CINAR MSE Workshop: Age Structured Model

Gavin Fay

January 2022

```
source("age-functions.R")
source("performance-metrics.R")
```

```
my_future_options <- future_options()
```

```
## Warning: `future_options()` was deprecated in furrr 0.2.0.
## Please use `furrr_options()` instead.
```

```
my_future_options$globals <- ls()
my_future_options$packages <- c("tidyverse")
future::plan(future::multisession)
```

Age-structured Fisheries Model

$$N_{1,t+1} = R_{t+1}$$

$$N_{a+1,t+1} = N_{a,t}e^{-(F_{a,t}+M)}$$

$$N_{A,t+1} = N_{A-1,t}e^{-(F_{A-1,t}+M)} + N_{A,t}e^{-(F_{A,t}+M)}$$

Matrix form Age-structured Fisheries Model

$$\mathbf{N}_{t+1} = \begin{bmatrix} R_{t+1} \\ 0 \\ 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & 0 \\ e^{-(F_{0,t}+M)} & 0 & 0 & \cdots & 0 & 0 \\ 0 & e^{-(F_{1,t}+M)} & 0 & \cdots & 0 & 0 \\ 0 & 0 & e^{-(F_{2,t}+M)} & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & e^{-(F_{A-1,t}+M)} & e^{-(F_{A,t}+M)} \end{bmatrix} \begin{bmatrix} N_{0,t} \\ N_{1,t} \\ N_{2,t} \\ N_{3,t} \\ \cdots \\ N_{A,t} \end{bmatrix}$$

```
# # ## read in biol data & create input object
#path <- "materials/exercises/day-02/"
input <- list()
input$nages <- scan("floundah_biology.txt",n=1,skip=9)
nages <- input$nages
input$maturity <- scan("floundah_biology.txt",n=nages,skip=11)
input$selex <- scan("floundah_biology.txt",n=nages,skip=13)
input$weight <- scan("floundah_biology.txt",n=nages,skip=15)
#
input$M <- 0.4
input$Fmult <- 1 #proportion of FSPRtarg during historical period
input$SPRtarg <- 0.4
input$Rbar <- 1000
input$sigmaR <- 0.6
input$cpueCV <- 0.3
#xx <- evaluate(input, seed=24601)
```

```
om_settings <- tibble(om=1,
                      input=list(input)) %>%
  I()
om_settings$input
```

```
## [[1]]
## [[1]]$nages
## [1] 10
##
## [[1]]$maturity
##  [1] 0.1 0.5 0.9 1.0 1.0 1.0 1.0 1.0 1.0 1.0
##
## [[1]]$selex
##  [1] 0.10 0.75 0.90 1.00 1.00 1.00 1.00 1.00 1.00
## [10] 1.00
##
## [[1]]$weight
##  [1] 0.040 0.160 0.331 0.511 0.675 0.812 0.922 1.006
##  [9] 1.070 1.117
##
## [[1]]$M
## [1] 0.4
##
## [[1]]$Fmult
## [1] 1
##
## [[1]]$SPRtarg
## [1] 0.4
##
## [[1]]$Rbar
## [1] 1000
##
## [[1]]$sigmaR
## [1] 0.6
##
## [[1]]$cpueCV
## [1] 0.3
```

```
om_settings
```

```
## # A tibble: 1 x 2
##      om input
##   <dbl> <list>
## 1     1 <named list [10]>
```

set up the random number seeds for each iteration

```
nsim <- 1000
set.seed(24601)
seeds <- ceiling(99999*runif(nsim))

sim_seeds <- tibble(isim = 1:nsim,
                    seed = seeds)
```

set up the table for running the mse over all the operating models (just 1 in this example)

```
mse_sim_setup <- om_settings %>%
  mutate(sim_seeds= list(sim_seeds)) %>%
  unnest(cols = sim_seeds) %>%
  rowid_to_column() %>%
  select(rowid,everything()) %>%
  ungroup() %>%
  arrange(isim) %>%   # organizes so first realization of each is done first
  I()
mse_sim_setup
```

```
## # A tibble: 1,000 x 5
##    rowid    om input              isim  seed
##    <int> <dbl> <list>            <int> <dbl>
##  1     1     1 <named list [10]>     1 39892
##  2     2     1 <named list [10]>     2  8881
##  3     3     1 <named list [10]>     3 72021
##  4     4     1 <named list [10]>     4 98908
##  5     5     1 <named list [10]>     5 68208
##  6     6     1 <named list [10]>     6 30042
##  7     7     1 <named list [10]>     7  3019
##  8     8     1 <named list [10]>     8 18135
##  9     9     1 <named list [10]>     9 39192
## 10    10     1 <named list [10]>    10 66541
## # ... with 990 more rows
```

