

Hierarchical run example

LB

08 April 2019

A short vignette showing how to run the hierarchical LFEM model on an example dataset. The example dataset in this case is a cleaned up version of white-bellied anglerfish data from the surveys that cover ICES subarea VII and divisions VIII a,b,d stock.

This method allows multiple surveys from different times of year.

Loading data and formatting

WD should be “LFEM/Vignettes” folder in order to locate .cpp files later

```
getwd()

## [1] "C:/Users/LukeB/Documents/LFEM/Vignettes"

#install.packages(c("Matrix", "data.table", "plyr", "reshape", "TMB", "ggplot2"))
library(TMB)
library(plyr)
library(reshape)
library(ggplot2)
library(data.table)
load("../data/lfdat_MON.RData")
```

The dataframe should be formatted as below for the function, with four columns: Survey, Year, Length and RF (raising factor). The data has been aggregated by year, survey and length so as to compact the dataframe for speed. Observed log-likelihood is the same if individual fish lengths for each haul were used.

```
##      Survey Year Length RF
## 1 SP-PORC 2005      3  1
## 2 IE-IGFS 2007      4  1
## 3 EVHOE 2011      4  1
## 4 IE-IGFS 2012      4  1
## 5 SP-PORC 2004      5  1
## 6 IE-IGFS 2005      5  1
```

Objective functions have already been compiled so no need to do this again but here is the code to do it. Also set root directory for cpp/dll files so the function can find the .dll files.

```
dllroot<-paste0(dirname(getwd()), '/tmb/')

compile(paste0(dllroot, "hier_cL_CSD.cpp"))
compile(paste0(dllroot, "hier_cL_LSD.cpp"))
compile(paste0(dllroot, "hier_cL_CSD_OBSLL.cpp"))
compile(paste0(dllroot, "hier_cL_LSD_OBSLL.cpp"))

compile(paste0(dllroot, "hier_ck_CSD.cpp"))
compile(paste0(dllroot, "hier_ck_LSD.cpp"))
compile(paste0(dllroot, "hier_ck_CSD_OBSLL.cpp"))
compile(paste0(dllroot, "hier_ck_LSD_OBSLL.cpp"))

compile(paste0(dllroot, "hier_yk_CSD.cpp"))
```

```
compile(paste0(dllroot,"hier_yk_LSD.cpp"))
compile(paste0(dllroot,"hier_yk_CSD_OBSLL.cpp"))
compile(paste0(dllroot,"hier_yk_LSD_OBSLL.cpp"))
```

Source the function

```
source("../R/hier_fun.R")
```

Function arguments and starting parameters

Survey information

Values should be entered in alphabetical order of surveys in each case.

How many years of data in each survey? What year does data for each survey start?

```
no.years<-c(14,14,14)
year0<-c(2003,2003,2003)
```

age1 is the assumed age of the first component in each of the surveys. In this case we assume that for the first two surveys alphabetically (EVHOE and IE-IGFS) that the first component observed in the length frequency data is approximately 0.875 years old, using the common assumption fish are born on the 1st of Jan (i.e. the midpoint of the fourth quarter of the year when these surveys are conducted). SP-PORC is mainly conducted over september so we set the age1 at 0.73

```
age1<-c(0.875,0.875,0.73)
```

Starting parameters

- L is the mean of the final component
- l is the mean of the first component
- $k.reparam$ is the starting growth parameter
- sigma.start is the starting standard deviation parameter(s)
- SD.type =
 - 3 -> linear SD relative to means, needs two sigma.start parameters e.g. c(5,10)
 - 4 -> constant SD
- RE.type =
 - 1 -> cohort random effects on l and L
 - 2 -> cohort random effects on l and k
 - 3 -> cohort random effects on l and yearly random effect on k
- fix.RESID =
 - TRUE -> standard deviations for the random effects are fixed to input values. These need to be specified in the function as sdl and sdL or sdk. This gives models more stability when data is lacking, but still provides enough flexibility (see haddock example of paper).
 - FALSE -> Estimates sd of random effects

#Run the function If rel.tolerance is set at 1e-8 as is standard then this will take some time to converge

#Example where sd of RE is estimated

```
hier_test<- hier.LFEM(year0=year0,no.years=no.years,age1=age1,L=130,l=(16),  
k.reparam=0.83,sigma.start=c(6,10),No.comp=9,SD.type=3,RE.type=1,fix.RESD=F,Lengths=lfdat,  
niter=10000,rel.tolerance=1e-1,dllroot = dllroot)
```

#Example where sd of RE is fixed by the user

```
hier_test_fixedRESD<- hier.LFEM(year0=year0,no.years=no.years,age1=age1,L=130,l=(16),  
k.reparam=0.83,sigma.start=c(6,10),No.comp=9,SD.type=3,RE.type=1,fix.RESD=T,Lengths=lfdat,  
niter=10000,rel.tolerance=1e-1,dllroot = dllroot,sd1=-2,sdL=-2)
```

Load the test object if you haven't run the model

```
load("../data/hier_test.RData")
```

