0.5, 0.75), size = 0.15) +
  facet_grid(. ~ Population, labeller = as_labeller(population_names)) +
  theme(text = element_text(family="Franklin Gothic Book"),
        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=7, vjust=-0.1),
```

```
        axis.text.x  = element_text(size=6, angle = 0, hjust = 0.5),
        axis.title.y = element_text(size=7, hjust=0.5, vjust = 3.5),
        axis.text.y  = element_text(size=6),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_line(size = 0.2, colour = "grey40"),
        axis.ticks.length = unit(0.1, "cm"),
        axis.ticks.x = element_line(size = 0.2, colour = "grey40"),
        panel.border = element_blank(),
        panel.margin = unit(0.3, "lines"),
        strip.background = element_blank(),
        strip.text = element_text(size=6)) +
  scale_fill_manual(values = cbPalette) +
  scale_y_continuous(limits=c(-0.5,0.5), expand = c(0, 0)) +
  xlab("Life-stage") +
  ylab(expression(paste("             Sex-bias in\napparent survival (year" ^-1, ")",
                        sep = "")))

grid.newpage()
pushViewport( viewport( layout = grid.layout( 1 , 1 , widths = unit( 1 , "npc" ) ) ) )
print( Figure_1a_background + theme(legend.position="none") ,
       vp = viewport( layout.pos.row = 1 , layout.pos.col = 1 ) )
print( Figure_1a + theme(legend.position="none") ,
       vp = viewport( layout.pos.row = 1 , layout.pos.col = 1 ) )
```



Adult sex ratio distribution

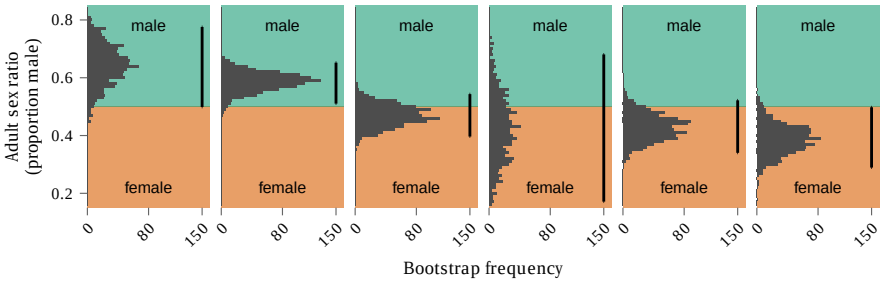
calculate the confidence interval, mean, and median of the ASR bootstraps

```
CI <- 0.95

ASR_boot_summary <-
  ASR_boot %>%
  dplyr::group_by(population) %>%
  dplyr::summarise(ucl = stats::quantile(ASR, (1 - CI)/2, na.rm = TRUE),
                   lcl = stats::quantile(ASR, 1 - (1 - CI)/2, na.rm = TRUE),
                   avg = mean(ASR),
                   med = median(ASR))

# define the factor levels of the population variable so that the populations
# are in an order that reflects the ASR (male biased to female biased)
ASR_boot$population <-
  factor(ASR_boot$population ,
         levels = c("SP",
```

```
                    "KPT",
                    "KPM",
                    "MP",
                    "WFP",
                    "KIP"))
```

Figure 1b: 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.

```
Figure_1b <-
  ggplot() +
  annotate("rect", xmin=0.15, xmax=0.5, ymin=0, ymax=160, alpha=0.6,
           fill= brewer.pal(8, "Dark2")[c(2)]) +
  annotate("rect", xmin=0.5, xmax=0.85, ymin=0, ymax=160, alpha=0.6,
           fill= brewer.pal(8, "Dark2")[c(1)]) +
  annotate("text", x = c(0.2), y = c(80),
           label = c("female"), size = 2,
           vjust = c(0), hjust = c(0.5)) +
  annotate("text", x = c(0.8), y = c(80),
           label = c("male"), size = 2,
           vjust = c(1), hjust = c(0.5)) +
  geom_histogram(binwidth = 0.01, data = ASR_boot, aes(x = ASR), fill = "grey30") +
  geom_errorbarh(data = ASR_boot_summary, aes(y = 150, x = lcl, xmin = lcl, xmax = ucl),
                 color = "black", size = 0.5, linetype = "solid") +
  coord_flip() +
  facet_grid(. ~ population) +
  theme_bw() +
  theme(text = element_text(family="Franklin Gothic Book"),
        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=7, vjust=-0.1),
        axis.text.x  = element_text(size=6, angle = 45, hjust = 1),
        axis.title.y = element_text(size=7, hjust=0.5, vjust = 3.5),
        axis.text.y  = element_text(size=6),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_line(size = 0.2, colour = "grey40"),
        axis.ticks.length = unit(0.1, "cm"),
        axis.ticks.x = element_line(size = 0.2, colour = "grey40"),
        panel.border = element_blank(),
        plot.margin = unit(c(0.2,0.2,0.2,0.22), "cm"),
        panel.margin = unit(0.3, "lines"),
        strip.background = element_blank(),
        strip.text = element_blank()) +
  ylab("Bootstrap frequency") +
  xlab("Adult sex ratio\n(proportion male)") +
  scale_x_continuous(limits = c(0.15, 0.85), expand = c(0, 0)) +
  scale_y_continuous(limits = c(0, 160), expand = c(0, 0), breaks=c(0, 80, 150))
Figure_1b
```



AIC model selection summary (panels in Supplementary Material Figure S5)

To illustrate the mark-recapture model selection going on during the bootstrap, we summarized AIC statistics for each model included in the survival analysis and visualized with ranked boxplots (Figure S5)

First, wrangle the bootstrap AIC table output

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

# summarize the average AIC stats for each candidate model across all 1000 iterations
juv_ad_AIC_tables_summary <-
  juv_ad_AIC_tables %>%
  dplyr::group_by(population,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
juv_ad_AIC_tables_summary <- dplyr::arrange(juv_ad_AIC_tables_summary, population, avg_Delta)
juv_ad_AIC_tables_summary$model_number <- as.numeric(juv_ad_AIC_tables_summary$model)

# merge the two datasets for plotting
juv_ad_AIC_tables <-
  dplyr::left_join(juv_ad_AIC_tables_summary, juv_ad_AIC_tables, by = c("population", "model_number"))

juv_ad_AIC_tables$p <-
  str_replace(juv_ad_AIC_tables$p, pattern = "time", replacement = "year")

juv_ad_AIC_tables_summary$model <-
  str_replace(juv_ad_AIC_tables_summary$model, pattern = "time", replacement = "year")

juv_ad_AIC_tables$p <-
  str_replace(juv_ad_AIC_tables$p, pattern = "Quadratic", replacement = "Time^2")

juv_ad_AIC_tables_summary$model <-
  str_replace(juv_ad_AIC_tables_summary$model, pattern = "Quadratic", replacement = "Time^2")

# extract the model structure explaining resighting probability
juv_ad_AIC_tables$p <-
  factor(juv_ad_AIC_tables$p,
         levels = c(str_sub(as.character(juv_ad_AIC_tables_summary$model),
                            start = 18, end = str_length(juv_ad_AIC_tables_summary$model)-1)))

juv_ad_AIC_tables$population <-
```

```
  factor(juv_ad_AIC_tables$population,
         levels = c("SP",
                    "KPT",
                    "KPM",
                    "MP",
                    "WFP",
                    "KIP"))

roles <- function(x) sub("[^_]*_","",x )
```

