# 2.1 Photo ID Mark Recapture

2024-01-08

*NOTE: I have to update the capture history with some 2023 data*

Capture mark recapture of Arctic killer whales using photo ID. We use the same methods as Kyle did for his estimate (Lefort et al. 2020).

## POPAN Jolly-Seber

Assumptions for Jolly-Seber Mark Recapture models: 1. Animals retain their tags throughout the experiment

2. Tags are read properly
3. Sampling is instantaneous
4. Survival probabilities are the same for all animals (marked and unmarked) between each pair of sampling occasions (homogenous survival)
5. Catchability is the same for all animals (marked and unmarked) at each sampling occasion (homogenous catchability)
6. The study area is constant

In the case of killer whales, do we meet the assumptions?

1. Yes - most nicks and scars used for ID are retained through the life of the animal.
2. Yes - we can assume that identified individuals are re-identified reliably. However, there are ways to account for identification error - to be explored later.
3. Yes - sampling period is short (1 to a few days)
4. Probably? - while survival probabilities might differ between sex and age classes, being "marked" does not affect an individual's survival
5. Unlikely? - Equal catchability could be affected by:
    - Behaviour - some individuals/groups may be more likely to approach the boat, and thus we may get more/better photographs
    - Individuals with more distinct markings may be more likely to be identified/re-identified when image quality is lower
    - Cooch and White (2014) describe this as the most critical assumption for JS models
6. Sort of? - The study area includes locations around Northern Baffin Island (mainly Admiralty Inlet and Eclipse Sound) and Cumberland Sound, but we have not consistently sampled in each location each year

Here we estimate population size using a POPAN Jolly-Seber Mark Recapture Model. The POPAN formulation of the JS model is an open population model that implies the existence of a super-population consisting of all animals that would ever be born to the population. Parameter *b_i* (entry probability) represents the probability that an animal from they hypothetical super-population would enter the population between occasion *t* and *t+1*. Entry could result from recruitment or immigration.

**Figure 12.2:** *Process model for POPAN parameterization of JS expe*
*capture at occasion $i$; $\varphi_i$ represents the probability of an animal*
*and $b_i$ represents the probability that an animal from the super-po*
*between occasions $i$ and $i$+1 and survive to the next sampling occa*
*not to happened, but are easily included.*

$$\begin{tikzcd}
b_0 & & b_1 & & b_2 & & b_3 & \\
& t_1 \arrow[rr, "\varphi_1"] & & t_2 \arrow[rr, "\varphi_2"] & & t_3 \arrow[rr, "\varphi_3"] & & t_4 \\
& p_1 \arrow[u] & & p_2 \arrow[u] & & p_3 \arrow[u] & & p_4 \arrow[u]
\end{tikzcd}$$

The parameterization of the POPAN JS Model is as follows:

**Apply to KW Data** Prep the environment:

```r
rm(list=ls())

setwd("~/Documents/Master's/Analysis/CMR")

library(RMark)
```

```
## This is RMark 3.0.0
##  Documentation available at http://www.phidot.org/software/mark/rmark/RMarkDocumentation.zip
```

```r
library(dplyr)
```

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

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

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

Prep the CMR data:

```r
CMR_data <- read.csv("photo_CMR_data.csv", header = TRUE)

# Inserting zeros insead of NAs in years where there are no sightings:
CMR_data[is.na(CMR_data)] <- 0

# Remove the id column:
CMR_data <- CMR_data %>%
  select(-ID) %>%
  mutate(cmr = NA) %>%
  relocate(cmr, "X2009")

# Creating a column with 1s and 0s for all years observed:
for (i in 1:nrow(CMR_data)){
  CMR_data[i,1] <- paste(CMR_data[i,2], CMR_data[i,3], CMR_data[i,4], CMR_data[i,5], CMR_data[i,6],CMR_
                         CMR_data[i,8], CMR_data[i,9], CMR_data[i,10], CMR_data[i,11], CMR_data[i,12],
                         sep = "")}
```

```r
head(CMR_data)
```

```
##            cmr X2009 X2010 X2011 X2012 X2013 X2014 X2015 X2016 X2017 X2018
## 1 100010001101     1     0     0     0     1     0     0     0     1     1
## 2 100010100101     1     0     0     0     1     0     1     0     0     1
## 3 100000000010     1     0     0     0     0     0     0     0     0     0
## 4 100000000000     1     0     0     0     0     0     0     0     0     0
## 5 100000000000     1     0     0     0     0     0     0     0     0     0
## 6 100000000000     1     0     0     0     0     0     0     0     0     0
##   X2019 X2020
## 1     0     1
## 2     0     1
## 3     1     0
## 4     0     0
## 5     0     0
## 6     0     0
```

```r
# Save to .csv:
#write.csv(CMR_data, "CMR_data.csv")
```

