Summarise IBM

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.

Read in Data

```
library(ibm)
library(ggplot2)
library(reshape2)
## read in base model output
ibm_base = extract.run("../base_ibm/run.log")
names(ibm_base) # all the reports.
```

```
##  [1] "init_1"                "init_2"
##  [3] "age_freq1"             "age_freq"
##  [5] "mature_report"         "fishing"
##  [7] "Jump_One"              "Movement_home"
##  [9] "Rec_EN"                "Rec_HG"
## [11] "Tagging"               "Rec_BP"
## [13] "derived_quants"        "model_attributes"
## [15] "fisher_age_freq"       "CPUE"
## [17] "HG_age_sample"         "EN_age_sample"
## [19] "BP_age_sample"         "tag_recapture_1995_EN"
## [21] "tag_recapture_1995_HG" "tag_recapture_1995_BP"
## [23] "tag_recapture_2005_EN" "tag_recapture_2005_HG"
## [25] "tag_recapture_2005_BP" "warnings_encounted"
## [27] "model_run_time"
```

Building up an ABM

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.

```
## 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")
# look at time difference
ibm_20$model_run_time
```

```
## [1] "12 seconds"
```

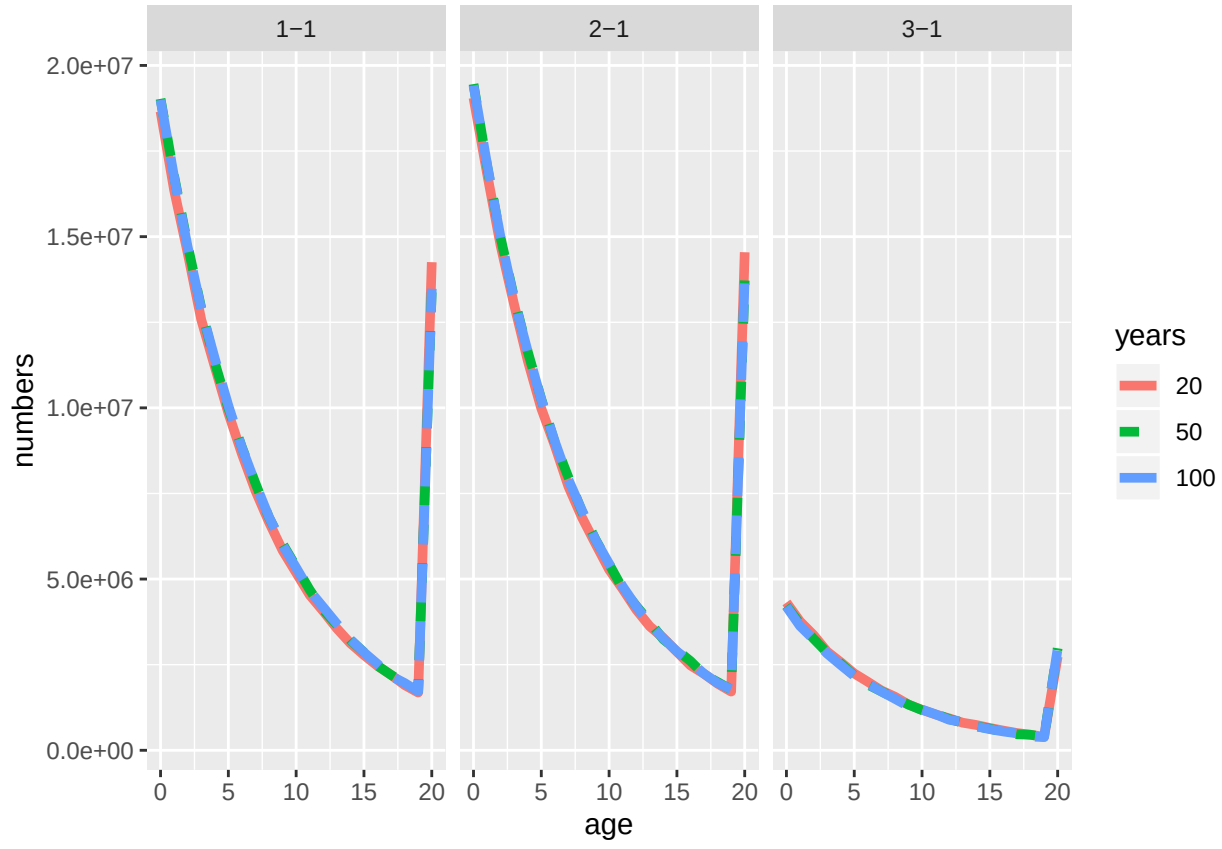
```
ibm_50$model_run_time
```

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

```
ibm_100$model_run_time
```

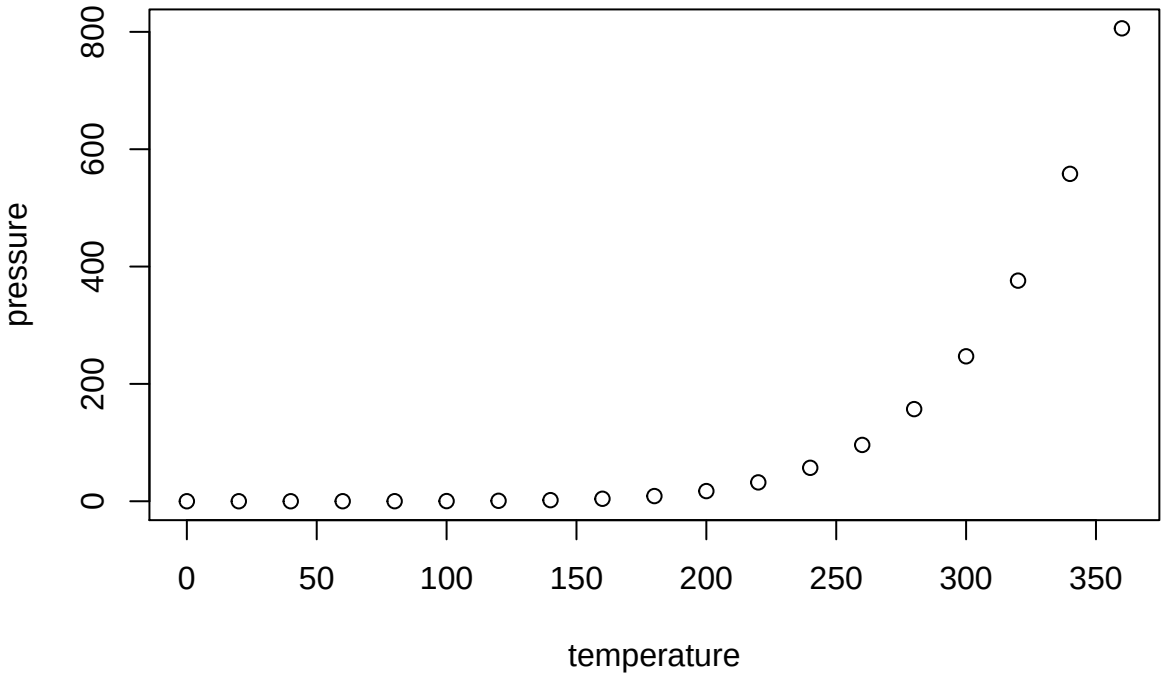
```
## [1] "49 seconds"
```

```
# 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_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 = 1.7) +
  facet_wrap( ~ area, nrow = 1)
```




```
## not much difference will settle with using 40 years.
```

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.