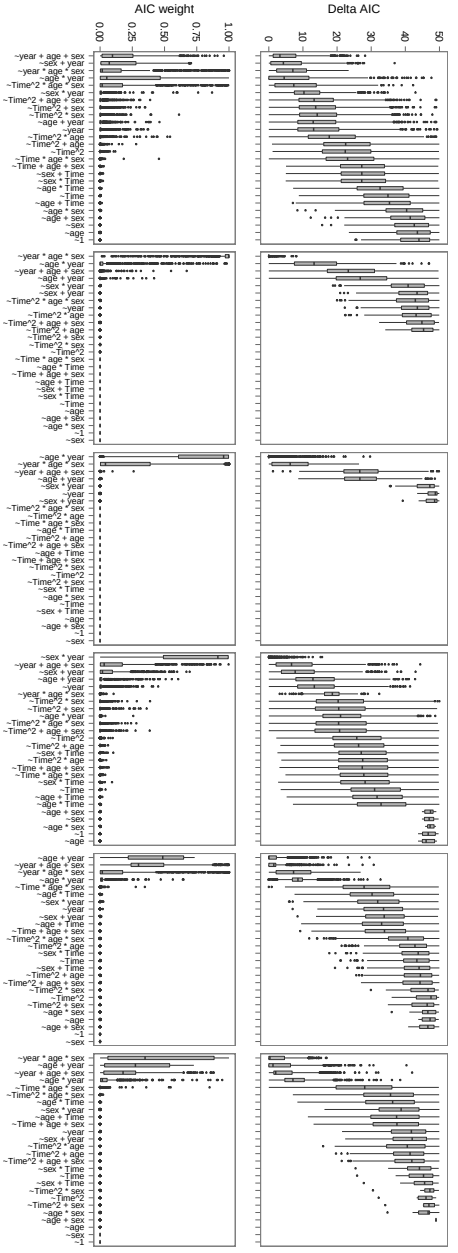
plot the overall model ranks of the juvenile and adult survival anlaysis based on Delta AIC. We specified that the model axis is ranked according to their AIC statistics.

```
Figure_S5_Delta_AIC <-
  ggplot(cbind(as.data.frame(juv_ad_AIC_tables),
               V4=paste(juv_ad_AIC_tables$population,juv_ad_AIC_tables$p,sep="_")),
         aes(x=reorder(V4,DeltaAICc), y=DeltaAICc) ) +
  theme_bw() +
  geom_boxplot(width = 0.6, fill = "grey70", outlier.size = 0.000, size = 0.2) +
  facet_grid(. ~ population, labeller = as_labeller(species_names), scales="free_x") +
  theme(
    legend.position = "none",
    axis.title.x = element_blank(),
    axis.text.x  = element_blank(),
    axis.title.y = element_text(size=7, margin = margin(0, 10, 0, 0)),
    axis.text.y  = element_text(size=6),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.ticks.y = element_line(size = 0.2, colour = "grey40"),
    axis.ticks.length = unit(0.1, "cm"),
    axis.ticks.x = element_line(size = 0.2, colour = "grey40"),
    plot.margin = unit(c(0.2,0.2,0.2,0.2), "cm"),
    panel.margin = unit(0.3, "lines"),
    strip.background = element_blank(),
    strip.text = element_text(size=6)) +
  scale_x_discrete(labels=roles) +
  scale_y_continuous(limits=c(0,50)) +
  xlab("Model") +
  ylab("Delta AIC")
```

plot the overall model ranks of the juvenile and adult survival anlaysis based on AIC weight. We specified that the model axis is ranked according to their AIC statistics.

```
Figure_S5_AIC_weight <-
  ggplot(cbind(as.data.frame(juv_ad_AIC_tables),
               V4=paste(juv_ad_AIC_tables$population,juv_ad_AIC_tables$p,sep="_")),
         aes(x=reorder(V4,DeltaAICc), y=weight) ) +
  theme_bw() +
  geom_boxplot(width = 0.6, fill = "grey70", outlier.size = 0.000, size = 0.2) +
  facet_grid(. ~ population, scales="free_x") +
  theme(
    legend.position = "none",
    axis.title.x = element_blank(),
    axis.text.x  = element_text(size=5, angle = 90, hjust = 1, vjust = 0.5),
    axis.title.y = element_text(size=7, margin = margin(0, 5, 0, 0)),
```

```
    axis.text.y  = element_text(size=6),
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank(),
    axis.ticks.y = element_line(size = 0.2, colour = "grey40"),
    axis.ticks.length = unit(0.1, "cm"),
    axis.ticks.x = element_line(size = 0.2, colour = "grey40"),
    plot.margin = unit(c(0.2,0.2,0.2,0.2), "cm"),
    panel.margin = unit(0.3, "lines"),
    strip.background = element_blank(),
    strip.text = element_blank(),
    plot.background = element_rect(fill = "transparent",colour = NA)) +
  scale_x_discrete(labels=roles) +
  scale_y_continuous(limits=c(0,1)) +
  xlab("Model") +
  ylab("AIC weight")

grid.newpage()
pushViewport(viewport(layout = grid.layout(4, 1)))
vplayout <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)
print(Figure_S5_Delta_AIC, vp = vplayout(1:2, 1))
print(Figure_S5_AIC_weight, vp = vplayout(3:4, 1))
```