Extract first column with CMR data:

```r
CMR <- CMR_data %>%
  select(cmr)

# Change name of first column:
colnames(CMR)[colnames(CMR)=="cmr"] <- "ch"
```

Save the CMR data to a txt file and import as ch data:

```r
#write.table(CMR, file = "_photo_CMR_data.txt", row.names = FALSE, quote = FALSE)

ch_CMR <- import.chdata("_photo_CMR_data.txt")

summary(ch_CMR)
attach(ch_CMR)
```

Start building model:

```r
kw.proc = process.data(ch_CMR, model = "POPAN")
kw.ddl  = make.design.data(kw.proc)
```

Specify effects to consider on survival and detection probabilities:

```r
# Survival process:
phi.ct   = list(formula = ~1)      # constant
phi.time = list(formula = ~time)   # year effect

# Detection process:
p.ct   = list(formula = ~1)        # constant
p.time = list(formula = ~time)     # year effect

# Entry process:
pent.ct   = list(formula = ~1)     # constant
pent.time = list(formula = ~time)  # year effect
```

## POPAN Jolly-Seber Model

Fit models:

```r
# phi  = survival
# p    = detection
# pent = entry

# Model 1: constant survival, constant recapture, constant entry
model.1 = mark(kw.proc, kw.ddl, output = FALSE, delete = T,
               model.parameters = list(Phi = phi.ct, p = p.ct, pent = pent.ct))

# Model 2: time-dependent survival, constant recapture, constant entry
model.2 = mark(kw.proc, kw.ddl, output = FALSE, delete = T,
               model.parameters = list(Phi = phi.time, p = p.ct, pent = pent.ct))

# Model 3: constant survival, time-dependent recapture, constant entry
model.3 = mark(kw.proc, kw.ddl, output = FALSE, delete = T,
               model.parameters = list(Phi = phi.ct, p = p.time, pent = pent.ct))

# Model 4: constant survival, constant recapture, time-dependent entry
model.4 = mark(kw.proc, kw.ddl, output = FALSE, delete = T,
               model.parameters = list(Phi = phi.ct, p = p.ct, pent = pent.time))

# Model 5: time-dependent survival, time-dependent recapture, constant entry
model.5 = mark(kw.proc, kw.ddl, output = FALSE, delete = T,
               model.parameters = list(Phi = phi.time, p = p.time, pent = pent.ct))

# Model 6: time-dependent survival, constant recapture, time-dependent entry
model.6 = mark(kw.proc, kw.ddl, output = FALSE, delete = T,
               model.parameters = list(Phi = phi.time, p = p.ct, pent = pent.time))

# Model 7: constant survival, time-dependent recapture, time-dependent entry
model.7 = mark(kw.proc, kw.ddl, output = FALSE, delete = T,
               model.parameters = list(Phi = phi.ct, p = p.time, pent = pent.time))

# Model 8: time-dependent survival, time-dependent recapture, time-dependent entry
model.8 = mark(kw.proc, kw.ddl, output = FALSE, delete = T,
               model.parameters = list(Phi = phi.time, p = p.time, pent = pent.time))
```

