Summarise ABM

C.Marsh

1 November 2019

Introduction

This is a document that demonstrates how to use the ibm R-library to interact with the agent based model (ABM) output, and show how you can use to tweak models and investigate the output. The ABM model is a farily simple three area model, where there is movement between areas, and a single fishery (this is easily extended). We show how users can incorporate time and spatially varying growth natural mortality and how that effects data that are ultimately used in Stock assessment models.

Prepare libraries

```
library(ibm)
library(ggplot2)
library(reshape2)
```

Investigating burn-in period for initial age-structure

Depending on the complexity (years, time-steps, number of cells, number of agents) of your model the model can become computationally slow. So there are stratigies you can imploy to minimise this computational burden. This model is not too complex (three areas, 13 years, two time steps) but still takes approx 2mins on my machine to run. Looking at number of years to run the initialisation phase for that reach an equilibrium age-structure.

```
## read in base model output
ibm_20 = extract.run("../BuildingIBM/run_20.log")
ibm_50 = extract.run("../BuildingIBM/run_50.log")
ibm_100 = extract.run("../BuildingIBM/run_100.log")
ibm_more_agents = extract.run("../BuildingIBM/run_more_agents.out")

# look at time difference
ibm_20$model_run_time
```

```
## [1] "11 seconds"
```

```
ibm_50$model_run_time
```

```
## [1] "26 seconds"
```

```
ibm_100$model_run_time
```

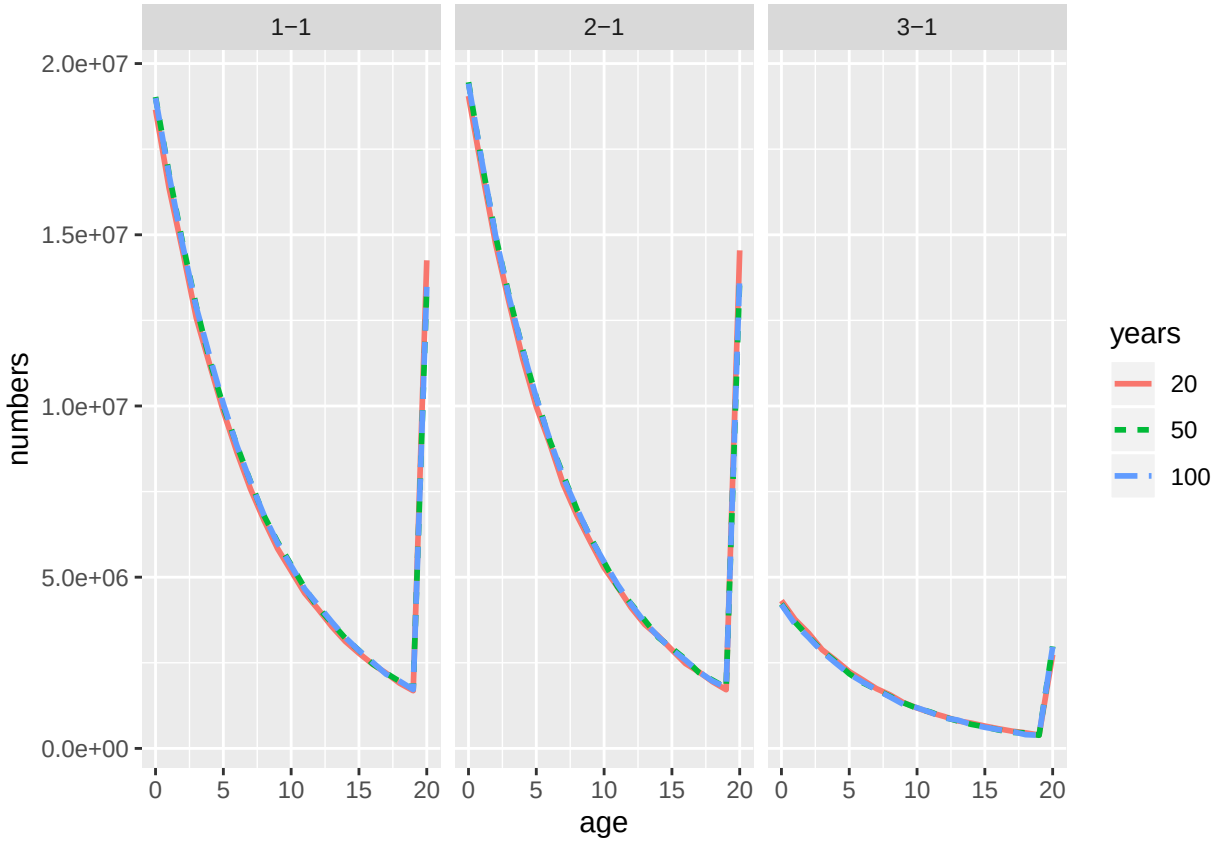
```
## [1] "50 seconds"
```

```
ibm_more_agents$model_run_time # run with 50 initialisaiton years
```

