2.3 Genetic Mark Recapture

2024-01-08

Capture mark recapture of Arctic killer whales using genetic (whole genome) mark recapture. Using the same statistical methods as Kyle did for his estimate (Lefort et al. 2020) and in 2.1_PhotoID_CMR.

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 ciritical assumption for JS models 6. Sort of? - The study area is confined to 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

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:

```
CMR_data_gen <- read.csv("genetic_CMR_data_PRELIM.csv", header = TRUE)

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

# Remove the id column:
CMR_data_gen <- CMR_data_gen %>%
  select(-genome_sample_ID) %>%
  mutate(cmr = NA) %>%
  relocate(cmr, "X2013")

# Creating a column with 1s and 0s for all years observed:
for (i in 1:nrow(CMR_data_gen)){
  CMR_data_gen[i,1] <- paste(CMR_data_gen[i,2], CMR_data_gen[i,3], CMR_data_gen[i,4], CMR_data_gen[i,5]
                             CMR_data_gen[i,6], CMR_data_gen[i,7], CMR_data_gen[i,8], CMR_data_gen[i,9]
                             CMR_data_gen[i,10], CMR_data_gen[i,11],sep = "")}

head(CMR_data_gen)
```

```
##          cmr X2013 X2014 X2015 X2016 X2017 X2018 X2019 X2020 X2021 X2022
## 1 1000000000     1     0     0     0     0     0     0     0     0     0
## 2 1000000100     1     0     0     0     0     0     0     1     0     0
## 3 1000000000     1     0     0     0     0     0     0     0     0     0
## 4 1000000000     1     0     0     0     0     0     0     0     0     0
## 5 1000000000     1     0     0     0     0     0     0     0     0     0
## 6 1000000100     1     0     0     0     0     0     0     1     0     0
```

```
# Save to .csv:
#write.csv(CMR_data_gen, "CMR_data_genetic.csv")
```

Extract first column with CMR data:

```
CMR_gen <- CMR_data_gen %>%
  select(cmr)

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

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

```
write.table(CMR_gen, file = "_genetic_CMR_data.txt", row.names = FALSE, quote = FALSE)

ch_CMR_gen <- import.chdata("_genetic_CMR_data.txt")

summary(ch_CMR_gen)
attach(ch_CMR_gen)
```

Start building model:

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

Specify effects to consider on survival and detection probabilities:

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

```r
# Detection (capture) 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
```

Fit models:

```r
# phi  = survival
# p    = detection (capture)
# pent = entry

# Model 1: constant survival, constant recapture, constant entry
model.1.gen = mark(kw.proc.gen, kw.ddl.gen, 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.gen = mark(kw.proc.gen, kw.ddl.gen, 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.gen = mark(kw.proc.gen, kw.ddl.gen, 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.gen = mark(kw.proc.gen, kw.ddl.gen, 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.gen = mark(kw.proc.gen, kw.ddl.gen, 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.gen = mark(kw.proc.gen, kw.ddl.gen, 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.gen = mark(kw.proc.gen, kw.ddl.gen, 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.gen = mark(kw.proc.gen, kw.ddl.gen, 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.gen)$AICc,
                summary(model.2.gen)$AICc,
                summary(model.3.gen)$AICc,
                summary(model.4.gen)$AICc,
                 summary(model.5.gen)$AICc,
                 summary(model.6.gen)$AICc,
                 summary(model.7.gen)$AICc,
                 summary(model.8.gen)$AICc)

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

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

AIC_table <- AIC_table %>%
  mutate(delta_AICc = AICc - min(AICc))
AIC_table # Model 3 has best support
```

```
##     Model      AICc delta_AICc
## 1 Model 1 157.18723   67.73995
## 2 Model 2 148.68950   59.24223
## 3 Model 3  89.44727    0.00000
## 4 Model 4 141.05611   51.60884
## 5 Model 5 128.33616   38.88889
## 6 Model 6 138.98284   49.53556
## 7 Model 7 128.33616   38.88889
## 8 Model 8 198.33616  108.88889
```

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