Take a look at AIC values:

```r
AIC_models <- c("Model 1", "Model 2", "Model 3", "Model 4", "Model 5", "Model 6", "Model 7", "Model 8")

AICc_values <- c(summary(model.1)$AICc,
                summary(model.2)$AICc,
                summary(model.3)$AICc,
                summary(model.4)$AICc,
                summary(model.5)$AICc,
                summary(model.6)$AICc,
                summary(model.7)$AICc,
                summary(model.8)$AICc)

AIC_table <- as.data.frame(cbind(AIC_models, AICc_values))
colnames(AIC_table)[1:2] = c("Model", "AICc")
```

```r
AIC_table$AICc <- as.numeric(AIC_table$AICc)

AIC_table <- AIC_table %>%
  mutate(delta_AICc = AICc - min(AICc))
AIC_table
```

```
##     Model     AICc delta_AICc
## 1 Model 1 338.8956  86.739534
## 2 Model 2 319.7535  67.597447
## 3 Model 3 252.1561   0.000000
## 4 Model 4 287.6777  35.521607
## 5 Model 5 269.7367  17.580654
## 6 Model 6 290.1222  37.966082
## 7 Model 7 254.7019   2.545764
## 8 Model 8 282.1385  29.982394
```

Model 3 has the best support according to the AIC. Let’s take a closer look at the parameter estimates for model 3:

```r
# Estimate for survival (constant):
phi.table = get.real(model.3,"Phi", se = TRUE) # Estimate for survival is 0.85
phi.table[c("estimate","se","lcl","ucl")][1,]
```

```
##                estimate        se      lcl       ucl
## Phi g1 a0 t1 0.836653 0.0624843 0.676428 0.9261952
```

```r
# Estimate for recapture (time-dependent):
p.table = get.real(model.3,"p", se= TRUE)
p.table[c("estimate","se","lcl","ucl")][1:11,]
```

```
##                    estimate           se           lcl          ucl
## p g1 a0 t1   3.848246e-01 2.801276e-01  5.795520e-02 8.641440e-01
## p g1 a1 t2   1.064052e-01 5.992720e-02  3.346200e-02 2.905559e-01
## p g1 a2 t3   1.019558e-01 4.755020e-02  3.940860e-02 2.390682e-01
## p g1 a3 t4   2.807533e-35 1.100529e-27 -2.157038e-27 2.157038e-27
## p g1 a4 t5   3.051018e-01 9.629090e-02  1.527369e-01 5.167575e-01
## p g1 a5 t6   5.404310e-02 2.795780e-02  1.918410e-02 1.430077e-01
## p g1 a6 t7   1.002710e-02 1.039350e-02  1.299400e-03 7.308810e-02
## p g1 a7 t8   4.378639e-10 1.355627e-06 -2.656590e-06 2.657466e-06
## p g1 a8 t9   3.570690e-02 2.073430e-02  1.124720e-02 1.075732e-01
## p g1 a9 t10  1.869804e-01 6.906270e-02  8.625940e-02 3.590900e-01
## p g1 a10 t11 1.553263e-01 5.997900e-02  6.982050e-02 3.105830e-01
```

```r
format(p.table, scientific = FALSE)
```

```
##                all.diff.index par.index
## p g1 a0 t1                 12         2
## p g1 a1 t2                 13         3
## p g1 a2 t3                 14         4
## p g1 a3 t4                 15         5
## p g1 a4 t5                 16         6
## p g1 a5 t6                 17         7
## p g1 a6 t7                 18         8
## p g1 a7 t8                 19         9
## p g1 a8 t9                 20        10
## p g1 a9 t10                21        11
## p g1 a10 t11             22        12
## p g1 a11 t12             23        13
##                                                 estimate
## p g1 a0 t1   0.38482460000000001665299009800946805626154
## p g1 a1 t2   0.10640520000000000533368904598319204524159
## p g1 a2 t3   0.10195579999999999920579085710414801724255
## p g1 a3 t4   0.00000000000000000000000000000000002807533
## p g1 a4 t5   0.30510179999999999785742943458893899524212
## p g1 a5 t6   0.05404309999999999966965091857673542108386 8
## p g1 a6 t7   0.01002710000000000059583449285582901211455
## p g1 a7 t8   0.00000000043786389999999997726028584102073
## p g1 a8 t9   