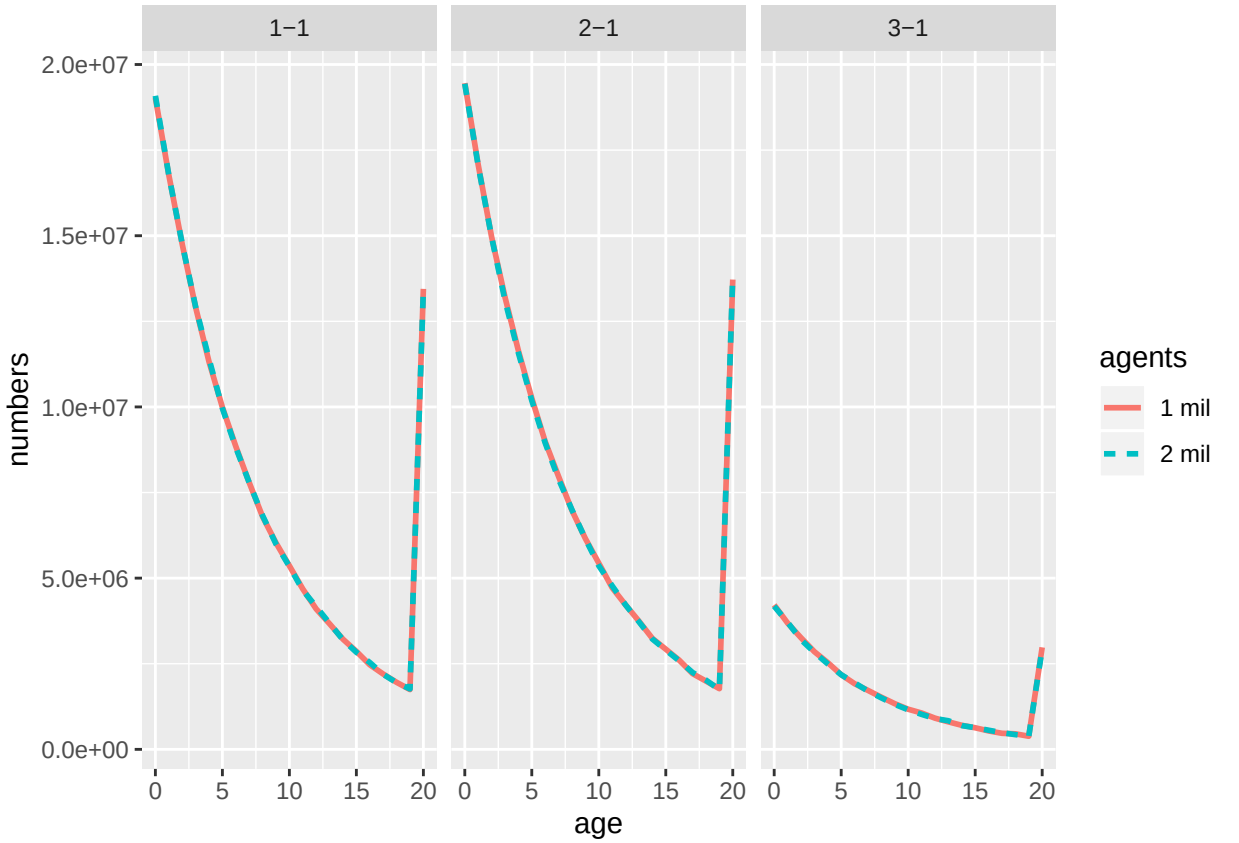
```
## [1] "53 seconds"
```

```
# if they generate the same initial age-structure we want the fastest lets see
init_20 = melt(as.matrix(ibm_20$init_2$values))
init_20$years = 20
```

```
init_50 = melt(as.matrix(ibm_50$init_2$values))
init_50$years = 50
init_50_extra = melt(as.matrix(ibm_more_agents$init_2$values))
init_50_extra$years = 50
init_100 = melt(as.matrix(ibm_100$init_2$values))
init_100$years = 100
all_init = rbind(init_20, init_50, init_100)
colnames(all_init) = c("area", "age", "numbers","years")
all_init$years = as.factor(all_init$years)
ggplot(data = all_init, aes(x = age, y = numbers, linetype = years, col = years)) +
  geom_line(size = 1) +
  facet_wrap( ~ area, nrow = 1)
```




```
## not much difference will settle with using 50 years.
## 20 years seems to shorter a time.
init_50$agents = "1 mil"
init_50_extra$agents = "2 mil"
all_agents = rbind(init_50, init_50_extra)
colnames(all_agents) = c("area", "age", "numbers","years","agents")
all_agents$years = as.factor(all_agents$years)
ggplot(data = all_agents, aes(x = age, y = numbers, linetype = agents, col = agents)) +
  geom_line(size = 1) +
  facet_wrap( ~ area, nrow = 1)
```



So in this instance we are going to choose to run with 1 million initial agents and a 40 year equilibrium phase.

look at the process model

```
## read in base model output
ibm_base = extract.run("../base_ibm/run.log")
# how long did the model take?
ibm_base$model_run_time
```

```
## [1] "40 seconds"
```

```
# are there any warnings
ibm_base$warnings_encounted$warnings_found
```

```
## [1] 78
```

```
# you will want to look at these
# number Agent = this many individuals
ibm_base$model_attributes$Recruitment_EN
```

```
## [1] 402.403
```

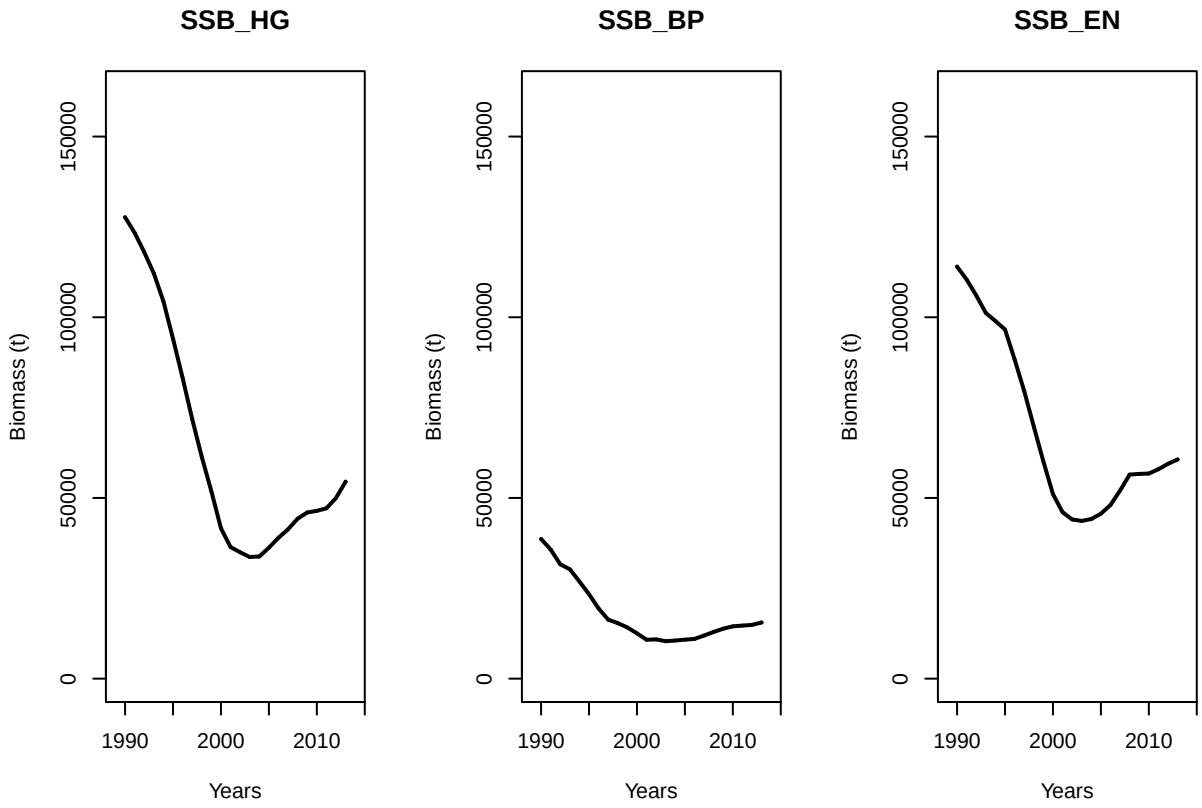
```
ibm_base$model_attributes$Recruitment_HG
```

```
## [1] 369.983
```

```
ibm_base$model_attributes$Recruitment_BP
```

```
## [1] 307.526
```

```
# plot SSB's
plot.derived_quantities(ibm_base, report_label = "derived_quants", lwd = 2, pch = 16, cex = 0.4)
```