```
# Estimate for survival (constant):
phi.table = get.real(model.3.gen,"Phi", se = TRUE) # Estimate for survival is 1
```

```
## Warning in get.real(model.3.gen, "Phi", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```
phi.table[c("estimate","se","lcl","ucl")][1,]
```

```
##                 estimate           se           lcl ucl
## Phi g1 a0 t1 0.9999999 9.759653e-05 1.107488e-301   1
```

```
# Estimate for recapture (time-dependent):
p.table = get.real(model.3.gen,"p", se= TRUE)
```

```
## Warning in get.real(model.3.gen, "p", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```
p.table[c("estimate","se","lcl","ucl")][1:9,]
```

```
##                 estimate           se           lcl          ucl
## p g1 a0 t1 4.410640e-02 2.364070e-02  1.514140e-02 1.216366e-01
## p g1 a1 t2 3.632856e-33 0.000000e+00  3.632856e-33 3.632856e-33
## p g1 a2 t3 1.464522e-31 0.000000e+00  1.464522e-31 1.464522e-31
## p g1 a3 t4 4.940680e-36 0.000000e+00  4.940680e-36 4.940680e-36
## p g1 a4 t5 3.686661e-35 8.376803e-27 -1.641853e-26 1.641853e-26
## p g1 a5 t6 5.040720e-02 2.616480e-02  1.785810e-02 1.341770e-01
## p g1 a6 t7 6.300920e-02 3.115180e-02  2.334890e-02 1.590645e-01
## p g1 a7 t8 1.512217e-01 6.523570e-02  6.173280e-02 3.254402e-01
## p g1 a8 t9 2.508653e-10 1.009105e-06 -1.977594e-06 1.978096e-06
```

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

```
##               all.diff.index par.index
## p g1 a0 t1                10         2
## p g1 a1 t2                11         3
## p g1 a2 t3                12         4
## p g1 a3 t4                13         5
## p g1 a4 t5                14         6
## p g1 a5 t6                15         7
## p g1 a6 t7                16         8
## p g1 a7 t8                17         9
## p g1 a8 t9                18        10
## p g1 a9 t10               19        11
##                                                       estimate
## p g1 a0 t1  0.0441063999999999969697994515190657693893
## p g1 a1 t2  0.0000000000000000000000000000000363285600
## p g1 a2 t3  0.0000000000000000000000000000014645220000
## p g1 a3 t4  0.0000000000000000000000000000000000494068
## p g1 a4 t5  0.0000000000000000000000000000000003686661
## p g1 a5 t6  0.0504071999999999922923876738423132337630
## p g1 a6 t7  0.0630092000000000131823441051892586983740
## p g1 a7 t8  0.1512216999999998668549494595936266705394
## p g1 a8 t9  0.0000000002508652999999998362065154634143
## p g1 a9 t10 0.0063007999999999963017360826711410481948
##                                               se
## p g1 a0 t1  0.0236407000000000060236260424062 5
## p g1 a1 t2  0.00000000000000000000000000000000
## p g1 a2 t3  0.00000000000000000000000000000000
## p g1 a3 t4  0.00000000000000000000000000000000
## p g1 a4 t5  0.00000000000000000000000008376803
## p g1 a5 t6  0.02616479999999998475486151505720
## p g1 a6 t7  0.03115180000000000323563398296756
## p g1 a7 t8  0.06523569999999993692334498973651
## p g1 a8 t9  0.00000100910499999999929406854955
## p g1 a9 t10 0.00674049999999999963806729397220
##                                                        lcl
## p g1 a0 t1   0.0151413999999999930402339032298186793923
## p g1 a1 t2   0.0000000000000000000000000000000363285600
## p g1 a2 t3   0.0000000000000000000000000000014645220000
## p g1 a3 t4   0.0000000000000000000000000000000000494068
## p g1 a4 t5  -0.0000000000000000000000000164185300000000
## p g1 a5 t6   0.0178580999999999928079744190745259402320
## p g1 a6 t7   0.0233488999999999982218659763566392939538
## p g1 a7 t8   0.0617327999999999739790368380454310681671
## p g1 a8 t9  -0.0000019775940000000017856853669340200241
## p g1 a9 t10  0.0007680968999999999476115730701053507801
##                                                     ucl fixed note group age
## p g1 a0 t1  0.1216365999999999750119883401566767133772                 1   0
## p g1 a1 t2  0.0000000000000000000000000000000363285600                 1   1
## p g1 a2 t3  0.0000000000000000000000000000014645220000                 1   2
## p g1 a3 t4  0.0000000000000000000000000000000000494068                 1   3
## p g1 a4 t5  0.0000000000000000000000000164185300000000                 1   4
## p g1 a5 t6  0.1341769999999999072031187097309157252312                 1   5
## p g1 a6 t7  0.1590644999999999755928570266405586153269                 1   6
## p g1 a7 t8  0.32544020000000001280326955566124524921179               1   7
## p g1 a8 t9  0.00000197809600000000000595821603860402504 4               1   8
## p g1 a9 t10 0.04970449999999999868283140358471428044140               1   9
##             time Age Time
## p g1 a0 t1     1   0    0
## p g1 a1 t2     2   1    1
## p g1 a2 t3     3   2    2
## p g1 a3 t4     4   3    3
## p g1 a4 t5     5   4    4
## p g1 a5 t6     6   5    5
## p g1 a6 t7     7   6    6
## p g1 a7 t8     8   7    7
## p g1 a8 t9     9   8    8
## p g1 a9 t10   10   9    9
```