Results

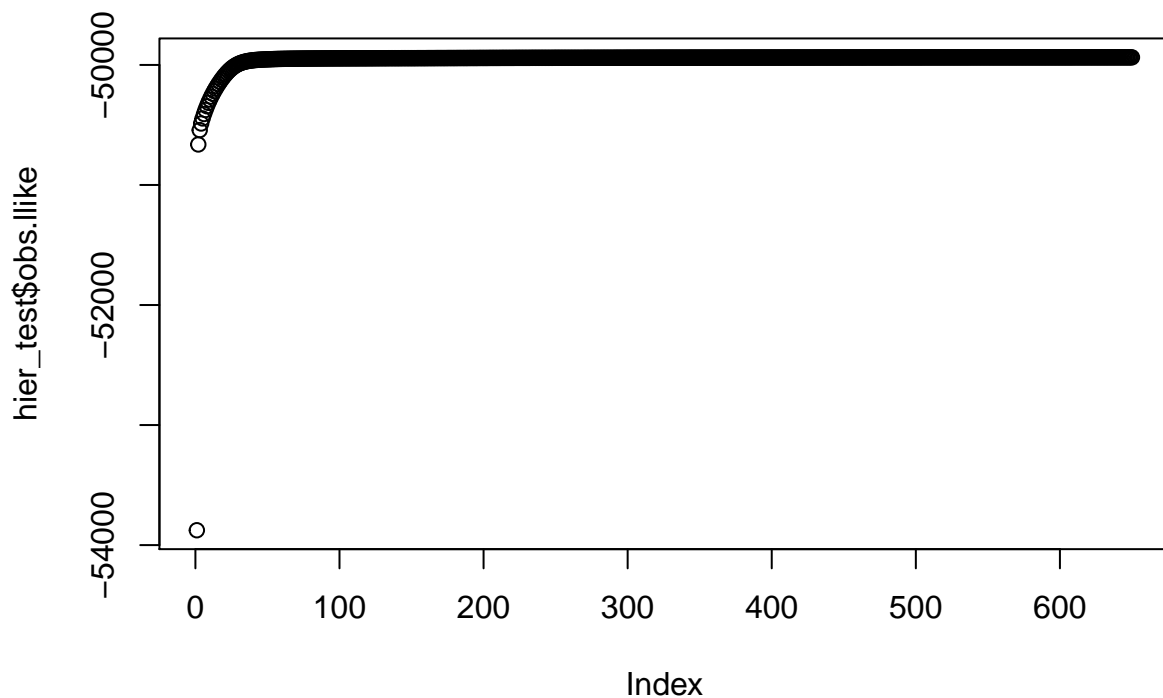
There are many results from the model

```
str(hier_test)

## List of 26
## $ obs.llike          : num [1:650] -53876 -50663 -50543 -50488 -50446 ...
## $ Mu.obs.years       : num [1:14, 1:9, 1:3] 17.6 16.5 14.9 14.4 16.6 ...
## $ Mu.all.years       : num [1:30, 1:9, 1:3] 16.4 16.4 16.4 16.4 16.4 ...
## $ Sd.obs.years       : num [1:14, 1:9, 1:3] 4.04 3.96 3.84 3.8 3.97 ...
## $ Sd.all.years       : num [1:30, 1:9, 1:3] 3.95 3.95 3.95 3.95 3.95 ...
## $ Lambda             : num [1:14, 1:9, 1:3] 0.279 0.637 0.171 0.306 0.285 ...
## $ k.reparam          : Named num 0.915
##   .. attr(*, "names")= chr "logit_k_reparam"
## $ l                   : Named num 16.4
##   .. attr(*, "names")= chr "log_l_mu"
## $ L                   : Named num 118
##   .. attr(*, "names")= chr "log_L_mu"
## $ sd.l                : Named num 0.0721
##   .. attr(*, "names")= chr "raw_sd1"
## $ sd.L                : Named num 0.0647
##   .. attr(*, "names")= chr "raw_sdL"
## $ Rho                 : Named num -0.691
##   .. attr(*, "names")= chr "raw_rho"
## $ RE.mat              : num [1:14, 1:2] 0.06989 0.00645 -0.09629 -0.12958 0.01445 ...
## $ l.par.vec           : num [1:14] 17.6 16.5 14.9 14.4 16.6 ...
## $ L.par.vec           : num [1:14] 114 115 129 133 125 ...
## $ sigma               : num [1:2] 3.95 11.65
## $ K                   : Named num 0.0889
##   .. attr(*, "names")= chr "logit_k_reparam"
## $ Linf.overall        : Named num 215
##   .. attr(*, "names")= chr "log_L_mu"
## $ tzero.overall       : Named num -0.0146
##   .. attr(*, "names")= chr "logit_k_reparam"
## $ Linf.cohort         : num [1:22] 216 216 216 216 216 ...
## $ tzero.cohort        : num [1:22] -0.015 -0.015 -0.015 -0.015 -0.015 ...
## $ Lambda.params       : num 336
## $ sample.size         : num 12060
## $ age1                : num [1:3] 0.875 0.875 0.73
## $ Final.Estimate.Error: num [1:71, 1:2] 2.8 4.77 -2.62 -2.72 -1.65 ...
##   .. attr(*, "dimnames")=List of 2
##     .. $ : chr [1:71] "log_l_mu" "log_L_mu" "raw_sd1" "raw_sdL" ...
##     .. $ : chr [1:2] "Estimate" "Std. Error"
## $ Entropy              : num 1217
```