Life table response experiment on ASR

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

The following two functions need to be specified first:

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

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

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

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

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

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

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

    # dataframe to store the perturbation results
    ASR_pert_results <-
      data.frame(parameter = c("F_Juv_survl", "F_Adt_survl",
                               "M_Juv_survl", "M_Adt_survl",
                               "h", "k", "HSR"),
                 sensitivities = numeric(7),
                 elasticities = numeric(7))

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

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

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

```
  # perturbation of vital rates survival rates
  for (g in 1:n) # pick a column (i.e., a variable)
  {
    vr2 <- vr # reset the vital rates to the original
    for (i in 1:length(vr_nums)) # pick a perturbation level
    {
      vr2[[g]] <- vr_nums[i] # specify the vital rate with the new perturbation level
      A <- matrix(sapply(matrix_str, eval, vr2, NULL),
                  nrow = sqrt(length(matrix_str)), byrow=TRUE,
                  dimnames = list(stages, stages)) # build the matrix with the new value
      # reset the starting stage distribution for simulation (all with 10 individuals)
      m <- rep(10, no_stages)
      for (j in 1:niter) { # project the matrix through t iteration
        # stage distribution at time t
        stage[,j] <- m
        # population size at time t
        pop[j] <- sum(m)
        # number of male adults at time t
        M2 <- stage[4, j]
        # number of female adults at time t
        F2 <- stage[2, j]
        # Female freq-dep fecundity of Female chicks
        A[1,no_stages/2]        <- ((k*M2)/(M2+(F2/h)))*HSR
        # Female freq-dep fecundity of Male chicks
        A[(no_stages/4)*3,no_stages/2]  <- ((k*M2)/(M2+(F2/h)))*HSR
        # Male freq-dep fecundity of Female chicks
        A[1,no_stages]          <- ((k*F2)/(M2+(F2/h)))*HSR
        # Male freq-dep fecundity of Male chicks
        A[(no_stages/4)*3,no_stages]    <- ((k*F2)/(M2+(F2/h)))*HSR
        # define the new n (i.e., new stage distribution at time t)
        m <- A %*% m
      }
      # define rownames of stage matrix
      rownames(stage) <- rownames(A)
      # define colnames of stage matrix
      colnames(stage) <- 0:(niter - 1)
      # calculate the proportional stable stage distribution
      stage <- apply(stage, 2, function(x) x/sum(x))
      # define stable stage as the last stage
      stable.stage <- stage[, niter]
      # calc ASR as the proportion of the adult stable stage class that is male
      vr_pert_ASR[i, g] <- stable.stage[no_stages]/(stable.stage[no_stages/2] +
                                                    stable.stage[no_stages])
    }
    # get the spline function of ASR
    spl_ASR <- smooth.spline(vr_pert_ASR[,g] ~ rownames(vr_pert_ASR))
    # estimate the slope of the tangent of the spline at the vital rate
    ASR_pert_results[g, 2] <- predict(spl_ASR, x=vr[[g]], deriv=1)$y
    # re-scale sensitivity into elasticity
    ASR_pert_results[g, 3] <- vr[[g]]/ASR * ASR_pert_results[g, 2]
  }
  # perturbation of the h index parameter
  for (i in 1:length(h_nums)) # pick a perturbation level
```

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

# perturbation of k parameter
for (i in 1:length(k_nums)) # pick a perturbation level
{
  A <- matrix(sapply(matrix_str, eval, vr, NULL),
              nrow = sqrt(length(matrix_str)), byrow=TRUE,
              dimnames = list(stages, stages)) # build the matrix with the new value
  # reset the starting stage distribution for simulation (all with 10 individuals)
  m <- rep(10, no_stages)
  for (j in 1:niter) { # project the matrix through t iteration
```

```
    # stage distribution at time t
    stage[,j] <- m
    # population size at time t
    pop[j] <- sum(m)
    # number of male adults at time t
    M2 <- stage[4, j]
    # number of female adults at time t
    F2 <- stage[2, j]
    # Female freq-dep fecundity of Female chicks
    A[1,no_stages/2]        <- ((k_nums[i]*M2)/(M2+(F2/h)))*HSR
    # Female freq-dep fecundity of Male chicks
    A[(no_stages/4)*3,no_stages/2] <- ((k_nums[i]*M2)/(M2+(F2/h)))*HSR
    # Male freq-dep fecundity of Female chicks
    A[1,no_stages]          <- ((k_nums[i]*F2)/(M2+(F2/h)))*HSR
    # Male freq-dep fecundity of Male chicks
    A[(no_stages/4)*3,no_stages] <- ((k_nums[i]*F2)/(M2+(F2/h)))*HSR
    # define the new n (i.e., new stage distribution at time t)
    m <- A %*% m
  }
  # define rownames of stage matrix
  rownames(stage) <- rownames(A)
  # define colnames of stage matrix
  colnames(stage) <- 0:(niter - 1)
  # calculate the proportional stable stage distribution
  stage <- apply(stage, 2, function(x) x/sum(x))
  # define stable stage as the last stage
  stable.stage <- stage[, niter]
  # calc ASR as the proportion of the adult stable stage class that is male
  k_pert_ASR[i,] <- stable.stage[no_stages]/(stable.stage[no_stages/2] +
                                               stable.stage[no_stages])
}
# get the spline function of ASR
spl_ASR <- smooth.spline(k_pert_ASR[,1] ~ rownames(k_pert_ASR))
# estimate the slope of the tangent of the spline at the vital rate
ASR_pert_results[n+2, 2] <- predict(spl_ASR, x=k, deriv=1)$y
# re-scale sensitivity into elasticity
ASR_pert_results[n+2, 3] <- k/ASR * ASR_pert_results[n+2, 2]