```
p.table
```

```
##             all.diff.index par.index     estimate           se           lcl
## p g1 a0 t1              10         2 4.410640e-02 2.364070e-02  1.514140e-02
## p g1 a1 t2              11         3 3.632856e-33 0.000000e+00  3.632856e-33
## p g1 a2 t3              12         4 1.464522e-31 0.000000e+00  1.464522e-31
## p g1 a3 t4              13         5 4.940680e-36 0.000000e+00  4.940680e-36
## p g1 a4 t5              14         6 3.686661e-35 8.376803e-27 -1.641853e-26
## p g1 a5 t6              15         7 5.040720e-02 2.616480e-02  1.785810e-02
## p g1 a6 t7              16         8 6.300920e-02 3.115180e-02  2.334890e-02
## p g1 a7 t8              17         9 1.512217e-01 6.523570e-02  6.173280e-02
## p g1 a8 t9              18        10 2.508653e-10 1.009105e-06 -1.977594e-06
## p g1 a9 t10             19        11 6.300800e-03 6.740500e-03  7.680969e-04
##                      ucl fixed note group age time Age Time
## p g1 a0 t1  1.216366e-01                1   0    1   0    0
## p g1 a1 t2  3.632856e-33                1   1    2   1    1
## p g1 a2 t3  1.464522e-31                1   2    3   2    2
## p g1 a3 t4  4.940680e-36                1   3    4   3    3
## p g1 a4 t5  1.641853e-26                1   4    5   4    4
## p g1 a5 t6  1.341770e-01                1   5    6   5    5
## p g1 a6 t7  1.590645e-01                1   6    7   6    6
## p g1 a7 t8  3.254402e-01                1   7    8   7    7
## p g1 a8 t9  1.978096e-06                1   8    9   8    8
## p g1 a9 t10 4.970450e-02                1   9   10   9    9
```

```
# Estimate for entry (constant):
pent.table = get.real(model.3.gen,"pent", se= TRUE)
```

```
## Warning in get.real(model.3.gen, "pent", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```
pent.table[c("estimate","se","lcl","ucl")][1,]
```

```
##                    estimate           se           lcl          ucl
## pent g1 a1 t2 7.857102e-10 6.002837e-06 -1.176477e-05 1.176635e-05
```

```
# Estimate for population size
N.table = get.real(model.3.gen,"N", se= TRUE)
```