0.03570689999999999975077713543214485980570
## p g1 a9 t10  0.18698039999999999910563985849876189604401 6
## p g1 a10 t11 0.15532630000000000047855053253442747518420
## p g1 a11 t12 0.15996640000000000085854878761892905458807 9
##                                                se
## p g1 a0 t1   0.28012759999999999766210123652854 24
## p g1 a1 t2   0.05992719999999999991313615055332 8
## p g1 a2 t3   0.04755020000000000079953821341405 3
## p g1 a3 t4   0.00000000000000000000000000001100529
## p g1 a4 t5   0.09629089999999999987151610980617 99
## p g1 a5 t6   0.02795780000000000140025768757823 2
## p g1 a6 t7   0.01039349999999999996092014953319 4
## p g1 a7 t8   0.00000135562700000000000990801824 27
## p g1 a8 t9   0.02073430000000000006466382984626 76
## p g1 a9 t10  0.06906270000000000046409098786170 94
## p g1 a10 t11 0.05997899999999999997594368750242 211
## p g1 a11 t12 0.06336340000000000002783551167340 19
##                                                lcl
## p g1 a0 t1    0.05795519999999999983836929118297 12
## p g1 a1 t2    0.03346199999999999986894927417324 65
## p g1 a2 t3    0.03940860000000000019616308577496 97
## p g1 a3 t4   -0.00000000000000000000000000002157038
## p g1 a4 t5    0.15273690000000000087702289874869 17
## p g1 a5 t6    0.01918409999999999990699439678110 14
## p g1 a6 t7    0.00129940000000000003416156246771 6
## p g1 a7 t8   -0.00000265658999999999998351985896 34
## p g1 a8 t9    0.01124720000000000005998090912839 87
## p g1 a9 t10   0.08625939999999999999639399561601 75
## p g1 a10 t11  0.06982049999999999993678834186994 209
## p g1 a11 t12  0.07026110000000000006882139302888 390
##                                               ucl fixed note group age time Age
## p g1 a0 t1   0.86414400000000000232205366046400 74                1   0    1   0
## p g1 a1 t2   0.29055589999999999780783355163293 89                1   1    2   1
## p g1 a2 t3   0.23906820000000000084760642948822 35                1   2    3   2
## p g1 a3 t4   0.00000000000000000000000000002157038                1   3    4   3
## p g1 a4 t5   0.51675749999999999807442918609012 85                1   4    5   4
## p g1 a5 t6   0.14300769999999999876056477887686 92                1   5    6   5
## p g1 a6 t7   0.07308810000000000030647484550172 52                1   6    7   6
## p g1 a7 t8   0.00000265746599999999998756005370 47                1   7    8   7
## p g1 a8 t9   0.10757319999999999938255612619286 69                1   8    9   8
## p g1 a9 t10  0.35909000000000000202859951059508 60                1   9   10   9
## p g1 a10 t11 0.31058299999999999979642950620473 13                1  10   11  10
## p g1 a11 t12 0.32425809999999999935567984721274 12                1  11   12  11
##                Time
## p g1 a0 t1      0
## p g1 a1 t2      1
## p g1 a2 t3      2
## p g1 a3 t4      3
## p g1 a4 t5      4
## p g1 a5 t6      5
## p g1 a6 t7      6
## p g1 a7 t8      7
## p g1 a8 t9      8
## p g1 a9 t10     9
## p g1 a10 t11   10
## p g1 a11 t12   11
```

```
# Estimate for entry (constant):
pent.table = get.real(model.3,"pent", se= TRUE)
pent.table[c("estimate","se","lcl","ucl")][1,]
```

```
##                  estimate        se       lcl       ucl
## pent g1 a1 t2 0.0776543 0.0092838 0.0613007 0.0979157
```

```
# Estimate for population size - I am not sure if this is correct, stuff online makes it seem like you
N.table = get.real(model.3,"N", se= TRUE)
N.table[c("estimate","se","lcl","ucl")][1,]
```

```
##             estimate       se      lcl      ucl
## N g1 a0 t1 285.1632 51.00353 209.2063 414.9366
```