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

```
      # number of female adults at time t
      F2 <- stage[2, j]
      # Female freq-dep fecundity of Female chicks
      A[1,no_stages/2]        <- ((k*M2)/(M2+(F2/h)))*vr_nums[i]
      # Female freq-dep fecundity of Male chicks
      A[(no_stages/4)*3,no_stages/2] <- ((k*M2)/(M2+(F2/h)))*vr_nums[i]
      # Male freq-dep fecundity of Female chicks
      A[1,no_stages]          <- ((k*F2)/(M2+(F2/h)))*vr_nums[i]
      # Male freq-dep fecundity of Male chicks
      A[(no_stages/4)*3,no_stages] <- ((k*F2)/(M2+(F2/h)))*vr_nums[i]
      # define the new n (i.e., new stage distribution at time t)
      m <- A %*% m
    }
    # define rownames of stage matrix
    rownames(stage) <- rownames(A)
    # define colnames of stage matrix
    colnames(stage) <- 0:(niter - 1)
    # calculate the proportional stable stage distribution
    stage <- apply(stage, 2, function(x) x/sum(x))
    # define stable stage as the last stage
    stable.stage <- stage[, niter]
    # calc ASR as the proportion of the adult stable stage class that is male
    HSR_pert_ASR[i,] <- stable.stage[no_stages]/(stable.stage[no_stages/2] +
                                                   stable.stage[no_stages])
  }
  # get the spline function of ASR
  spl_ASR <- smooth.spline(HSR_pert_ASR[,1] ~ rownames(HSR_pert_ASR))
  # estimate the slope of the tangent of the spline at the vital rate
  ASR_pert_results[n+3, 2] <- predict(spl_ASR, x=HSR, deriv=1)$y
  # re-scale sensitivity into elasticity
  ASR_pert_results[n+3, 3] <- HSR/ASR * ASR_pert_results[n+3, 2]

  result <- list(ASR_pert_results = ASR_pert_results)
}
```

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

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

    # make empty dataframes to stroe LTRE results for ASR
    LTRE_ASR <-
      data.frame(parameter = c("Juvenile survival",
                               "Adult survival", "Hatching sex ratio",
                               "Mating system"),
                 contribution = numeric(4))

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

```
              ifelse(i == 3, ((1-vital_rates[[7]]) - vital_rates[[7]]) *
                       Mprime_sens$ASR_pert_results$sensitivities[7],
                     (1 - vital_rates[[5]]) * Mprime_sens$ASR_pert_results$sensitivities[5]))
    }

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

    LTRE_ASR$model <- "ASR"
    LTRE_ASR$population <- pop_name
    LTRE_df <- LTRE_ASR
    LTRE_df
  }
```

define the iterations variable as a factor

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

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

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

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

```
KIP_VR_treat <-
  list(F_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KIP")[2,4],
       F_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KIP")[1,4],
       M_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KIP")[4,4],
       M_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KIP")[3,4],
       # Define h (harem size) and k (clutch size)
       h = KIP_h,
       k = 3,
       # Define primary sex ratio
       HSR = HSR_KIP)

KPT_VR_treat <-
  list(F_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KPT")[2,4],
       F_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KPT")[1,4],
       M_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KPT")[4,4],
       M_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KPT")[3,4],
       # Define h (harem size) and k (clutch size)
```

```
       h = KPT_h,
       k = 3,
       # Define primary sex ratio
       HSR = HSR_KPT)

KPM_VR_treat <-
  list(F_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KPM")[2,4],
       F_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KPM")[1,4],
       M_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KPM")[4,4],
       M_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "KPM")[3,4],
       # Define h (harem size) and k (clutch size)
       h = KPM_h,
       k = 3,
       # Define primary sex ratio
       HSR = HSR_KPM)

MP_VR_treat <-
  list(F_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "MP")[2,4],
       F_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "MP")[1,4],
       M_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "MP")[4,4],
       M_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "MP")[3,4],
       # Define h (harem size) and k (clutch size)
       h = MP_h,
       k = 2,
       # Define primary sex ratio
       HSR = HSR_MP)

WFP_VR_treat <-
  list(F_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "WFP")[2,4],
       F_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "WFP")[1,4],
       M_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "WFP")[4,4],
       M_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "WFP")[3,4],
       # Define h (harem size) and k (clutch size)
       h = WFP_h,
       k = 3,
       # Define primary sex ratio
       HSR = HSR_WFP)

SP_VR_treat <-
  list(F_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "SP")[2,4],
```

```
       F_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "SP")[1,4],
       M_Juv_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "SP")[4,4],
       M_Adt_survl = dplyr::filter(survival_rates_boot_summary,
                                   population == "SP")[3,4],
       # Define h (harem size) and k (clutch size)
       h = SP_h,
       k = 3,
       # Define primary sex ratio
       HSR = HSR_SP)
```