First thing to check is the convergence

```
plot(hier_test$obs.llike)
```



Then explore the rest of the results

Mu's and Sd's are now matrices rather than a vector for each survey. If we take a look at Mu.all.years this displays all component means including those that are used in the unobserved years. In this case the model assumes nine components(m), so we have (m-1) unobserved years before and after our survey data. Adding these in allows us to model cohorts. l values for unobserved years are given the average, as are their corresponding L . This gives the hierarchical model more stability but also allows us to focus on the variability in the cohorts we observe from their very first year (i.e. the first year of our survey data).

```
hier_test$Mu.all.years[, ,1]
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	16.40885	NA	NA	NA	NA	NA	NA
[2,]	16.40885	33.36121	NA	NA	NA	NA	NA
[3,]	16.40885	33.36121	48.87134	NA	NA	NA	NA
[4,]	16.40885	33.36121	48.87134	63.06195	NA	NA	NA
[5,]	16.40885	33.36121	48.87134	63.06195	76.04528	NA	NA
[6,]	16.40885	33.36121	48.87134	63.06195	76.04528	87.92406	NA
[7,]	16.40885	33.36121	48.87134	63.06195	76.04528	87.92406	98.79224
[8,]	16.40885	33.36121	48.87134	63.06195	76.04528	87.92406	98.79224
[9,]	17.55225	33.36121	48.87134	63.06195	76.04528	87.92406	98.79224
[10,]	16.47330	33.68552	48.87134	63.06195	76.04528	87.92406	98.79224
[11,]	14.86485	33.00703	48.44624	63.06195	76.04528	87.92406	98.79224
[12,]	14.37815	34.00537	48.13415	61.95119	76.04528	87.92406	98.79224
[13,]	16.60571	34.13554	51.51750	61.97433	74.30720	87.92406	98.79224
[14,]	17.28760	34.64750	52.21206	67.53978	74.63705	85.61202	98.79224
[15,]	17.59978	33.86577	51.15437	68.75072	82.19896	86.22248	95.95508
[16,]	16.86383	34.09310	49.03355	66.25691	83.88234	95.61101	96.82228
[17,]	14.60102	32.59601	49.18324	62.91093	80.07460	97.72664	107.88202
[18,]	17.48241	32.78112	46.98977	62.98958	75.60768	92.71675	110.39313
[19,]	15.56920	33.49085	49.41455	60.15898	75.62134	87.22426	104.28336
[20,]	16.78517	32.68935	48.13737	64.63288	72.20782	87.17845	97.85255
[21,]	17.81319	31.08733	48.35300	61.53783	78.55651	83.23159	97.75234
[22,]	15.84743	33.85515	44.17274	62.68406	73.79825	91.29558	93.31752
[23,]	NA	33.11731	48.53233	56.14490	75.79591	85.01560	102.95087
[24,]	NA	NA	48.91795	61.96084	67.09852	87.79225	95.27863
[25,]	NA	NA	NA	63.37435	74.24692	77.12026	98.76801
[26,]	NA	NA	NA	NA	76.60087	85.48775	86.28940
[27,]	NA	NA	NA	NA	NA	88.70213	95.77227
[28,]	NA	NA	NA	NA	NA	NA	99.77388
[29,]	NA	NA	NA	NA	NA	NA	NA
[30,]	NA	NA	NA	NA	NA	NA	NA
	[,8]	[,9]					
[1,]	NA	NA					
[2,]	NA	NA					
[3,]	NA	NA					
[4,]	NA	NA					
[5,]	NA	NA					
[6,]	NA	NA					
[7,]	NA	NA					
[8,]	108.73580	NA					
[9,]	108.73580	117.8334					
[10,]	108.73580	117.8334					
[11,]	108.73580	117.8334					
[12,]	108.73580	117.8334					
[13,]	108.73580	117.8334					
[14,]	108.73580	117.8334					
[15,]	108.73580	117.8334					
[16,]	105.41819	117.8334					
[17,]	106.52030	114.0762					
[18,]	119.10907	115.3933					
[19,]	121.98201	129.3810					
[20,]	114.86594	132.5850					

```
[21,] 107.57663 124.5482
[22,] 107.42665 116.4734
[23,] 102.54538 116.2779
[24,] 113.61458 110.9882
[25,] 104.66853 123.3711
[26,] 108.81000 113.2596
[27,]  94.67847 117.9977
[28,] 105.18183 102.3538
[29,] 109.90369 113.7909
[30,]          NA 119.1717
```

We can plot the results with same function as we did the basic example.

```
source("../R/plot_lfem.R")
library(grid)
library(gridBase)
```

```
## Warning: package 'gridBase' was built under R version 3.5.2
```

```
plot.lfem(model=hier_test,Lengths=lfdat,Survey.num=2,xlimit=140)
```