Put estimates for superpopulation size from each model in a table, and add columns for an adjusted estimate + adjusted SE (2/3, as done in Lefort et al. 2020)):

```
N.table1 = get.real(model.1,"N", se= TRUE)
N.table2 = get.real(model.2,"N", se= TRUE)
N.table3 = get.real(model.3,"N", se= TRUE)
N.table4 = get.real(model.4,"N", se= TRUE)
N.table5 = get.real(model.5,"N", se= TRUE)
N.table6 = get.real(model.6,"N", se= TRUE)
N.table7 = get.real(model.7,"N", se= TRUE)
N.table8 = get.real(model.8,"N", se= TRUE)

results_table <- rbind(N.table1[c("estimate","se","lcl","ucl")][1,],
                       N.table2[c("estimate","se","lcl","ucl")][1,],
                       N.table3[c("estimate","se","lcl","ucl")][1,],
                       N.table4[c("estimate","se","lcl","ucl")][1,],
                       N.table5[c("estimate","se","lcl","ucl")][1,],
                       N.table6[c("estimate","se","lcl","ucl")][1,],
                       N.table7[c("estimate","se","lcl","ucl")][1,],
                       N.table8[c("estimate","se","lcl","ucl")][1,])

rownames(results_table) <- NULL

Model_table <- cbind(AIC_table, results_table)

Model_table <- Model_table %>%
  arrange(delta_AICc) %>%
  mutate(adj_estimate = estimate/(2/3),
         adj_se = se/(2/3),
         across(where(is.numeric), ~ round(., 2)))
Model_table
```

```
##     Model   AICc delta_AICc estimate    se    lcl    ucl adj_estimate adj_se
## 1 Model 3 252.16       0.00   285.16 51.00 209.21 414.94       427.74  76.51
## 2 Model 7 254.70       2.55   355.33 95.64 225.88 620.08       533.00 143.45
## 3 Model 5 269.74      17.58   219.30 47.23 156.87 352.75       328.95  70.85
## 4 Model 8 282.14      29.98   357.11  0.00 357.11 357.11       535.66   0.00
## 5 Model 4 287.68      35.52   291.35 53.47 212.03 427.86       437.03  80.20
## 6 Model 6 290.12      37.97   294.84 57.31 211.03 443.06       442.26  85.96
## 7 Model 2 319.75      67.60   326.36 59.73 236.34 476.69       489.54  89.59
## 8 Model 1 338.90      86.74   309.39 56.62 224.63 452.76       464.09  84.93
```

### Pradel Survival-Lambda

Unfortunately, the POPAN formulation does not allow inference of a population growth rate. We can use a Pradel Survival-Lambda model to estimate the realized population growth rate ($\lambda$).

Cooch and White (2014) note that the lambda estimated from Pradel models is the realized growth rate of the age class from which the encounter histories were generated, and thus not necessarily equivalent to the growth rate of the population.

The assumptions are the same as the POPAN formulation - see discussion above.

Start building model:

```
kw.pradel.proc = process.data(ch_CMR, model = "Pradlambda")
kw.pradel.ddl  = make.design.data(kw.pradel.proc)
```

Specify effects to consider on survival and detection probabilities:

```
# Survival process:
phi.pradel.ct   = list(formula = ~1)      # constant
phi.prade.time = list(formula = ~time)   # year effect

# Detection process:
p.pradel.ct    = list(formula = ~1)        # constant
p.pradel.time = list(formula = ~time)     # year effect

# We assume that the growth rate is constant
```