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

```
KIP_VR_mprime <- list(F_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "KIP")[2,4] +
                                     dplyr::filter(survival_rates_boot_summary,
                                                   population == "KIP")[4,4])/2,
                      F_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "KIP")[1,4] +
                                     dplyr::filter(survival_rates_boot_summary,
                                                   population == "KIP")[3,4])/2,
                      M_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "KIP")[4,4] +
                                     dplyr::filter(survival_rates_boot_summary,
                                                   population == "KIP")[4,4])/2,
                      M_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "KIP")[3,4] +
                                     dplyr::filter(survival_rates_boot_summary,
                                                   population == "KIP")[3,4])/2,
                      # Define h (harem size) and k (clutch size)
                      h = (KIP_h+1)/2,
                      k = 2,
                      # Define primary sex ratio
                      HSR = (HSR_KIP+0.5)/2)

KPT_VR_mprime <- list(F_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "KPT")[2,4] +
                                     dplyr::filter(survival_rates_boot_summary,
                                                   population == "KPT")[4,4])/2,
                      F_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "KPT")[1,4] +
                                     dplyr::filter(survival_rates_boot_summary,
                                                   population == "KPT")[3,4])/2,
                      M_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "KPT")[4,4] +
                                     dplyr::filter(survival_rates_boot_summary,
                                                   population == "KPT")[4,4])/2,
                      M_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "KPT")[3,4] +
```

```
                                    dplyr::filter(survival_rates_boot_summary,
                                                  population == "KPT")[3,4])/2,
                    # Define h (harem size) and k (clutch size)
                    h = (KPT_h+1)/2,
                    k = 3,
                    # Define primary sex ratio
                    HSR = (HSR_KPT+0.5)/2)

KPM_VR_mprime <- list(F_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "KPM")[2,4] +
                                    dplyr::filter(survival_rates_boot_summary,
                                                  population == "KPM")[4,4])/2,
                    F_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                 population == "KPM")[1,4] +
                                   dplyr::filter(survival_rates_boot_summary,
                                                 population == "KPM")[3,4])/2,
                    M_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                 population == "KPM")[4,4] +
                                   dplyr::filter(survival_rates_boot_summary,
                                                 population == "KPM")[4,4])/2,
                    M_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                 population == "KPM")[3,4] +
                                   dplyr::filter(survival_rates_boot_summary,
                                                 population == "KPM")[3,4])/2,
                    # Define h (harem size) and k (clutch size)
                    h = (KPM_h+1)/2,
                    k = 3,
                    # Define primary sex ratio
                    HSR = (HSR_KPM+0.5)/2)

MP_VR_mprime <- list(F_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                  population == "MP")[2,4] +
                                    dplyr::filter(survival_rates_boot_summary,
                                                  population == "MP")[4,4])/2,
                    F_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                 population == "MP")[1,4] +
                                   dplyr::filter(survival_rates_boot_summary,
                                                 population == "MP")[3,4])/2,
                    M_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                 population == "MP")[4,4] +
                                   dplyr::filter(survival_rates_boot_summary,
                                                 population == "MP")[4,4])/2,
                    M_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                 population == "MP")[3,4] +
                                   dplyr::filter(survival_rates_boot_summary,
                                                 population == "MP")[3,4])/2,
                    # Define h (harem size) and k (clutch size)
                    h = (MP_h+1)/2,
                    k = 2,
                    # Define primary sex ratio
                    HSR = (HSR_MP+0.5)/2)

WFP_VR_mprime <- list(F_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
```

```
                                                  population == "WFP")[2,4] +
                                  dplyr::filter(survival_rates_boot_summary,
                                                  population == "WFP")[4,4])/2,
                  F_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                population == "WFP")[1,4] +
                                  dplyr::filter(survival_rates_boot_summary,
                                                  population == "WFP")[3,4])/2,
                  M_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                population == "WFP")[4,4] +
                                  dplyr::filter(survival_rates_boot_summary,
                                                  population == "WFP")[4,4])/2,
                  M_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                population == "WFP")[3,4] +
                                  dplyr::filter(survival_rates_boot_summary,
                                                  population == "WFP")[3,4])/2,
                  # Define h (harem size) and k (clutch size)
                  h = (WFP_h+1)/2,
                  k = 3,
                  # Define primary sex ratio
                  HSR = (HSR_WFP+0.5)/2)

SP_VR_mprime <- list(F_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                   population == "SP")[2,4] +
                                  dplyr::filter(survival_rates_boot_summary,
                                                  population == "SP")[4,4])/2,
                  F_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                population == "SP")[1,4] +
                                  dplyr::filter(survival_rates_boot_summary,
                                                  population == "SP")[3,4])/2,
                  M_Juv_survl = (dplyr::filter(survival_rates_boot_summary,
                                                population == "SP")[4,4] +
                                  dplyr::filter(survival_rates_boot_summary,
                                                  population == "SP")[4,4])/2,
                  M_Adt_survl = (dplyr::filter(survival_rates_boot_summary,
                                                population == "SP")[3,4] +
                                  dplyr::filter(survival_rates_boot_summary,
                                                  population == "SP")[3,4])/2,
                  # Define h (harem size) and k (clutch size)
                  h = (SP_h+1)/2,
                  k = 3,
                  # Define primary sex ratio
                  HSR = (HSR_SP+0.5)/2)
```

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

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

                              # second row of matrix
                              F_Juv_survl, F_Adt_survl, 0, 0,

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

```
                                # fourth row of matrix
                                0, 0, M_Juv_survl, M_Adt_survl

                                )
```

build the treatment matrix

```
KIP_treatment_matrix <- plover_matrix(KIP_VR_treat)
KPT_treatment_matrix <- plover_matrix(KPT_VR_treat)
KPM_treatment_matrix <- plover_matrix(KPM_VR_treat)
MP_treatment_matrix <- plover_matrix(MP_VR_treat)
WFP_treatment_matrix <- plover_matrix(WFP_VR_treat)
SP_treatment_matrix <- plover_matrix(SP_VR_treat)
```

build the M-prime matrix

```
KIP_M_prime_matrix <- plover_matrix(KIP_VR_mprime)
KPT_M_prime_matrix <- plover_matrix(KPT_VR_mprime)
KPM_M_prime_matrix <- plover_matrix(KPM_VR_mprime)
MP_M_prime_matrix <- plover_matrix(MP_VR_mprime)
WFP_M_prime_matrix <- plover_matrix(WFP_VR_mprime)
SP_M_prime_matrix <- plover_matrix(SP_VR_mprime)
```