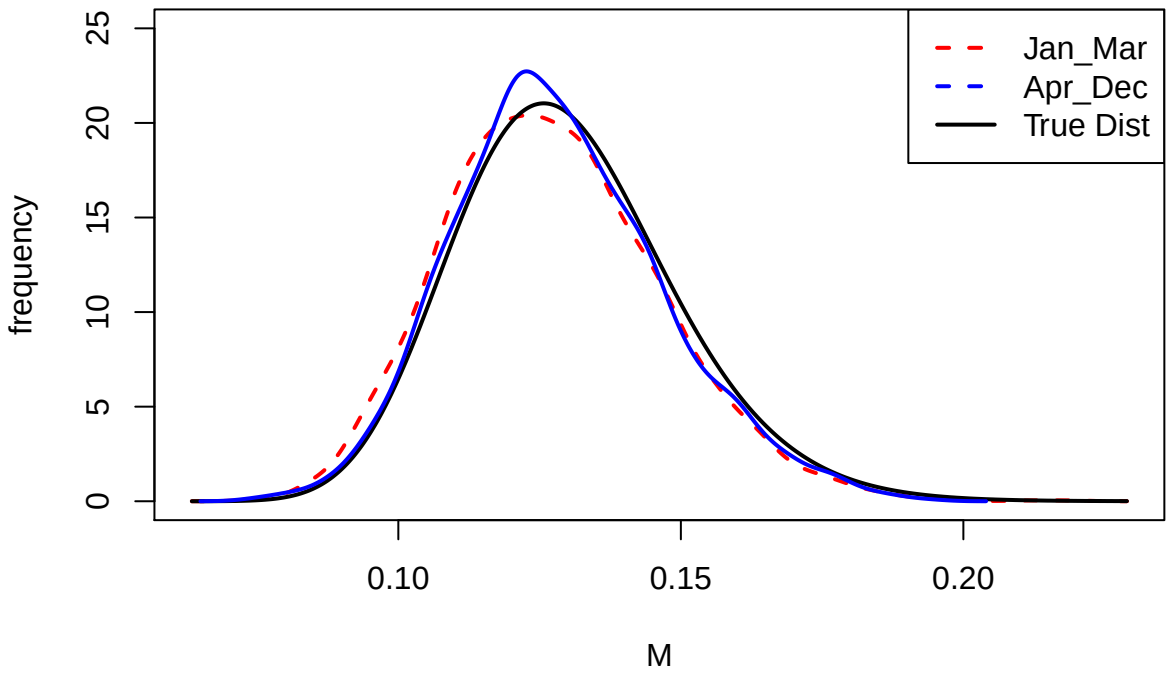
Look natural morality values assigned to each agent, which is generated from a distribution in this case lognormal with expectation 0.13, and cv 0.15 (see line 102 in `Population.ibm`). When each agent is created it is assigned a random M value that is used in the natural mortality process. This is the same as how L_∞ and k are assigned in the growth function.

```
# View Agent attributes for each time-step
jan_mar_1990 = ibm_base$Jan_Mar_agents$`1990`$values
jan_mar_2000 = ibm_base$Jan_Mar_agents$`2000`$values
jan_mar_2010 = ibm_base$Jan_Mar_agents$`2010`$values
apr_dec_1990 = ibm_base$Apr_Dec_agents$`1990`$values
apr_dec_2000 = ibm_base$Apr_Dec_agents$`2000`$values
apr_dec_2010 = ibm_base$Apr_Dec_agents$`2010`$values
# Attributes you can look at
head(jan_mar_1990)
```

```
##   row-col agent_ndx age age_index  length length_index      weight  scalar
## 1     2-1     48449  18        18 46.2863           22 0.002002740 369.983
## 2     1-1    382229  20        20 46.9771           22 0.002087350 402.403
## 3     1-1    119088   3         3 16.1953            7 0.000106621 402.403
## 4     3-1     41370  16        16 50.6731           24 0.002579050 307.526
## 5     3-1    107166  12        12 43.2498           20 0.001656990 307.526
## 6     2-1    351973   7         7 39.1792           18 0.001257240 369.983
##   sex mature         M   L_inf        k         a     b tag alive lat long
```