run the mse for all the iterations

```
### run the MSE over each row of the mse_sims todo
#add a safe mode (returns error safely rather than crashing)
safe_evaluate <- purrr::safely(evaluate, otherwise = NA_real_)
#do the MSE for all simulations and scenarios
#profvis::profvis(
  #system.time(
mse_output <- mse_sim_setup %>%
#   mutate(mse = furrr::future_pmap(list(seed = seed, input = input),
#                            safe_evaluate, .options = my_future_options)) %>%
# this is the regular purrr code for iterating over the simulations
mutate(mse = purrr::pmap(list(seed = seed, input = input), evaluate)) %>%
  rowwise() %>%
  mutate(om_SSB = list(mse$om_SSB)) %>%
  mutate(om_F = list(mse$om_F)) %>%
  mutate(tac = list(mse$tac)) %>%
  mutate(ftarg = mse$ftarg) %>%
  I()
mse_output
```

```
## # A tibble: 1,000 x 10
## # Rowwise:
##    rowid    om input   isim  seed mse   om_SSB om_F
##    <int> <dbl> <list> <int> <dbl> <lis> <list> <lis>
##  1     1     1 <name~     1 39892 <nam~ <dbl ~ <dbl~
##  2     2     1 <name~     2  8881 <nam~ <dbl ~ <dbl~
##  3     3     1 <name~     3 72021 <nam~ <dbl ~ <dbl~
##  4     4     1 <name~     4 98908 <nam~ <dbl ~ <dbl~
##  5     5     1 <name~     5 68208 <nam~ <dbl ~ <dbl~
```

```
##  6     6     1 <name~     6 30042 <nam~ <dbl ~ <dbl~
##  7     7     1 <name~     7  3019 <nam~ <dbl ~ <dbl~
##  8     8     1 <name~     8 18135 <nam~ <dbl ~ <dbl~
##  9     9     1 <name~     9 39192 <nam~ <dbl ~ <dbl~
## 10    10     1 <name~    10 66541 <nam~ <dbl ~ <dbl~
## # ... with 990 more rows, and 2 more variables:
## #   tac <list>, ftarg <dbl>
```

```
mse_results <- mse_output %>%
  rowwise() %>%
  mutate(bmsy = input$Rbar*SBPR(ftarg,input$M,input$weight,input$selex,input$maturity)) %>%
  mutate(msy = input$Rbar*YPR(ftarg,input$M,input$weight,input$selex)) %>%
  ungroup() %>%
  mutate(ssb_metrics = pmap(list(om_SSB, bmsy), get_ssb_metrics, nprojyrs = 50),
         catch_metrics = pmap(list(tac, msy), get_catch_metrics, nprojyrs = 50),
         f_metrics = pmap(list(om_F, ftarg), get_F_metrics, nprojyrs = 50)) %>%
  select(rowid, om, isim, ssb_metrics, catch_metrics, f_metrics) %>%
  I()
mse_results
```

```
## # A tibble: 1,000 x 6
##    rowid    om  isim ssb_metrics     catch_metrics
##    <int> <dbl> <int> <list>          <list>
##  1     1     1     1 <named list [8~ <named list [1~
##  2     2     1     2 <named list [8~ <named list [1~
##  3     3     1     3 <named list [8~ <named list [1~
##  4     4     1     4 <named list [8~ <named list [1~
##  5     5     1     5 <named list [8~ <named list [1~
##  6     6     1     6 <named list [8~ <named list [1~
##  7     7     1     7 <named list [8~ <named list [1~
##  8     8     1     8 <named list [8~ <named list [1~
##  9     9     1     9 <named list [8~ <named list [1~
## 10    10     1    10 <named list [8~ <named list [1~
## # ... with 990 more rows, and 1 more variable:
## #   f_metrics <list>
```

`mse_results` is a tibble containing the results, and a list of SSB, catch, and F performance metrics