determine the ASR at the stable stage distribution

```
KIP_treatment_ASR_analysis <-
  matrix_ASR(M = KIP_treatment_matrix, h = KIP_VR_treat$h, HSR = KIP_VR_treat$HSR,
             iterations = 1000)
KIP_ASR_treat <- KIP_treatment_ASR_analysis$ASR
KIP_ASR_treat
#>     M_Adt
#> 0.3969559

KIP_M_prime_ASR_analysis <-
  matrix_ASR(M = KIP_M_prime_matrix, h = 1, HSR = KIP_VR_mprime$HSR,
             iterations = 1000)
KIP_ASR_mprime <- KIP_M_prime_ASR_analysis$ASR
KIP_ASR_mprime
#>     M_Adt
#> 0.4430878

KPT_treatment_ASR_analysis <-
  matrix_ASR(M = KPT_treatment_matrix, h = KPT_VR_treat$h, HSR = KPT_VR_treat$HSR,
             iterations = 1000)
KPT_ASR_treat <- KPT_treatment_ASR_analysis$ASR
KPT_ASR_treat
#>    M_Adt
#> 0.586768

KPT_M_prime_ASR_analysis <-
  matrix_ASR(M = KPT_M_prime_matrix, h = 1, HSR = KPT_VR_mprime$HSR,
             iterations = 1000)
KPT_ASR_mprime <- KPT_M_prime_ASR_analysis$ASR
KPT_ASR_mprime
#>    M_Adt
```

```
#> 0.539473

KPM_treatment_ASR_analysis <-
  matrix_ASR(M = KPM_treatment_matrix, h = KPM_VR_treat$h, HSR = KPM_VR_treat$HSR,
             iterations = 1000)
KPM_ASR_treat <- KPM_treatment_ASR_analysis$ASR
KPM_ASR_treat
#>     M_Adt
#> 0.4686537

KPM_M_prime_ASR_analysis <-
  matrix_ASR(M = KPM_M_prime_matrix, h = 1, HSR = KPM_VR_mprime$HSR,
             iterations = 1000)
KPM_ASR_mprime <- KPM_M_prime_ASR_analysis$ASR
KPM_ASR_mprime
#>    M_Adt
#> 0.483884

MP_treatment_ASR_analysis <-
  matrix_ASR(M = MP_treatment_matrix, h = MP_VR_treat$h, HSR = MP_VR_treat$HSR,
             iterations = 1000)
MP_ASR_treat <- MP_treatment_ASR_analysis$ASR
MP_ASR_treat
#>     M_Adt
#> 0.4315276

MP_M_prime_ASR_analysis <-
  matrix_ASR(M = MP_M_prime_matrix, h = 1, HSR = MP_VR_mprime$HSR,
             iterations = 1000)
MP_ASR_mprime <- MP_M_prime_ASR_analysis$ASR
MP_ASR_mprime
#>     M_Adt
#> 0.4631169

WFP_treatment_ASR_analysis <-
  matrix_ASR(M = WFP_treatment_matrix, h = WFP_VR_treat$h, HSR = WFP_VR_treat$HSR,
             iterations = 1000)
WFP_ASR_treat <- WFP_treatment_ASR_analysis$ASR
WFP_ASR_treat
#>     M_Adt
#> 0.4261912

WFP_M_prime_ASR_analysis <-
  matrix_ASR(M = WFP_M_prime_matrix, h = 1, HSR = WFP_VR_mprime$HSR,
             iterations = 1000)
WFP_ASR_mprime <- WFP_M_prime_ASR_analysis$ASR
WFP_ASR_mprime
#>     M_Adt
#> 0.4600911

SP_treatment_ASR_analysis <-
  matrix_ASR(M = SP_treatment_matrix, h = SP_VR_treat$h, HSR = SP_VR_treat$HSR,
             iterations = 1000)
```

```
SP_ASR_treat <- SP_treatment_ASR_analysis$ASR
SP_ASR_treat
#>    M_Adt
#> 0.608363

SP_M_prime_ASR_analysis <-
  matrix_ASR(M = SP_M_prime_matrix, h = 1, HSR = SP_VR_mprime$HSR,
             iterations = 1000)
SP_ASR_mprime <- SP_M_prime_ASR_analysis$ASR
SP_ASR_mprime
#>     M_Adt
#> 0.5498809
```

conduct a sensitivity analysis on the treatment matrix

```
KIP_treat_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = KIP_VR_treat,
                       matrix_str = matrix_structure,
                       h = KIP_VR_treat$h,
                       k = KIP_VR_treat$k,
                       HSR = KIP_VR_treat$HSR,
                       niter = 1000,
                       ASR = KIP_ASR_treat)

KPT_treat_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = KPT_VR_treat,
                       matrix_str = matrix_structure,
                       h = KPT_VR_treat$h,
                       k = KPT_VR_treat$k,
                       HSR = KPT_VR_treat$HSR,
                       niter = 1000,
                       ASR = KPT_ASR_treat)

KPM_treat_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = KPM_VR_treat,
                       matrix_str = matrix_structure,
                       h = KPM_VR_treat$h,
                       k = KPM_VR_treat$k,
                       HSR = KPM_VR_treat$HSR,
                       niter = 1000,
                       ASR = KPM_ASR_treat)

MP_treat_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = MP_VR_treat,
                       matrix_str = matrix_structure,
                       h = MP_VR_treat$h,
                       k = MP_VR_treat$k,
                       HSR = MP_VR_treat$HSR,
                       niter = 1000,
                       ASR = MP_ASR_treat)

WFP_treat_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = WFP_VR_treat,
                       matrix_str = matrix_structure,
```

```
                        h = WFP_VR_treat$h,
                        k = WFP_VR_treat$k,
                        HSR = WFP_VR_treat$HSR,
                        niter = 1000,
                        ASR = WFP_ASR_treat)

SP_treat_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = SP_VR_treat,
                        matrix_str = matrix_structure,
                        h = SP_VR_treat$h,
                        k = SP_VR_treat$k,
                        HSR = SP_VR_treat$HSR,
                        niter = 1000,
                        ASR = SP_ASR_treat)
```

conduct a sensitivity analysis on the M-Prime matrix

```
KIP_Mprime_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = KIP_VR_mprime,
                        matrix_str = matrix_structure,
                        h = KIP_VR_mprime$h,
                        k = KIP_VR_mprime$k,
                        HSR = KIP_VR_mprime$HSR,
                        niter = 1000,
                        ASR = KIP_ASR_mprime)

KPT_Mprime_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = KPT_VR_mprime,
                        matrix_str = matrix_structure,
                        h = KPT_VR_mprime$h,
                        k = KPT_VR_mprime$k,
                        HSR = KPT_VR_mprime$HSR,
                        niter = 1000,
                        ASR = KPT_ASR_mprime)

KPM_Mprime_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = KPM_VR_mprime,
                        matrix_str = matrix_structure,
                        h = KPM_VR_mprime$h,
                        k = KPM_VR_mprime$k,
                        HSR = KPM_VR_mprime$HSR,
                        niter = 1000,
                        ASR = KPM_ASR_mprime)

MP_Mprime_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = MP_VR_mprime,
                        matrix_str = matrix_structure,
                        h = MP_VR_mprime$h,
                        k = MP_VR_mprime$k,
                        HSR = MP_VR_mprime$HSR,
                        niter = 1000,
                        ASR = MP_ASR_mprime)

WFP_Mprime_sensitivity_analysis <-
```

```
  sensitivity_analysis(vital_rates = WFP_VR_mprime,
                       matrix_str = matrix_structure,
                       h = WFP_VR_mprime$h,
                       k = WFP_VR_mprime$k,
                       HSR = WFP_VR_mprime$HSR,
                       niter = 1000,
                       ASR = WFP_ASR_mprime)

SP_Mprime_sensitivity_analysis <-
  sensitivity_analysis(vital_rates = SP_VR_mprime,
                       matrix_str = matrix_structure,
                       h = SP_VR_mprime$h,
                       k = SP_VR_mprime$k,
                       HSR = SP_VR_mprime$HSR,
                       niter = 1000,
                       ASR = SP_ASR_mprime)
```

