Week 3 Length Frequency Analysis

Nesslage

2022-09-15

Load libraries and clear console

```
library(TropFishR)
graphics.off()
rm(list=ls(all=TRUE))
```

Step 1: Load and examine dataset

In this exercise you will use ELEFAN to estimate growth parameters with the help of the package TropFishR. You will examine a length frequency data set for the mollusc *Abra alba* sampled in Kiel Bay in the Baltic Sea. Start by loading and inspecting your data.

At what frequency are the LF samples collected and across what length time period?

```
data("alba")
alba
```

Step 2: Restructure your data

Determine the number of bins to include in your moving average. Original ELEFAN (available in a program called FiSAT) used 5 bins. Thus in TropFishR the default value is 5, but you can specify an alternative value using the "MA" argument. Try a few different values before making a final decision.

Rule of thumb: MA should approximate the number of bins spanning the width covered by the smallest (i.e. youngest) cohort. Hint: go back and look at the raw data and find the sample with LF most likely representing the youngest cohort. How many bins does it cover? Also, note that MA must be odd with +/- 2 bins each side of middle bin. Note: when working with your own data, you will need to balance the decision about how wide to make your length bins with your decision about how many bins to include in your moving average.

How does the value of MA affect the number of positive and negative bins?

Plot your final restructured data.

What MA value did you decide upon and why? Can you visualize vonB growth across sample periods (focusing on relationship between black bars across samples)?

```
alba <- lfqRestructure(alba, MA=7)
plot(alba)
```

Step 3: Identify appropriate limits for vonB parameters to constrain the search space in ELEFAN_GA

TropFishR offers two optimization approaches for fitting growth curves: (1) Generalized Simulated Annealing (ELEFAN_SA), and (2) Genetic Algorithm (ELEFAN_GA). Taylor & Mildenberger (2017) demonstrate how both optimization routines generate similar results for ELEFAN, so just focus for now on ELEFAN_GA in this exercise.

These optimization routines need to be given some information about the potential range of values each parameter might take to constrain the search. That means you need to provide the lower and upper bounds of Linf and K as an argument when calling the function.

Hint for Linf: Try estimating Linf using a length-based catch curve and use the confidence interval to determine the lower and upper bounds.

Which catch curve did you use to get an estimate of Linf and why?

Hint for K: In general, small, short-lived species will have higher values of K (e.g. > 1.5), whereas large, longer-lived species will have lower values (< 1.0). In real life you would consult the literature for published values of K for this or related species, if available.

```
PW <- powell_wetherall(alba, catch_columns = 1:7)
```

Step 4: Compare the Linf and K values you generated with Response Surface Analysis (RSA)

Use the traditional ELEFAN function in TropFishR to produce a visualization of the Rn scoring values across a range of discrete Linf and K combinations (set using Linf_range and K_range arguments). ID the Linf and K values that generate the maximum value of Rn. Compare these with the constraints you've selected for Linf and K.

```
alba2 <- ELEFAN(
  lfq = alba,  MA = 7,
  Linf_range = seq(7, 20, length.out = 30),
  K_range = exp(seq(log(0.1),log(4), length.out = 30)),
  method = "cross",
  cross.date = alba$dates[3],
  cross.midLength = alba$midLengths[5],
  contour = TRUE, add.values = FALSE,
  hide.progressbar = TRUE # change to 'TRUE' to follow algorithm's progression
)
points(alba2$par["Linf"], alba2$par["K"], pch="*", cex=2, col=2)

unlist(alba2$par)
alba2$Rn_max
```

Make a final decision on what values to use as limits for Linf and K in your analysis. Make sure you give the optimizer sufficient search space (i.e., don't make the limits too narrow or you might exclude the best values!).

What values for limits for Linf and K did you select and why?

Step 5: Estimate vonB parameters using ELEFAN_GA

Run ELEFAN_GA using your values for constraining Linf and K from Step 4. Do this by changing the Linf and K values in the low_par and up_par arguments.

Generate growth curve parameter value estimates and plot those curves on your restructured data.

```r
set.seed(1)
alba4 <- ELEFAN_GA(
  lfq = alba,
  seasonalised = FALSE,
  low_par = list(Linf=PW$confidenceInt_Linf[1], K=1, t_anchor=0, ts=0, C=0),
  up_par = list(Linf=PW$confidenceInt_Linf[2], K=4, t_anchor=1, ts=1, C=1),
  popSize = 60,
  pmutation = 0.2,
  maxiter = 100,
  run = 20,
  MA = 7,
  plot.score = TRUE,
  monitor = FALSE,
  parallel = FALSE
)

unlist(alba4$par)
alba4$Rn_max
plot(alba4)
```

Inspect the plot of the change in Rn (fitness value) as the optimization routine progresses (generation). Notice how quickly (or not) it settles on a max Rn.

How does max Rn in ELEFAN_GA compare with that generated by RSA? Why do they differ?

What are your final ELEFAN_GA-based estimates of growth curve parameter Linf and how does that compare with your catch curve estimate?

Do you think your final growth curve fits the data well?