Fit models:

```
# phi  = survival
# p    = detection
# pent = entry

# Model 1: constant survival, constant recapture
model.pradel.1 = mark(kw.pradel.proc, kw.pradel.ddl,
                      model.parameters = list(Phi = phi.pradel.ct, p = p.pradel.ct))

# Model 2: constant survival, time-dependent recapture
model.pradel.2 = mark(kw.pradel.proc, kw.pradel.ddl,
                      model.parameters = list(Phi = phi.pradel.ct, p = p.pradel.time))

# Model 3: time-dependent survival, constant recapture
```

```r
model.pradel.3 = mark(kw.pradel.proc, kw.pradel.ddl,
                      model.parameters = list(Phi = phi.prade.time, p = p.pradel.ct))

# Model 4: time-dependent survival, time-dependent recapture
model.pradel.4 = mark(kw.pradel.proc, kw.pradel.ddl,
                      model.parameters = list(Phi = phi.prade.time, p = p.pradel.time))
```

Take a look at AIC values:

```r
AIC_pradel_models <- c("Pradel Model 1", "Pradel Model 2", "Pradel Model 3", "Pradel Model 4")

AICc_pradel_values <- c(summary(model.pradel.1)$AICc,
                        summary(model.pradel.2)$AICc,
                        summary(model.pradel.3)$AICc,
                        summary(model.pradel.4)$AICc)

AIC_pradel_table <- as.data.frame(cbind(AIC_pradel_models, AICc_pradel_values))
colnames(AIC_pradel_table)[1:2] = c("Model", "AICc")

AIC_pradel_table$AICc <- as.numeric(AIC_pradel_table$AICc)

AIC_pradel_table <- AIC_pradel_table %>%
  mutate(delta_AICc = AICc - min(AICc))
AIC_pradel_table # Model 2 has best support - constant survival, time-dependent recapture
```

```
##            Model     AICc delta_AICc
## 1 Pradel Model 1 687.2661   84.24295
## 2 Pradel Model 2 603.0232    0.00000
## 3 Pradel Model 3 697.1136   94.09042
## 4 Pradel Model 4 619.9825   16.95937
```

Model 2 has the best support according to the AIC. Let's take a closer look at the parameter estimates for model 2:

```r
# Estimate for survival (constant):
phi.pradel.table = get.real(model.pradel.2,"Phi", se = TRUE)
phi.pradel.table[c("estimate","se","lcl","ucl")][1,] # Estimate for survival is 0.83
```

```
##              estimate        se       lcl       ucl
## Phi g1 a0 t1 0.829041 0.0604804 0.6775352 0.9179809
```

```r
# Estimate for recapture (time-dependent):
p.pradel.table = get.real(model.pradel.2,"p", se= TRUE)
p.pradel.table[c("estimate","se","lcl","ucl")][1:11,]
```

```
##                  estimate           se           lcl          ucl
## p g1 a0 t1   4.281708e-01 1.676523e-01  1.636372e-01 7.413072e-01
## p g1 a1 t2   1.417881e-01 6.972530e-02  5.099910e-02 3.368343e-01
## p g1 a2 t3   1.452436e-01 6.594510e-02  5.659790e-02 3.249122e-01
## p g1 a3 t4   6.077996e-10 2.955333e-06 -5.791845e-06 5.793061e-06
## p g1 a4 t5   4.347973e-01 1.401205e-01  2.010326e-01 7.016661e-01
## p g1 a5 t6   7.341970e-02 3.800590e-02  2.582360e-02 1.914965e-01
## p g1 a6 t7   1.264010e-02 1.307180e-02  1.640600e-03 9.068690e-02
## p g1 a7 t8   1.950573e-09 4.490059e-06 -8.798565e-06 8.802466e-06
## p g1 a8 t9   3.742250e-02 2.158130e-02  1.187230e-02 1.117411e-01
## p g1 a9 t10  1.765529e-01 6.581190e-02  8.112660e-02 3.423993e-01
## p g1 a10 t11 1.311586e-01 5.287790e-02  5.731160e-02 2.726391e-01
```

```
# Estimate for lambda (constant):
lambda.pradel.table = get.real(model.pradel.2,"Lambda", se= TRUE)
lambda.pradel.table[c("estimate","se","lcl","ucl")][1,]
```

```
##                   estimate        se      lcl      ucl
## Lambda g1 a0 t1 1.149689 0.0557031 1.045593 1.264149
```

```
# Get derived estimates:
model.pradel.2$results$derived
```

```
## $`Lambda Population Change`
##    estimate         se      lcl      ucl
## 1  1.149689 0.05570308 1.045593 1.264149
## 2  1.149689 0.05570308 1.045593 1.264149
## 3  1.149689 0.05570308 1.045593 1.264149
## 4  1.149689 0.05570308 1.045593 1.264149
## 5  1.149689 0.05570308 1.045593 1.264149
## 6  1.149689 0.05570308 1.045593 1.264149
## 7  1.149689 0.05570308 1.045593 1.264149
## 8  1.149689 0.05570308 1.045593 1.264149
## 9  1.149689 0.05570308 1.045593 1.264149
## 10 1.149689 0.05570308 1.045593 1.264149
## 11 1.149689 0.05570308 1.045593 1.264149
##
## $`log(Lambda) Population Change`
##     estimate         se        lcl       ucl
## 1  0.1394917 0.04845055 0.04452865 0.2344548
## 2  0.1394917 0.04845055 0.04452865 0.2344548
## 3  0.1394917 0.04845055 0.04452865 0.2344548
## 4  0.1394917 0.04845055 0.04452865 0.2344548
## 5  0.1394917 0.04845055 0.04452865 0.2344548
## 6  0.1394917 0.04845055 0.04452865 0.2344548
## 7  0.1394917 0.04845055 0.04452865 0.2344548
## 8  0.1394917 0.04845055 0.04452865 0.2344548
## 9  0.1394917 0.04845055 0.04452865 0.2344548
## 10 0.1394917 0.04845055 0.04452865 0.2344548
## 11 0.1394917 0.04845055 0.04452865 0.2344548
```