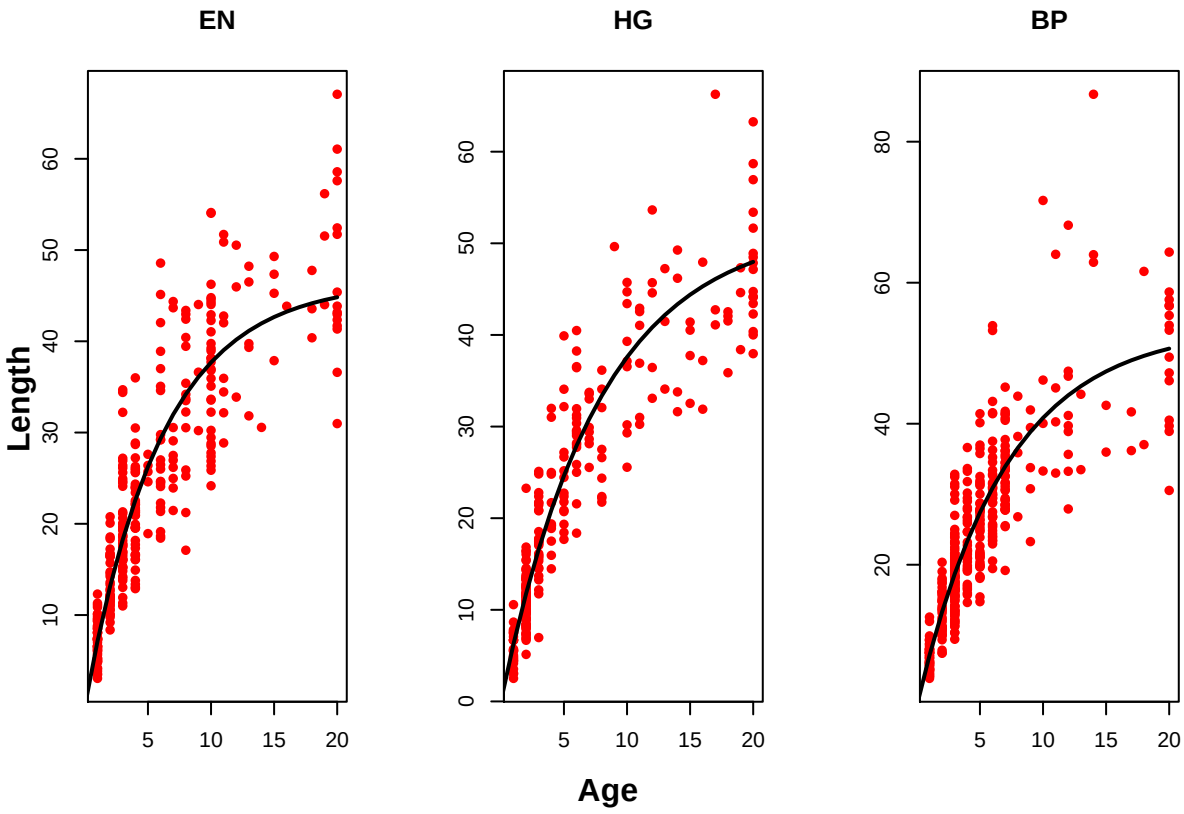
```
## 1   0      1 0.1029470 54.6470 0.104292 4.467e-08 2.793   0     1   0    0
## 2   0      1 0.0982251 47.0008 0.223305 4.467e-08 2.793   0     1   0    0
## 3   0      0 0.1220330 36.9687 0.192070 4.467e-08 2.793   0     1   0    0
## 4   0      1 0.1532360 53.5044 0.183677 4.467e-08 2.793   0     1   0    0
## 5   0      1 0.1275240 55.9563 0.123528 4.467e-08 2.793   0     1   0    0
## 6   0      1 0.1335130 57.2965 0.164458 4.467e-08 2.793   0     1   0    0
##   birth-row-col
## 1           2-1
## 2           1-1
## 3           1-1
## 4           3-1
## 5           3-1
## 6           2-1
```

```r
# check distributions are as we expected
dens_jan = density(c(jan_mar_1990$M, jan_mar_2000$M, jan_mar_2010$M))
dens_dec = density(c(apr_dec_2010$M, apr_dec_2000$M, apr_dec_2010$M))

# close enough
plot(dens_jan$x,dens_jan$y, type = "l", lwd = 2, lty = 2, col = "red",
     xlab = "M", ylab = "frequency", ylim = c(0,25))
log_sigma = sqrt(log(0.15*0.15 + 1.0));
log_mean  = log(0.13) - (log_sigma * log_sigma) / 2.0;
lines(dens_jan$x, dlnorm(dens_jan$x, log_mean, log_sigma), lty = 1,
      lwd = 2, col = "black")
lines(dens_dec$x, dens_dec$y, type = "l", lwd = 2, col = "blue")
legend('topright', legend = c("Jan_Mar", "Apr_Dec","True Dist"),
       lty = c(2,2,1), lwd = 2, col = c("red","blue","black"))
```