conduct the LTRE comparing the two matrices

```
KIP_LTRE_plover <-
  LTRE_analysis(Mprime_sens = KIP_Mprime_sensitivity_analysis,
                matrix_str = matrix_str,
                vital_rates = KIP_VR_treat,
                pop_name = "KIP")

KPT_LTRE_plover <-
  LTRE_analysis(Mprime_sens = KPT_Mprime_sensitivity_analysis,
                matrix_str = matrix_str,
                vital_rates = KPT_VR_treat,
                pop_name = "KPT")

KPM_LTRE_plover <-
  LTRE_analysis(Mprime_sens = KPM_Mprime_sensitivity_analysis,
                matrix_str = matrix_str,
                vital_rates = KPM_VR_treat,
                pop_name = "KPM")

MP_LTRE_plover <-
  LTRE_analysis(Mprime_sens = MP_Mprime_sensitivity_analysis,
                matrix_str = matrix_str,
                vital_rates = MP_VR_treat,
                pop_name = "MP")

WFP_LTRE_plover <-
  LTRE_analysis(Mprime_sens = WFP_Mprime_sensitivity_analysis,
                matrix_str = matrix_str,
                vital_rates = WFP_VR_treat,
                pop_name = "WFP")

SP_LTRE_plover <-
  LTRE_analysis(Mprime_sens = SP_Mprime_sensitivity_analysis,
                matrix_str = matrix_str,
                vital_rates = SP_VR_treat,
                pop_name = "SP")
```

```
LTRE_plover <- rbind(KIP_LTRE_plover, KPT_LTRE_plover, KPM_LTRE_plover,
                     MP_LTRE_plover, WFP_LTRE_plover, SP_LTRE_plover)

# define the factor levels of the population variable so that the populations
# are in an order that reflects the ASR (male biased to female biased)
LTRE_plover$population <-
  factor(LTRE_plover$population ,
         levels = c("SP",
                    "KPT",
                    "KPM",
                    "MP",
                    "WFP",
                    "KIP"))
```

custom color palette for the plotting of Juvenile and Adult stats

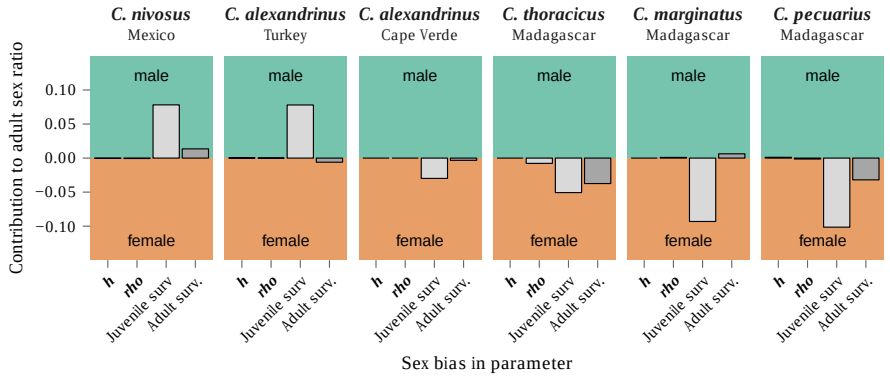
```
cbPalette <- c("#A6A6A6", "#D9D9D9", "#D9D9D9", "#A6A6A6")
```

Figure S2: plot the comparative LTRE results

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

population_names <- c(
  'SP'="\nMexico",
  'KPT'="\nTurkey",
  'KPM'="\nCape Verde",
  'MP'="\nMadagascar",
  'WFP'="\nMadagascar",
  'KIP'="\nMadagascar"
)

species_names <- c(
  'SP'="C. nivosus\n",
  'KPT'="C. alexandrinus\n",
  'KPM'="C. alexandrinus\n",
  'MP'="C. thoracicus\n",
  'WFP'="C. marginatus\n",
  'KIP'="C. pecuarius\n"
)

Figure_S2_background <-
  ggplot2::ggplot(data = LTRE_plover[which(LTRE_plover$model == "ASR"), ],
                  aes(x = parameter, y = contribution, fill = parameter)) +
  theme_bw() +
  annotate("rect", xmin=0, xmax=4, ymin=-0.15, ymax=0, alpha=0.6,
           fill=brewer.pal(8, "Dark2")[c(2)]) +
  annotate("rect", xmin=0, xmax=4, ymin=0, ymax=0.15, alpha=0.6,
           fill=brewer.pal(8, "Dark2")[c(1)]) +
  annotate("text", x = c(2), y = c(-0.13),
           label = c("female"), size = 2,
           vjust = c(0), hjust = c(0.5)) +
  annotate("text", x = c(2), y = c(0.13),
           label = c("male"), size = 2,
```