```
## Warning in get.real(model.3.gen, "N", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```r
N.table[c("estimate","se","lcl","ucl")][1,]
```

```
##            estimate       se      lcl      ucl
## N g1 a0 t1 158.7074 61.61705 87.06034 352.4006
```

Put estimates for superpopulation size from each model in a table:

```r
N.table1 = get.real(model.1.gen,"N", se= TRUE)
N.table2 = get.real(model.2.gen,"N", se= TRUE)
```

```
## Warning in get.real(model.2.gen, "N", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```r
N.table3 = get.real(model.3.gen,"N", se= TRUE)
```

```
## Warning in get.real(model.3.gen, "N", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```r
N.table4 = get.real(model.4.gen,"N", se= TRUE)
N.table5 = get.real(model.5.gen,"N", se= TRUE)
```

```
## Warning in get.real(model.5.gen, "N", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```r
N.table6 = get.real(model.6.gen,"N", se= TRUE)
```

```
## Warning in get.real(model.6.gen, "N", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```r
N.table7 = get.real(model.7.gen,"N", se= TRUE)
```

```
## Warning in get.real(model.7.gen, "N", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```r
N.table8 = get.real(model.8.gen,"N", se= TRUE)
```

```
## Warning in get.real(model.8.gen, "N", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```r
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  89.45       0.00   158.71  61.62  87.06 352.40       238.06  92.43
## 2 Model 5 128.34      38.89   158.70   0.00 158.70 158.70       238.05   0.00
## 3 Model 7 128.34      38.89   158.71  61.62  87.06 352.40       238.06  92.43
## 4 Model 6 138.98      49.54   207.61  85.72 106.61 474.18       311.41 128.58
## 5 Model 4 141.06      51.61   195.17  81.03 100.76 449.45       292.76 121.54
## 6 Model 2 148.69      59.24   257.41 110.25 126.55 598.26       386.12 165.38
## 7 Model 1 157.19      67.74   244.95 101.90 122.97 557.78       367.43 152.84
## 8 Model 8 198.34     108.89   158.71   0.00 158.71 158.71       238.06   0.00
```

Much smaller estimate than given from the Photo ID CMR model, but probably to be expected given there is a lot less data.

Pradel Lambda

Start building model:

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

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.gen = mark(kw.pradel.proc.gen, kw.pradel.ddl.gen,
                          model.parameters = list(Phi = phi.pradel.ct, p = p.pradel.ct))

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

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

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

AIC values:

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

```r
AICc_pradel_values_gen <- c(summary(model.pradel.1.gen)$AICc,
                            summary(model.pradel.2.gen)$AICc,
                            summary(model.pradel.3.gen)$AICc,
                            summary(model.pradel.4.gen)$AICc)