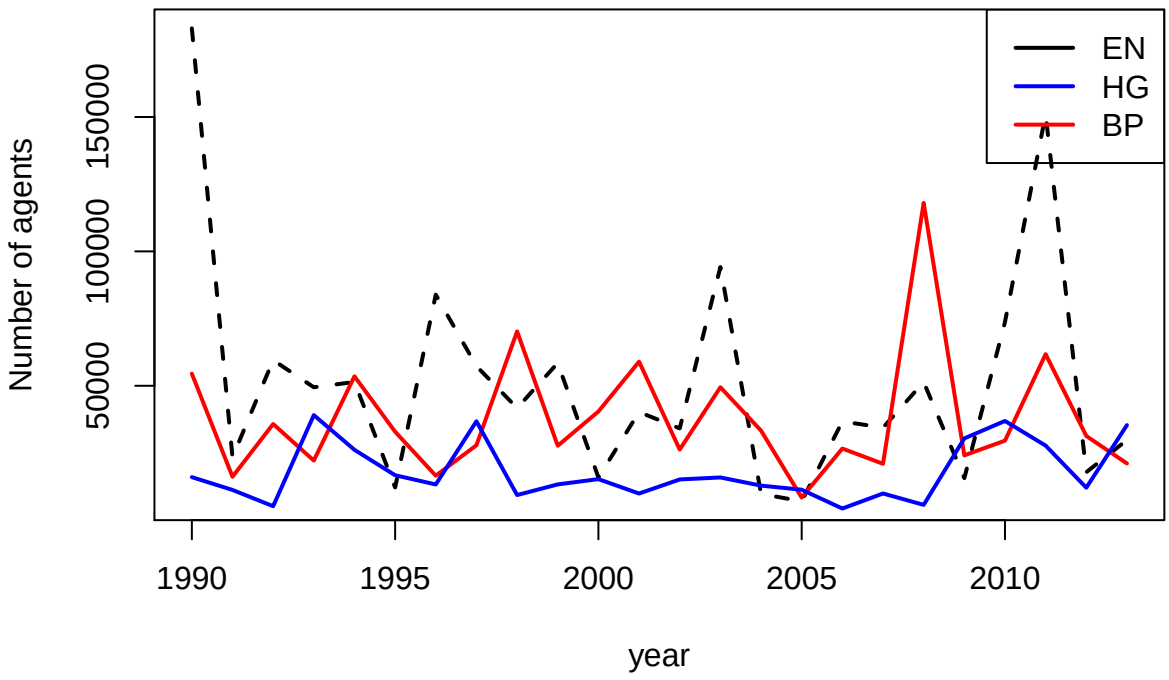
Growth - age length distribution for a random sample of agents.

```
# Plot age-length relationship
EN_growth = jan_mar_2000[jan_mar_2000$`row-col` == "1-1",]
HG_growth = jan_mar_2000[jan_mar_2000$`row-col` == "2-1",]
BP_growth = jan_mar_2000[jan_mar_2000$`row-col` == "3-1",]
par(mfrow = c(1,3))
VB = function (age, K, L_inf, t0) { return(L_inf * (1 - exp(-K * (age - t0))))}
plot(EN_growth$age, EN_growth$length, pch = 16, col = "red", main = "EN",
     xlab = "", ylab = "")
lines(0:20, VB(0:20, 0.166321 ,46.496554 ,0), lwd = 2)
plot(HG_growth$age, HG_growth$length, pch = 16, col = "red", main = "HG",
     xlab = "", ylab = "")
lines(0:20, VB(0:20, 0.1285049 ,51.9492243 ,0), lwd = 2)
plot(BP_growth$age, BP_growth$length, pch = 16, col = "red", main = "BP",
     xlab = "", ylab = "")
lines(0:20, VB(0:20, 0.1424809  ,53.7596026 ,0), lwd = 2)
mtext(adj = 0.5, side = 1, line = -2, outer=T, text = "Age", font = 2)
mtext(adj = 0.5, side = 2, las = 3, line = -2, outer=T, text = "Length", font = 2)
```



Look at recruitmnet over time for the three stocks

```
plot(ibm_base$Rec_EN$year, ibm_base$Rec_EN$recruits, type = "l", lwd = 2, lty = 2,
     xlab = "year", ylab = "Number of agents")
lines(ibm_base$Rec_HG$year, ibm_base$Rec_HG$recruits, lwd = 2, col = "red")
lines(ibm_base$Rec_BP$year, ibm_base$Rec_BP$recruits, lwd = 2, col = "blue")
legend('topright', legend = c("EN", "HG","BP"),
       lty = c(1), lwd = 2, col = c("black","blue","red"))
```



Check that we removed the correct number of catch, some poorly configured models will not let you take all the catch, as there might not be enough vulnerable.

```
# difference between actual removed catches and what we wanted to take out,
t(ibm_base$fishing$actual_catches) - t(ibm_base$fishing$catches)
```

```
##            1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001
## Fishing_LL  2.3  1.1  0.6  0.8    1  1.5    2  0.6  0.4    2  0.5  1.1
##            2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
## Fishing_LL 0.63 0.38 1.11 0.89 1.25 1.54 1.81 0.42 0.37 1.15 3.81 0.82
```

```
# every the subcommand print_extra_info true then you will have recorded every agent that was
# caught during this process.
```