An estimate of 1.18 is pretty big.

Put it all in a table:

```
lamb.table1 = get.real(model.pradel.1,"Lambda", se= TRUE)
lamb.table2 = get.real(model.pradel.2,"Lambda", se= TRUE)
lamb.table3 = get.real(model.pradel.3,"Lambda", se= TRUE)
lamb.table4 = get.real(model.pradel.4,"Lambda", se= TRUE)

results_pradel_table <- rbind(lamb.table1[c("estimate","se","lcl","ucl")][1,],
                              lamb.table2[c("estimate","se","lcl","ucl")][1,],
                              lamb.table3[c("estimate","se","lcl","ucl")][1,],
                              lamb.table4[c("estimate","se","lcl","ucl")][1,]
                              )

rownames(results_pradel_table) <- NULL
```

```r
Model_pradel_table <- cbind(AIC_pradel_table, results_pradel_table)

Model_pradel_table <- Model_pradel_table %>%
  arrange(delta_AICc)
Model_pradel_table
```

```
##            Model     AICc delta_AICc estimate        se       lcl      ucl
## 1 Pradel Model 2 603.0232    0.00000 1.149689 0.0557031 1.0455931 1.264149
## 2 Pradel Model 4 619.9825   16.95937 1.155513 0.1036422 0.9695694 1.377117
## 3 Pradel Model 1 687.2661   84.24295 1.085595 0.0287423 1.0307057 1.143406
## 4 Pradel Model 3 697.1136   94.09042 1.052646 0.0314277 0.9928278 1.116067
```

## Link-Barker Model:

The Link-Barker model is similar to the Pradel survival-lambda in that it estimates realized population growth rate, but does so as an extension of the POPAN Jolly-Seber Model.

