Comparing OTP and PIFG mark-recapture data sets

Background

Two large scale mark-recapture studies targeting opakapaka have been conducted in Hawaii, separated in time by ~20 years.

The integrated growth model approach in the current manuscript includes data from ~1980 - 1991. As this is already a significant period of time, and Joe has expressed concerns about the quality of the PIFG data, it seems prudent to compare PIFG data to the OTP data before we include it in any models.

Here we compare data collected from the two studies using the likelihood ratio test developed by Kimura (1980) to determine if the growth curves estimated from each data set siginficantly differ.

Analysis

first we need to set up our workspace

```
proj_dir = '/Volumes/GoogleDrive/My Drive/Weng Lab/Data/Bottomfish/Okamoto_Mark_Recapture'
if(!"Okamoto_Mark_Recapture" %in% strsplit(proj_dir, '/')[[1]]){
  proj_dir = file.path(getwd(), "Okamoto_Mark_Recapture")
}
data_dir = file.path(proj_dir, "data")

## Installing principle dependencies
library('FSA')
```

```
## ## FSA v0.8.20. See citation('FSA') if used in publication.
## ## Run fishR() for related website and fishR('IFAR') for related book.
```

Loading and cleaning data

Now we'll load in and clean up OTP and PIFG data

Loading OTP data set

```
#### OTP Data
mark_recapture_data = read.csv(file.path(data_dir, 'HO Mstr, temp (version 1).csv'), stringsAsFactors =

### Renaming data columns
colnames(mark_recapture_data) = c('tag_date', 'location', 'station', 'depth_f', 'species', 'previously_
                                  'recapture_2_date', 'recapture_2_location', 'recapture_2_station', 'r
                                  'recapture_3_date', 'recapture_3_location', 'recapture_3_station', 'r
                                  'recapture_4_date', 'recapture_4_location', 'recapture_4_station', 'r


## How many total fish do we have in the data set?
# dim(mark_recapture_data)[1] # 4245!

### Subsetting out only Opakapaka with tag IDs - That is, fish that were marked
```

```
mark_recapture_data = mark_recapture_data[mark_recapture_data$species == '1' & mark_recapture_data$tag_
# dim(mark_recapture_data)[1] # This gets you to the previously published 4179 tagged paka number from 

#### Adusting Data Classes
### Formatting Dates (Converting Characters to POSIXct)
mark_recapture_data$tag_date = as.POSIXct(mark_recapture_data$tag_date, format = "%Y-%m-%d")
mark_recapture_data$recapture_1_date = as.POSIXct(mark_recapture_data$recapture_1_date, format = "%Y-%m
mark_recapture_data$recapture_2_date = as.POSIXct(mark_recapture_data$recapture_2_date, format = "%Y-%m
mark_recapture_data$recapture_3_date = as.POSIXct(mark_recapture_data$recapture_3_date, format = "%Y-%m
mark_recapture_data$recapture_4_date = as.POSIXct(mark_recapture_data$recapture_4_date, format = "%Y-%m

### Formatting fork lengths
## Note: There are a couple fork lengths that have ?, *, or have no lengths recorded.
## I have no idea what these are but they're qualifiers and so I'm going to let them go to NA and get d
in_to_cm = 2.54
mark_recapture_data$fork_length_cm = as.numeric(mark_recapture_data$fork_length_in) * in_to_cm
```

```
## Warning: NAs introduced by coercion
```

```
mark_recapture_data$recapture_1_fork_length_cm = as.numeric(mark_recapture_data$recapture_1_fork_length
```

```
## Warning: NAs introduced by coercion
```

```
mark_recapture_data$recapture_2_fork_length_cm = as.numeric(mark_recapture_data$recapture_2_fork_length
```

```
## Warning: NAs introduced by coercion
```

```
mark_recapture_data$recapture_3_fork_length_cm = as.numeric(mark_recapture_data$recapture_3_fork_length
mark_recapture_data$recapture_4_fork_length_cm = as.numeric(mark_recapture_data$recapture_4_fork_length

#### Now we want to format a table with lm (length at marking), lr (length at recapture), and dt (elaps
### Note: If a fish was recaptured multiple times, there is a single entry for that individual corrospo
otp_data = data.frame(stringsAsFactors = FALSE)
for(i in 1:length(mark_recapture_data$tag_id)){
  if(!is.na(mark_recapture_data$recapture_4_fork_length_cm[i])){
    otp_data = rbind(otp_data, data.frame('tag_id' = mark_recapture_data$tag_id[i], 'Lm' = mark_recaptu
  }else if(!is.na(mark_recapture_data$recapture_3_fork_length_cm[i])){
    otp_data = rbind(otp_data, data.frame('tag_id' = mark_recapture_data$tag_id[i], 'Lm' = mark_recaptu
  }else if(!is.na(mark_recapture_data$recapture_2_fork_length_cm[i])){
    otp_data = rbind(otp_data, data.frame('tag_id' = mark_recapture_data$tag_id[i], 'Lm' = mark_recaptu
  }else if(!is.na(mark_recapture_data$recapture_1_fork_length_cm[i])){
    otp_data = rbind(otp_data, data.frame('tag_id' = mark_recapture_data$tag_id[i], 'Lm' = mark_recaptu
  }
}
otp_data$dt = abs(difftime(otp_data$tm, otp_data$tr, units = "days"))    ## Converting dt from days to 
otp_data$dt = as.numeric(otp_data$dt) / 365 # Converting to years
### Constructing derived variable dl (change in growth)
otp_data$dL = otp_data$Lr - otp_data$Lm
### Removing any fish that have a NA value for dL or dt (There is a single fish which had no tagging le
# length(which(is.na(otp_data$dL))) # 1
# length(which(is.na(otp_data$dt))) # 7
otp_data = otp_data[!is.na(otp_data$dL) & !is.na(otp_data$dt), ]

#### Removing data with recording errors in length and time
# otp_data = subset(otp_data, dL > 0)
# length(which(otp_data$dt <= 60/365)) #46
otp_data = subset(otp_data, dt >= 60/365)
tagdat = as.matrix(data.frame('L1' = otp_data$Lm, "L2" = otp_data$Lr, " " = rep(0, length(otp_data$Lr))
```