Look at Tag release distirbutions

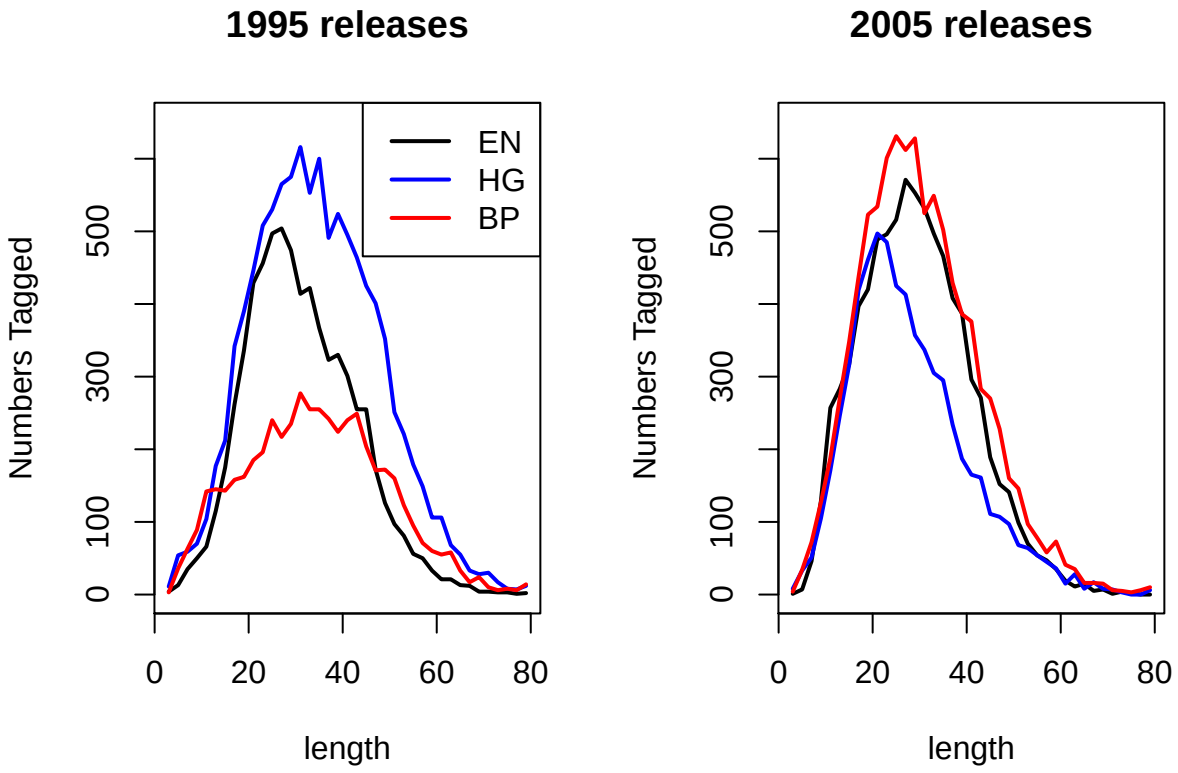
```
# should check that we tagged the correct number of agents.
tag_release_95 = ibm_base$Tagging$`tag_release_by_length-1995`
tag_release_05 = ibm_base$Tagging$`tag_release_by_length-2005`
rowSums(tag_release_95)
```

```
##   1-1   2-1   3-1
##  6782 10234  5043
```

```
rowSums(tag_release_05)
```

```
##  1-1  2-1  3-1
## 8190 6345 9322
```

```r
# look at the length distribution of the tag-release
par(mfrow = c(1,2))
plot(colnames(tag_release_95), tag_release_95[1,], type = "l", lwd = 2,
     xlab = "length", ylab = "Numbers Tagged", main = "1995 releases",
     ylim = c(0,max(tag_release_95,tag_release_05) + 20))
lines(colnames(tag_release_95), tag_release_95[2,], col = "blue", lwd = 2)
lines(colnames(tag_release_95), tag_release_95[3,], col = "red", lwd = 2)
legend('topright', legend = c("EN", "HG","BP"),
       lty = c(1), lwd = 2, col = c("black","blue","red"))

plot(colnames(tag_release_05), tag_release_05[1,], type = "l", lwd = 2,
     xlab = "length", ylab = "Numbers Tagged", main = "2005 releases",
     ylim = c(0,max(tag_release_05,tag_release_05) + 20))
lines(colnames(tag_release_05), tag_release_05[2,], col = "blue", lwd = 2)
lines(colnames(tag_release_05), tag_release_05[3,], col = "red", lwd = 2)
```



Look at movement between areas

```r
## check proportion moving is the same as what we specified in the config files
round(ibm_base$Jump_One$`year-area-1990_1-1`$destination_values
      / ibm_base$Jump_One$`year-area-1990_1-1`$initial_numbers_in_cell,3)
```

```
##       [,1]
## [1,] 0.922
## [2,] 0.075
## [3,] 0.003
```

```
round(ibm_base$Jump_One$`year-area-1990_2-1`$destination_values
      / ibm_base$Jump_One$`year-area-1990_2-1`$initial_numbers_in_cell,3)
```

```
##       [,1]
## [1,] 0.084
## [2,] 0.914
## [3,] 0.002
```

```
round(ibm_base$Jump_One$`year-area-1990_3-1`$destination_values
      / ibm_base$Jump_One$`year-area-1990_3-1`$initial_numbers_in_cell,3)
```

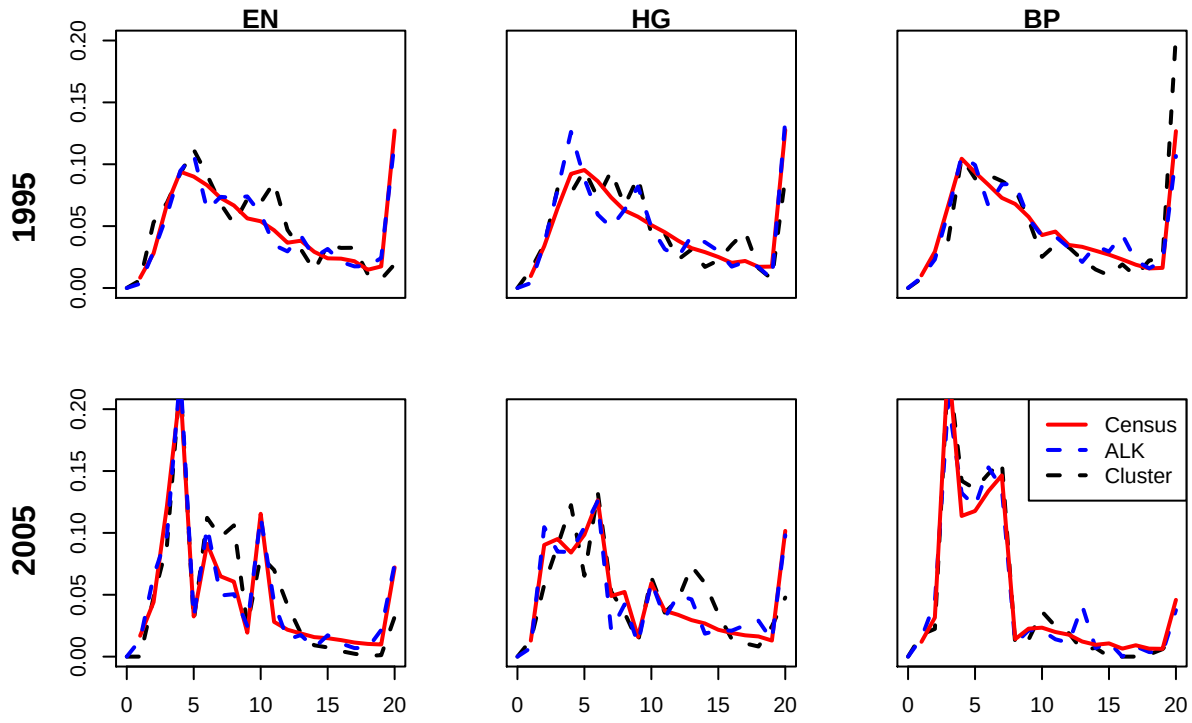
```
##       [,1]
## [1,] 0.054
## [2,] 0.095
## [3,] 0.851
```