Start building model:

```r
kw.lb.proc = process.data(ch_CMR, model = "LinkBarker")
kw.lb.ddl  = make.design.data(kw.lb.proc)
```

Specify effects to consider on survival and detection probabilities:

```r
# Survival process:
phi.lb.ct   = list(formula = ~1)      # constant
phi.lb.time = list(formula = ~time)    # year effect

# Detection process:
p.lb.ct   = list(formula = ~1)        # constant
p.lb.time = list(formula = ~time)     # year effect
```

Fit models:

```r
# phi  = survival
# p    = detection
# pent = entry

# Model 1: constant survival, constant recapture
model.lb.1 = mark(kw.lb.proc, kw.lb.ddl,
                      model.parameters = list(Phi = phi.lb.ct, p = p.lb.ct))

# Model 2: constant survival, time-dependent recapture
model.lb.2 = mark(kw.lb.proc, kw.lb.ddl,
                      model.parameters = list(Phi = phi.lb.ct, p = p.lb.time))

# Model 3: time-dependent survival, constant recapture
model.lb.3 = mark(kw.lb.proc, kw.lb.ddl,
                      model.parameters = list(Phi = phi.lb.time, p = p.lb.ct))

# Model 4: time-dependent survival, time-dependent recapture
model.lb.4 = mark(kw.lb.proc, kw.lb.ddl,
                      model.parameters = list(Phi = phi.lb.time, p = p.lb.time))
```

Take a look at AIC values:

```r
AIC_lb_models <- c("LinkBarker Model 1", "LinkBarker Model 2", "LinkBarker Model 3", "LinkBarker Model 

AICc_lb_values <- c(summary(model.lb.1)$AICc,
                        summary(model.lb.2)$AICc,
                        summary(model.lb.3)$AICc,
                        summary(model.lb.4)$AICc)

AIC_lb_table <- as.data.frame(cbind(AIC_lb_models, AICc_lb_values))
colnames(AIC_lb_table)[1:2] = c("Model", "AICc")

AIC_lb_table$AICc <- as.numeric(AIC_lb_table$AICc)

AIC_lb_table <- AIC_lb_table %>%
  mutate(delta_AICc = AICc - min(AICc))
AIC_lb_table # Model 2 has best support - constant survival, time-dependent recapture
```

```
##                Model     AICc delta_AICc
## 1 LinkBarker Model 1 687.2661   84.24295
## 2 LinkBarker Model 2 603.0232    0.00000
## 3 LinkBarker Model 3 674.6131   71.58995
## 4 LinkBarker Model 4 620.5056   17.48244
```

Model 2 has the best support according to the AIC. Parameter estimates for model 2:

```r
# Get derived estimates:
model.lb.2$results$derived
```

```
## $`Lambda Population Change`
##    estimate         se      lcl      ucl
## 1  1.149689 0.05570311 1.045593 1.264149
## 2  1.149689 0.05570311 1.045593 1.264149
## 3  1.149689 0.05570311 1.045593 1.264149
## 4  1.149689 0.05570311 1.045593 1.264149
## 5  1.149689 0.05570311 1.045593 1.264149
## 6  1.149689 0.05570311 1.045593 1.264149
## 7  1.149689 0.05570311 1.045593 1.264149
## 8  1.149689 0.05570311 1.045593 1.264149
## 9  1.149689 0.05570311 1.045593 1.264149
## 10 1.149689 0.05570311 1.045593 1.264149
## 11 1.149689 0.05570311 1.045593 1.264149
##
## $`log(Lambda) Population Change`
##     estimate         se        lcl       ucl
## 1  0.1394917 0.04845058 0.04452861 0.2344549
## 2  0.1394917 0.04845058 0.04452861 0.2344549
## 3  0.1394917 0.04845058 0.04452861 0.2344549
## 4  0.1394917 0.04845058 0.04452861 0.2344549
## 5  0.1394917 0.04845058 0.04452861 0.2344549
## 6  0.1394917 0.04845058 0.04452861 0.2344549
## 7  0.1394917 0.04845058 0.04452861 0.2344549
## 8  0.1394917 0.04845058 0.04452861 0.2344549
## 9  0.1394917 0.04845058 0.04452861 0.2344549
## 10 0.1394917 0.04845058 0.04452861 0.2344549
## 11 0.1394917 0.04845058 0.04452861 0.2344549
```