Loading in PIFG tagging datasets

```
#### Creating Second tagging dataset from PIFG data
pifg20072013 = read.csv(file.path(data_dir, 'PIFG 2007-2013.csv'), stringsAsFactors = FALSE)
pifg20072013$rel_date = as.POSIXct(pifg20072013$rel_date, format = "%m/%d/%Y")
pifg20072013$recap_date = as.POSIXct(pifg20072013$recap_date, format = "%m/%d/%Y")
pifg20072013$dt = difftime(pifg20072013$recap_date, pifg20072013$rel_date)

### 2014-2015 data
pifg20142015 = read.csv(file.path(data_dir, 'PIFG 2014-2015.csv'), stringsAsFactors = FALSE)
pifg20142015$rel_date = as.POSIXct(pifg20142015$rel_date, format = "%m/%d/%Y")
pifg20142015$recap_date = as.POSIXct(pifg20142015$recap_date, format = "%m/%d/%Y")
pifg20142015$rel_FL[pifg20142015$Units == 'in'] = pifg20142015$rel_FL[pifg20142015$Units == 'in'] * in_
pifg20142015$recap_FL[pifg20142015$Units == 'in'] = pifg20142015$recap_FL[pifg20142015$Units == 'in'] *
pifg20142015$dt = difftime(pifg20142015$recap_date, pifg20142015$rel_date)

pifg_data = data.frame('Lm' = c(pifg20072013$rel_FL, pifg20142015$rel_FL), 'Lr' = c(pifg20072013$recap_

pifg_data$dL = pifg_data$Lr - pifg_data$Lm
```

Now we have two data sets, otp_data and pifg_data with the following variables in common:

Lm: length at marking in cm (class = numeric)

Lr: length at time of last recapture in cm (class = numeric)

dL: The difference between Lr and Lm

dt: time elapsed between marking and recapture in years (or fractions there of) (class = numeric)

We are now prepared to compare growth curves between these data sets. Because we're only trying to determine if the two data sets produce similar parameter estimates and not accuracty of those estimates, we'll use faben's method, despite it's shortcomings in estimation

Model fitting procedure

The general procedure for testing for coinident curves was proposed by Kaimura in 1980. The genera procedure is as follows:

1. For each data set i, fit a curve and calculate the sum of squared residuals, RSSi, and an associated degree of freedom, DFi
2. The resultant RSS and DF for each curve are added together
3. Both data sets are pooled and a new curve is fitted to the combined data. The total or poosled RSSp and DFp are calculated.
4. Using these statistics, an F statistic is calculated and compared to the critical value from an F table.

1. Fit a curve and calculate residual sum of squares and degrees of freedom for each data set

We use Faben's model to fit simple curves:

Lr = Lm + (Linf - Lm)(1 - e^(-k*dt))

We'll select starting values for each data set such that Linf is the maximum length at recapture for the data set and K is the average change in fork length (cm) divided by the time at liberty (years)

OTP Data

```
l_inf_init = max(otp_data$Lr)
k_init = mean((otp_data$Lr - otp_data$Lm) / otp_data$dt)

 otp_fit = nls((dL ~ (l.inf - Lm) * (1-exp((-K * dt)))), data = otp_data,
                      start = list(K = k_init, l.inf = l_inf_init))
    summary(otp_fit)
```

```
##
## Formula: dL ~ (l.inf - Lm) * (1 - exp((-K * dt)))
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## K      0.23731    0.01713   13.85   <2e-16 ***
## l.inf 65.92613    1.54834   42.58   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.673 on 385 degrees of freedom
##
## Number of iterations to convergence: 8
## Achieved convergence tolerance: 6.489e-06
```

```
    rss_otp = sum(summary(otp_fit)$residuals^2)
    df_otp = summary(otp_fit)$df[2]
```