Observations

Compare generated observations with the census fishery data to confirm that they generate unbiased expectations i.e. the model is doing what we hope it is doing

```
EN_fish_sample = ibm_base$EN_age_sample$Values
HG_fish_sample = ibm_base$HG_age_sample$Values
BP_fish_sample = ibm_base$BP_age_sample$Values
EN_fish_sample_alk = ibm_base$fishery_age_ALK_EN$Values
HG_fish_sample_alk = ibm_base$fishery_age_ALK_HG$Values
BP_fish_sample_alk = ibm_base$fishery_age_ALK_BP$Values
fish_years = unique(EN_fish_sample$year)
fish_age = unique(EN_fish_sample$age)
# row 1 = age, row 2 = length_midpoint, row 3 = sex
EN_1990 = tabulate(ibm_base$fishing$`census_info-Fishing_LL-1990-1-1`[1,], nbins = 20)
HG_1990 = tabulate(ibm_base$fishing$`census_info-Fishing_LL-1990-2-1`[1,], nbins = 20)
BP_1990 = tabulate(ibm_base$fishing$`census_info-Fishing_LL-1990-3-1`[1,], nbins = 20)
EN_2000 = tabulate(ibm_base$fishing$`census_info-Fishing_LL-2000-1-1`[1,], nbins = 20)
HG_2000 = tabulate(ibm_base$fishing$`census_info-Fishing_LL-2000-2-1`[1,], nbins = 20)
BP_2000 = tabulate(ibm_base$fishing$`census_info-Fishing_LL-2000-3-1`[1,], nbins = 20)

par(mfrow = c(2,3), mar = c(3,3,1,1),oma = c(3,3,2,0))
plot(fish_age, EN_fish_sample[EN_fish_sample$year == 1990,"simulated"], type = "l",
     lwd = 2, xaxt = "n", xlab = "", ylab = "proportions", main = "EN", ylim = c(0,0.2), lty = 2)
lines(1:20, EN_1990 / sum(EN_1990), lwd = 2, col = "red", lty = 1)
lines(fish_age, EN_fish_sample_alk[EN_fish_sample_alk$year == 1990,"simulated"], lwd = 2, col = "blue",

plot(fish_age, HG_fish_sample[HG_fish_sample$year == 1990,"simulated"], type = "l",
     lwd = 2, xaxt = "n", yaxt = "n", xlab = "", ylab = "", ylim = c(0,0.2), main = "HG", lty = 2)
lines(1:20, HG_1990 / sum(HG_1990), lwd = 2, col = "red", lty = 1)
lines(fish_age, HG_fish_sample_alk[HG_fish_sample_alk$year == 1990,"simulated"], lwd = 2, col = "blue",

plot(fish_age, BP_fish_sample[BP_fish_sample$year == 1990,"simulated"], type = "l",
     lwd = 2, xaxt = "n", yaxt = "n", xlab = "", ylab = "", ylim = c(0,0.2), main = "BP", lty = 2)
lines(1:20, BP_1990 / sum(BP_1990), lwd = 2, col = "red", lty = 1)
lines(fish_age, BP_fish_sample_alk[BP_fish_sample_alk$year == 1990,"simulated"], lwd = 2, col = "blue",

plot(fish_age, EN_fish_sample[EN_fish_sample$year == 2000,"simulated"], type = "l",
```

```
     lwd = 2, xlab = "", ylab = "proportions", ylim = c(0,0.2), lty = 2)
lines(1:20, EN_2000 / sum(EN_2000), lwd = 2, col = "red", lty = 1)
lines(fish_age, EN_fish_sample_alk[EN_fish_sample_alk$year == 2000,"simulated"], lwd = 2, col = "blue",

plot(fish_age, HG_fish_sample[HG_fish_sample$year == 2000,"simulated"], type = "l",
     lwd = 2, xlab = "", yaxt = "n", ylab = "", ylim = c(0,0.2), lty = 2)
lines(1:20, HG_2000 / sum(HG_2000), lwd = 2, col = "red", lty = 1)
lines(fish_age, HG_fish_sample_alk[HG_fish_sample_alk$year == 2000,"simulated"], lwd = 2, col = "blue",

plot(fish_age, BP_fish_sample[BP_fish_sample$year == 2000,"simulated"], type = "l",
     lwd = 2, xlab = "", yaxt = "n", ylab = "", ylim = c(0,0.2), lty = 2)
lines(fish_age, BP_fish_sample_alk[BP_fish_sample_alk$year == 2000,"simulated"], lwd = 2, col = "blue",
lines(1:20, BP_2000 / sum(BP_2000), lwd = 2, col = "red", lty = 1)

mtext(side = 2, las = 3, adj = 0.75, outer = T, line = 0, text = "1995", font = 2)
mtext(side = 2, las = 3, adj = 0.25, outer = T, line = 0, text = "2005", font = 2)
legend('topright', legend = c("Census","ALK","Cluster"), col = c("red","blue","black"), lty = c(1,2,2),
```



The cluster method doesn't seem to be describing the plus group well, best to look at many simualtions and take an average. This can be done using the `ibm -s` command in the ABM. See the file `simulated_obs.log` to see the command call.

```
sim_ibm = extract.run(file = "../base_ibm/simulated_obs.log")
```

```
## loading a run from -i or -s format
```

```
sim_ibm$warnings_encounted$`1`$warnings_found
```

```
## [1] 7206
```

```
sim_ibm$model_run_time
```