```
          vjust = c(1), hjust = c(0.5)) +
  facet_grid(. ~ population, labeller = as_labeller(species_names)) +
  theme(text = element_text(family="Franklin Gothic Book",
                            colour = "white"),
        legend.position = "none",
        axis.title.x = element_text(size=7, vjust=-0.1),
        axis.text.x  = element_text(size=6, angle = 45, hjust = 1),
        axis.title.y = element_text(size=7, hjust=0.5, vjust = 3.5),
        axis.text.y  = element_text(size=6),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_blank(),
        axis.ticks.x = element_line(size = 0.2, colour = "white"),
        axis.ticks.length = unit(0.1, "cm"),
        panel.border = element_blank(),
        plot.margin = unit(c(0.15,0.2,1.32,0.21), "cm"),
        panel.margin = unit(0.3, "lines"),
        strip.background = element_blank(),
        strip.text = element_text(size=7, face = "italic")) +
  scale_x_discrete(labels = c("Adult survival" = "Adult surv.",
                              "Juvenile survival" = "Juvenile surv",
                              "Hatching sex ratio" = expression(italic(rho)),
                              "Mating system" = expression(italic("h")))) +
  scale_y_continuous(limits=c(-0.15,0.15), expand = c(0, 0)) +
  ylab("Contribution to adult sex ratio") +
  xlab("Sex bias in parameter")

Figure_S2 <-
  ggplot2::ggplot() +
  theme_bw() +
  geom_bar(data = LTRE_plover[which(LTRE_plover$model == "ASR"), ],
           aes(x = parameter, y = contribution, fill = parameter),
           color = "black", stat = "identity", size = 0.2) +
  facet_grid(. ~ population, labeller = as_labeller(population_names)) +
  theme(text = element_text(family="Franklin Gothic Book"),
        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=7, vjust=-0.1),
        axis.text.x  = element_text(size=6, angle = 45, hjust = 1),
        axis.title.y = element_text(size=7, hjust=0.5, vjust = 3.5),
        axis.text.y  = element_text(size=6),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        axis.ticks.y = element_line(size = 0.2, colour = "grey40"),
        axis.ticks.length = unit(0.1, "cm"),
        axis.ticks.x = element_line(size = 0.2, colour = "grey40"),
        panel.border = element_blank(),
        panel.margin = unit(0.3, "lines"),
        strip.background = element_blank(),
        strip.text = element_text(size=6)) +
  scale_fill_manual(values = cbPalette) +
  scale_y_continuous(limits=c(-0.15,0.15), expand = c(0, 0)) +
```

```
  scale_x_discrete(labels = c("Adult survival" = "Adult surv.",
                              "Juvenile survival" = "Juvenile surv",
                              "Hatching sex ratio" = expression(italic("rho")),
                              "Mating system" = expression(italic("h")))) +
  ylab("Contribution to adult sex ratio") +
  xlab("Sex bias in parameter")

grid.newpage()
pushViewport( viewport( layout = grid.layout( 1 , 1 , widths = unit( 1 , "npc" ) ) ) )
print( Figure_S2_background + theme(legend.position="none") ,
       vp = viewport( layout.pos.row = 1 , layout.pos.col = 1 ) )
print( Figure_S2 + theme(legend.position="none") ,
       vp = viewport( layout.pos.row = 1 , layout.pos.col = 1 ) )
```



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

```
mean(c(abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                             LTRE_plover$parameter == "Juvenile survival" &
                             LTRE_plover$population == "KIP"),2])/
         abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                               LTRE_plover$parameter == "Hatching sex ratio" &
                               LTRE_plover$population == "KIP"),2]),
       abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                             LTRE_plover$parameter == "Hatching sex ratio" &
                             LTRE_plover$population == "KPT"),2])/
         abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                               LTRE_plover$parameter == "Adult survival" &
                               LTRE_plover$population == "KPT"),2]),
       abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                             LTRE_plover$parameter == "Hatching sex ratio" &
                             LTRE_plover$population == "SP"),2])/
         abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                               LTRE_plover$parameter == "Adult survival" &
                               LTRE_plover$population == "SP"),2])))
#> [1] 24.7591
```

juvenile vs adult:

```
mean(c(abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                             LTRE_plover$parameter == "Juvenile survival" &
                             LTRE_plover$population == "KIP"),2])/
         abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
```

```
                                LTRE_plover$parameter == "Adult survival" &
                                LTRE_plover$population == "KIP"),2]),
       abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                             LTRE_plover$parameter == "Juvenile survival" &
                             LTRE_plover$population == "KPT"),2])/
         abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                                LTRE_plover$parameter == "Adult survival" &
                                LTRE_plover$population == "KPT"),2]),
       abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                             LTRE_plover$parameter == "Juvenile survival" &
                             LTRE_plover$population == "SP"),2])/
         abs(LTRE_plover[which(LTRE_plover$model == "ASR" &
                                LTRE_plover$parameter == "Adult survival" &
                                LTRE_plover$population == "SP"),2])))
#> [1] 7.300752
```

R session information

```
sessionInfo()
#> R version 3.3.1 (2016-06-21)
#> Platform: x86_64-apple-darwin13.4.0 (64-bit)
#> Running under: OS X 10.12.2 (Sierra)
#>
#> 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] extrafont_0.17     magrittr_1.5       lme4_1.1-12
#>  [4] Matrix_1.2-6       Rmisc_1.5          plyr_1.8.4
#>  [7] lattice_0.20-33    RColorBrewer_1.1-2 reshape2_1.4.1
#> [10] gridExtra_2.2.1    dplyr_0.5.0        ggplot2_2.1.0
#> [13] stringr_1.1.0      RMark_2.2.0
#>
#> loaded via a namespace (and not attached):
#>  [1] Rcpp_0.12.7      nloptr_1.0.4     formatR_1.4      tools_3.3.1
#>  [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_3.3.1
#> [13] mvtnorm_1.0-5    expm_0.999-0     Rttf2pt1_1.3.4   coda_0.18-1
#> [17] knitr_1.14       rprojroot_1.2    reshape_0.8.5    R6_2.1.3
#> [21] survival_2.40-1  rmarkdown_1.3    minqa_1.2.4      extrafontdb_1.0
#> [25] backports_1.0.5  scales_0.4.0     htmltools_0.3.5  matrixcalc_1.0-3
#> [29] splines_3.3.1    MASS_7.3-45      assertthat_0.1   colorspace_1.2-6
#> [33] labeling_0.3     stringi_1.1.1    lazyeval_0.2.0   munsell_0.4.3
#> [37] msm_1.6.1
```