pull out the metrics

```
ssb_results <- mse_results %>%
  select(rowid, om, isim, ssb_metrics) %>%
  mutate(ssb_metrics = map(ssb_metrics, enframe)) %>%
  unnest(cols = c(ssb_metrics)) %>%
  mutate(value = map_dbl(value, I)) %>%
  rename(metric = name) %>%
  I()
ssb_results
```

```
## # A tibble: 8,000 x 5
##    rowid    om  isim metric            value
##    <int> <dbl> <int> <chr>             <dbl>
##  1     1     1     1 s_n_less_01_bmsy  0
##  2     1     1     1 s_n_less_05_bmsy  0
##  3     1     1     1 s_n_ge_bmsy       2
##  4     1     1     1 l_n_less_01_bmsy  0
```

```
##  5     1     1     1 l_n_less_05_bmsy  0
##  6     1     1     1 l_n_ge_bmsy      17
##  7     1     1     1 s_avg_ssb_ssbmsy  0.884
##  8     1     1     1 l_avg_ssb_ssbmsy  1.09
##  9     2     1     2 s_n_less_01_bmsy  0
## 10     2     1     2 s_n_less_05_bmsy  0
## # ... with 7,990 more rows
```

```
unique(ssb_results$metric)
```

```
## [1] "s_n_less_01_bmsy" "s_n_less_05_bmsy"
## [3] "s_n_ge_bmsy"      "l_n_less_01_bmsy"
## [5] "l_n_less_05_bmsy" "l_n_ge_bmsy"
## [7] "s_avg_ssb_ssbmsy" "l_avg_ssb_ssbmsy"
```

```
###pull out the f metrics
f_results <- mse_results %>%
  select(rowid, om, isim, f_metrics) %>%
  mutate(f_metrics = map(f_metrics, enframe)) %>%
  unnest(cols = c(f_metrics)) %>%
  mutate(value = map_dbl(value, I)) %>%
  rename(metric = name) %>%
  I()
f_results
```

```
## # A tibble: 6,000 x 5
##    rowid    om  isim metric         value
##    <int> <dbl> <int> <chr>          <dbl>
##  1     1     1     1 s_n_gr_fmsy    0
##  2     1     1     1 s_n_less_fmsy  6
##  3     1     1     1 l_n_gr_fmsy    0
##  4     1     1     1 l_n_less_fmsy 30
##  5     1     1     1 s_avg_f_fmsy   0.363
##  6     1     1     1 l_avg_f_fmsy   0.374
##  7     2     1     2 s_n_gr_fmsy    0
##  8     2     1     2 s_n_less_fmsy  6
##  9     2     1     2 l_n_gr_fmsy    0
## 10     2     1     2 l_n_less_fmsy 30
## # ... with 5,990 more rows
```

```
unique(f_results$metric)
```

```
## [1] "s_n_gr_fmsy"   "s_n_less_fmsy" "l_n_gr_fmsy"
## [4] "l_n_less_fmsy" "s_avg_f_fmsy"  "l_avg_f_fmsy"
```

```
###pull out the catch metrics
catch_results <- mse_results %>%
  select(rowid, om, isim, catch_metrics) %>%
  mutate(catch_metrics = map(catch_metrics, enframe)) %>%
  unnest(cols = c(catch_metrics)) %>%
  mutate(value = map_dbl(value, I)) %>%
  rename(metric = name) %>%
  I()
catch_results
```

```
## # A tibble: 10,000 x 5
##    rowid    om  isim metric               value
```

```
##    <int> <dbl> <int> <chr>                <dbl>
##  1     1     1     1 s_avg_catch         39.6
##  2     1     1     1 l_avg_catch         78.2
##  3     1     1     1 s_avg_catch_msy      0.418
##  4     1     1     1 l_avg_catch_msy      0.826
##  5     1     1     1 s_sd_catch          20.0
##  6     1     1     1 l_sd_catch          30.5
##  7     1     1     1 l_iav_catch         NA
##  8     1     1     1 s_iav_catch          0.461
##  9     1     1     1 a_iav_catch         NA
## 10     1     1     1 l_prop_g_msy_2_of_3 NA
## # ... with 9,990 more rows
```

```
full_metrics <- bind_rows(ssb_results, catch_results, f_results)
```

summarize across simulations by OM scenario 25%, 50%, 75% quantiles

```
quibble <- function(x, q = c(0.25, 0.5, 0.75)) {
  tibble(x = quantile(x, q, na.rm = TRUE), q = q)
}

summary <- full_metrics %>%
  group_by(metric, om) %>%
  summarise(y = list(quibble(value, c(0.25, 0.5, 0.75)))) %>%
  tidyr::unnest(y) %>%
  I()
```

```
## `summarise()` has grouped output by 'metric'. You can override using the `.groups` argument.
```

```
summary
```

```
## # A tibble: 72 x 4
## # Groups:   metric [24]
##    metric             om      x     q
##    <chr>           <dbl>  <dbl> <dbl>
##  1 a_iav_catch         1 NA      0.25
##  2 a_iav_catch         1 NA      0.5
##  3 a_iav_catch         1 NA      0.75
##  4 l_avg_catch         1 68.9    0.25
##  5 l_avg_catch         1 74.6    0.5
##  6 l_avg_catch         1 81.1    0.75
##  7 l_avg_catch_msy     1  0.728  0.25
##  8 l_avg_catch_msy     1  0.787  0.5
##  9 l_avg_catch_msy     1  0.856  0.75
## 10 l_avg_f_fmsy        1  0.375  0.25
## # ... with 62 more rows
```