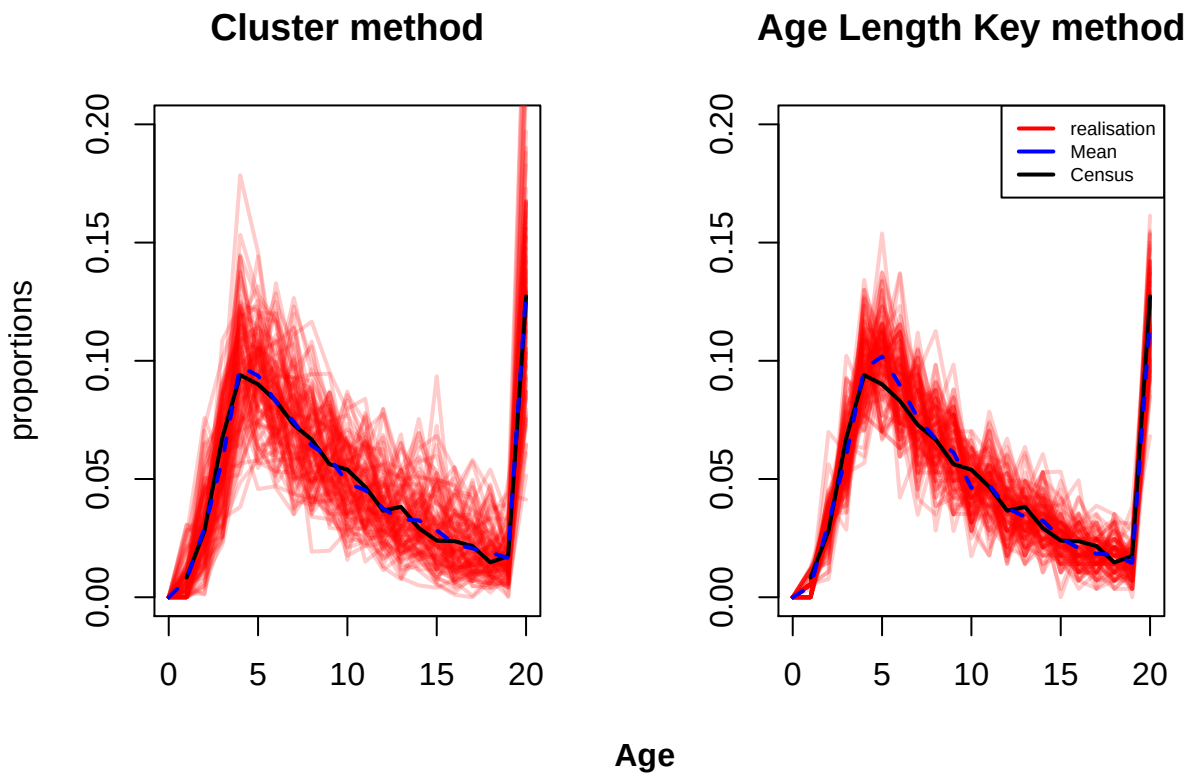
```
## [1] "1.167 minutes"
```

```
#plot.derived_quantities(sim_ibm, report_label = "derived_quants")
# read in data
sim_obs_location = "../base_ibm/sim_obs"
sim_file_names = unique(sapply(strsplit(list.files(sim_obs_location), split = "\\."), "[", 1))
sim_file_names
```

```
##  [1] "BP_age_sample"         "EN_age_sample"
##  [3] "fishery_age_ALK_BP"    "fishery_age_ALK_EN"
##  [5] "fishery_age_ALK_HG"    "HG_age_sample"
##  [7] "tag_recapture_1995_BP" "tag_recapture_1995_EN"
##  [9] "tag_recapture_1995_HG" "tag_recapture_2005_BP"
## [11] "tag_recapture_2005_EN" "tag_recapture_2005_HG"
```

```
extensions = unique(sapply(strsplit(list.files(sim_obs_location), split = "\\."), "[", 2))
EN_1990_age = EN_2000_age = matrix(0, nrow = length(extensions), ncol = 21)
EN_1990_age_alk = EN_2000_age_alk = matrix(0, nrow = length(extensions), ncol = 21)
HG_1990_age = HG_2000_age = matrix(0, nrow = length(extensions), ncol = 21)
HG_1990_age_alk = HG_2000_age_alk = matrix(0, nrow = length(extensions), ncol = 21)
BP_1990_age = BP_2000_age =  matrix(0, nrow = length(extensions), ncol = 21)
BP_1990_age_alk = BP_2000_age_alk = matrix(0, nrow = length(extensions), ncol = 21)
```

```
  par(mfrow = c(1,2))
  plot(1:20, EN_1990 / sum(EN_1990), type = "l", lwd = 2, xlab = "", ylab = "proportions",
       main = "Cluster method",xlim = c(0,20), ylim = c(0,0.2), lty = 2)
  for(i in 1:nrow(EN_1990_age)) {
      lines(fish_age, EN_1990_age[i,], lwd = 2, col = adjustcolor(col = "red", alpha.f = 0.2),
            lty = 1)
  }
  lines(1:20, EN_1990 / sum(EN_1990), lwd = 2, col = "black", lty = 1)
  lines(fish_age, apply(EN_1990_age, MARGIN = 2,FUN = function(x){mean(as.numeric(x))}),
        lwd = 2, col = "blue", lty = 2)

  plot(1:20, EN_1990 / sum(EN_1990), type = "l", lwd = 2, xlab = "", ylab = "",
       main = "Age Length Key method",xlim = c(0,20), ylim = c(0,0.2), lty = 2)
  for(i in 1:nrow(EN_1990_age_alk)) {
      lines(fish_age, EN_1990_age_alk[i,], lwd = 2, col = adjustcolor(col = "red", alpha.f = 0.2),
            lty = 1)
  }
  lines(1:20, EN_1990 / sum(EN_1990), lwd = 2, col = "black", lty = 1)
  lines(fish_age, apply(EN_1990_age_alk, MARGIN = 2,FUN = function(x){mean(as.numeric(x))}), lwd = 2,
        col = "blue", lty = 2)
  mtext(side = 1, adj = 0.52, outer = T, line = -2, text = "Age", font = 2)
  legend('topright', legend = c("realisation","Mean","Census"), col = c("red","blue","black"),
         lty = c(1), lwd = 2, cex = 0.6)
```



Looks like a bias occurs using the Age length Key method, this will need to be investigated.