PIFG Data

```
l_inf_init = max(pifg_data$Lr)
k_init = mean((pifg_data$Lr - pifg_data$Lm) / pifg_data$dt)

  pifg_fit = nls((dL ~ (l.inf - Lm) * (1-exp((-K * dt)))), data = pifg_data,
                      start = list(K = k_init, l.inf = l_inf_init))
    summary(pifg_fit)
```

```
##
## Formula: dL ~ (l.inf - Lm) * (1 - exp((-K * dt)))
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## K      0.38410    0.04803   7.998 3.16e-13 ***
## l.inf 55.61464    1.41287  39.363  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 3.105 on 150 degrees of freedom
##
## Number of iterations to convergence: 6
## Achieved convergence tolerance: 9.187e-06
```

```
    rss_pifg = sum(summary(pifg_fit)$residuals^2)
    df_pifg = summary(pifg_fit)$df[2]
```

2. The resultant RSS and DF for each curve are added together

```
rss_indv = rss_otp + rss_pifg
df_indv  = df_otp + df_pifg

print(paste('Additive Residual Sum of Squares from Individual Fits:', round(rss_indv, digits = 3)))
```

```
## [1] "Additive Residual Sum of Squares from Individual Fits: 6640.761"
```

```
print(paste('Additive Degrees of Freedom from Individual Fits:', round(df_indv, digits = 3)))
```

```
## [1] "Additive Degrees of Freedom from Individual Fits: 535"
```

3. Both datasets are pooled, a new vonB curve is fitted to the combined data, and the total or poosled RSS and DF are calculated.

```
pooled_data = data.frame('Lm' = c(otp_data$Lm, pifg_data$Lm), 'Lr' = c(otp_data$Lr, pifg_data$Lr), 'dt'

l_inf_init = max(pooled_data$Lr)
k_init = mean((pooled_data$Lr - pooled_data$Lm) / pooled_data$dt)

   pooled_fit = nls((dL ~ (l.inf - Lm) * (1-exp((-K * dt)))), data = pooled_data,
                        start = list(K = k_init, l.inf = l_inf_init))
    summary(pooled_fit)
```

```
##
## Formula: dL ~ (l.inf - Lm) * (1 - exp((-K * dt)))
##
## Parameters:
##       Estimate Std. Error t value Pr(>|t|)
## K      0.26680    0.01661   16.06   <2e-16 ***
## l.inf 62.93399    1.15517   54.48   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.589 on 537 degrees of freedom
##
## Number of iterations to convergence: 8
## Achieved convergence tolerance: 4.73e-06
```

```
    rss_pooled = sum(summary(pooled_fit)$residuals^2)
    df_pooled = summary(pooled_fit)$df[2]

print(paste('Residual Sum of Squares for Pooled Fits:', round(rss_pooled, digits = 3)))
```

```
## [1] "Residual Sum of Squares for Pooled Fits: 6915.335"
```

```
print(paste('Degrees of Freedom for Pooled Fits:', round(df_pooled, digits = 3)))
```

```
## [1] "Degrees of Freedom for Pooled Fits: 537"
```

4. Calculating an F Statistic

Our F value is calculated with the following formula F_value = (rss_pooled - rss_indv) / (df_pooled - df_indv) / (rss_indv / df_indv)

```
F_value = (rss_pooled - rss_indv) / (df_pooled - df_indv) / (rss_indv / df_indv)
print(paste('Our calculated F value is:', round(F_value, digits = 3)))
```

```
## [1] "Our calculated F value is: 11.06"
```

We now compare the F value we've just calculated to the critical value from an F distribution with the first degrees of freedom equal to the sum of the degrees of freedom from PIFG and OTP data sets fit independently of one another and the second degrees of freedom equal to the degrees of freedom from the model fit with both data sets pooled

If the F we've calculated is greater than the critical value from the F distribution at alpha (0.95), then curves differ significantly.

```
print(paste('The critical value at alpha = 0.95 when df1 =', df_indv, 'and df2 =', df_pooled, 'is', rou
```

```
## [1] "The critical value at alpha = 0.95 when df1 = 535 and df2 = 537 is 1.153"
```

```
if(F_value > qf(p = .95, df1 = df_pooled, df2 = df_indv)){
  print("Kaimura's likelihood ratio test indicates that the two curves are NOT coincident")
  print(paste(round(F_value, digits = 3), '>',  round(qf(p = .95, df1 = df_pooled, df2 = df_indv), digi
} else {
  print("Kaimura's likelihood ratio test indicates that the two curves ARE coincident")
  print(paste(round(F_value, digits = 3), '<=',  round(qf(p = .95, df1 = df_pooled, df2 = df_indv), dig
}
```

```
## [1] "Kaimura's likelihood ratio test indicates that the two curves are NOT coincident"
## [1] "11.06 > 1.153"
```

Conclusion

After comparing the fit of the two growth curves and the resulting F statistic, we conclude that the models are not coincident. From this result, the decision to ommit the mark recapture data collected by PIFG is supported.