AIC_pradel_table <- as.data.frame(cbind(AIC_pradel_models_gen, AICc_pradel_values_gen))
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 1 254.8798   84.52633
## 2 Pradel Model 2 170.3535    0.00000
## 3 Pradel Model 3 277.3055  106.95196
## 4 Pradel Model 4 205.5314   35.17792
```

Model 2 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.pradel.table = get.real(model.pradel.2.gen,"Phi", se = TRUE)
```

```
## Warning in get.real(model.pradel.2.gen, "Phi", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```r
phi.pradel.table[c("estimate","se","lcl","ucl")][1,] # Estimate for survival is 0.83
```

```
##               estimate se lcl ucl
## Phi g1 a0 t1         1  0   1   1
```

```r
# Estimate for recapture (time-dependent):
p.pradel.table = get.real(model.pradel.2.gen,"p", se= TRUE)
```

```
## Warning in get.real(model.pradel.2.gen, "p", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```r
p.pradel.table[c("estimate","se","lcl","ucl")][1:11,]
```

```
##                  estimate           se           lcl          ucl
## p g1 a0 t1  1.395623e-11 1.774759e-12  1.047770e-11 1.743475e-11
## p g1 a1 t2  7.684207e-25 0.000000e+00  7.684207e-25 7.684207e-25
## p g1 a2 t3  1.883432e-23 6.930744e-15 -1.358426e-14 1.358426e-14
## p g1 a3 t4  8.032089e-22 0.000000e+00  8.032089e-22 8.032089e-22
## p g1 a4 t5  1.078253e-20 2.416952e-12 -4.737226e-12 4.737226e-12
## p g1 a5 t6  1.064923e-04 4.659384e-05  4.517209e-05 2.510327e-04
## p g1 a6 t7  3.059000e-03 1.327500e-03  1.305600e-03 7.150200e-03
## p g1 a7 t8  1.875709e-01 7.216170e-02  8.363080e-02 3.687146e-01
## p g1 a8 t9  2.285386e-17 0.000000e+00  2.285386e-17 2.285386e-17
## p g1 a9 t10 8.485103e-01 1.323935e-01  4.265883e-01 9.768359e-01
## NA                     NA           NA            NA           NA
```

```r
# Estimate for lambda (constant):
lambda.pradel.table = get.real(model.pradel.2.gen,"Lambda", se= TRUE)
```

```
## Warning in get.real(model.pradel.2.gen, "Lambda", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```
lambda.pradel.table[c("estimate","se","lcl","ucl")][1,]
```

```
##                   estimate        se      lcl       ucl
## Lambda g1 a0 t1 0.0410238 0.0025694 0.036289 0.0463765
```

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

```
## $`Lambda Population Change`
##     estimate          se        lcl        ucl
## 1 0.04102382 0.002569416 0.03628896 0.04637646
## 2 0.04102382 0.002569416 0.03628896 0.04637646
## 3 0.04102382 0.002569416 0.03628896 0.04637646
## 4 0.04102382 0.002569416 0.03628896 0.04637646
## 5 0.04102382 0.002569416 0.03628896 0.04637646
## 6 0.04102382 0.002569416 0.03628896 0.04637646
## 7 0.04102382 0.002569416 0.03628896 0.04637646
## 8 0.04102382 0.002569416 0.03628896 0.04637646
## 9 0.04102382 0.002569416 0.03628896 0.04637646
##
## $`log(Lambda) Population Change`
##    estimate         se       lcl       ucl
## 1 -3.193602 0.06263229 -3.316362 -3.070843
## 2 -3.193602 0.06263229 -3.316362 -3.070843
## 3 -3.193602 0.06263229 -3.316362 -3.070843
## 4 -3.193602 0.06263229 -3.316362 -3.070843
## 5 -3.193602 0.06263229 -3.316362 -3.070843
## 6 -3.193602 0.06263229 -3.316362 -3.070843
## 7 -3.193602 0.06263229 -3.316362 -3.070843
## 8 -3.193602 0.06263229 -3.316362 -3.070843
## 9 -3.193602 0.06263229 -3.316362 -3.070843
```

```
# This didn't really work
```

Put it all in a table:

```
lamb.table1 = get.real(model.pradel.1.gen,"Lambda", se= TRUE)
lamb.table2 = get.real(model.pradel.2.gen,"Lambda", se= TRUE)
```

```
## Warning in get.real(model.pradel.2.gen, "Lambda", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```
lamb.table3 = get.real(model.pradel.3.gen,"Lambda", se= TRUE)
lamb.table4 = get.real(model.pradel.4.gen,"Lambda", se= TRUE)
```

```
## Warning in get.real(model.pradel.4.gen, "Lambda", se = TRUE):
## Improper V-C matrix for beta estimates. Some variances non-positive.
```

```
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
```

```
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 2 170.3535    0.00000 0.0410238 2.569400e-03 0.0362890 0.0463765
## 2 Pradel Model 4 205.5314   35.17792 0.0317133 9.631443e-07 0.0317114 0.0317152
## 3 Pradel Model 1 254.8798   84.52633 1.1264723 5.631480e-02 1.0213930 1.2423621
## 4 Pradel Model 3 277.3055  106.95196 1.1264723 5.629650e-02 1.0214254 1.2423225
```

This didn’t really work - maybe not enough data?