Misspecification Demonstration

Chris Legault

2022-07-01

Overview

This demonstration explores a single realization of catch misspecification with different levels of state-space models applied to the misspecified data. The goal is to demonstrate the diagnostics that can detect the misspecification and/or account for the problem. For comparison, a correctly specified model is also shown.

Building the data sets

The process starts with creating an intial data set using the make_basic_info function and assigning values for selectivity, natural mortality, and recruitment. For this example, there is a single fleet and two indices all with logisitic selectivity, natural mortality is constant at 0.2 for all years and ages, there is no stock-recruitment relationship (recruitment varies as an identical and independent deviation about the mean). The prepare_wham_input function creates the correctly specified input formulation.

```
# only doing one simulation here, but keep structure for multiple sims for use later
nsim <- 1

# create input
groundfish_info <- make_basic_info()

gf_selectivity = list(
  model = c(rep("logistic", groundfish_info$n_fleets),rep("logistic", groundfish_info$n_indices)),
  initial_pars = rep(list(c(5,1)), groundfish_info$n_fleets + groundfish_info$n_indices)) #fleet, index

gf_M = list(initial_means = rep(0.2, length(groundfish_info$ages)))

gf_NAA_re = list(
  N1_pars = exp(10)*exp(-(0:(length(groundfish_info$ages)-1))*gf_M$initial_means[1]),
  sigma = "rec", #random about mean
  cor="iid", #random effects are independent
  use_steepness = 0,
  recruit_model = 2, #random effects with a constant mean
  recruit_pars = exp(10)
)

input <- prepare_wham_input(basic_info = groundfish_info,
                            selectivity = gf_selectivity,
                            NAA_re = gf_NAA_re,
                            M= gf_M)
```

```
## [1] "number of selblocks, 3, is being determined by input$data$selblock_pointer_fleets and input$data
```

The fit_wham function is run without fitting the model to create the operating model.

```
# run starter input
om <- fit_wham(input, do.fit = FALSE, MakeADFun.silent = TRUE)
```

Four estimation models are created using different levels of "state-spaceness" for the population numbers at age and recruitment formulations. The first estimation model exactly matches the operating model, meaning recruitment is estimated as a random effect about a mean value as independent and identical deviates. Ages 2 and older follow the population equation just as a statistical catch-at-age model would. The second estimation model also only has random effects for recruitment, but in this case the deviations are estimated as an autoregressive process over time. The third estimation model allows for random effects in both recruitment and for all other ages. This is the first full state-space model in the series. The random effects are all treated as independent and identical deviates with two separate sigmas estimated, one for recruitment and the other for all other ages. The fourth estimation model is also a full state-space model but the random effects are modeled as a two dimensional autoregressive process over both ages and years, with recruitment being treated separately from all other ages.

```
# set up estimation models
em_input1 <- input # self-test

gf_NAA_re2 <- gf_NAA_re
gf_NAA_re2$cor <- "ar1_y"
em_input2 <- prepare_wham_input(basic_info = groundfish_info, selectivity = gf_selectivity, NAA_re = gf_
```

```
## [1] "number of selblocks, 3, is being determined by input$data$selblock_pointer_fleets and input$data
```

```
gf_NAA_re3 <- gf_NAA_re
gf_NAA_re3$sigma <- "rec+1"
em_input3 <- prepare_wham_input(basic_info = groundfish_info, selectivity = gf_selectivity, NAA_re = gf_
```

```
## [1] "number of selblocks, 3, is being determined by input$data$selblock_pointer_fleets and input$data
```

```
gf_NAA_re4 <- gf_NAA_re
gf_NAA_re4$cor <- "2dar1"
gf_NAA_re4$sigma <- "rec+1"
em_input4 <- prepare_wham_input(basic_info = groundfish_info, selectivity = gf_selectivity, NAA_re = gf_
```

```
## [1] "number of selblocks, 3, is being determined by input$data$selblock_pointer_fleets and input$data
```

Simulated data are created under two conditions: correctly an incorrectly reported catch in the recent years of the assessment. The incorrectly reported catch uses the bias_data function. In this example, catch is under-reported by half (the catch provided to the wham model is only half of the actual catch) for the econd half of the assessment time period. The four estimation models are then set up to be applied to the two sets of data, resulting in a total of 8 models. The first four models use the correct catch data while the last four models use the under-reported catch data.

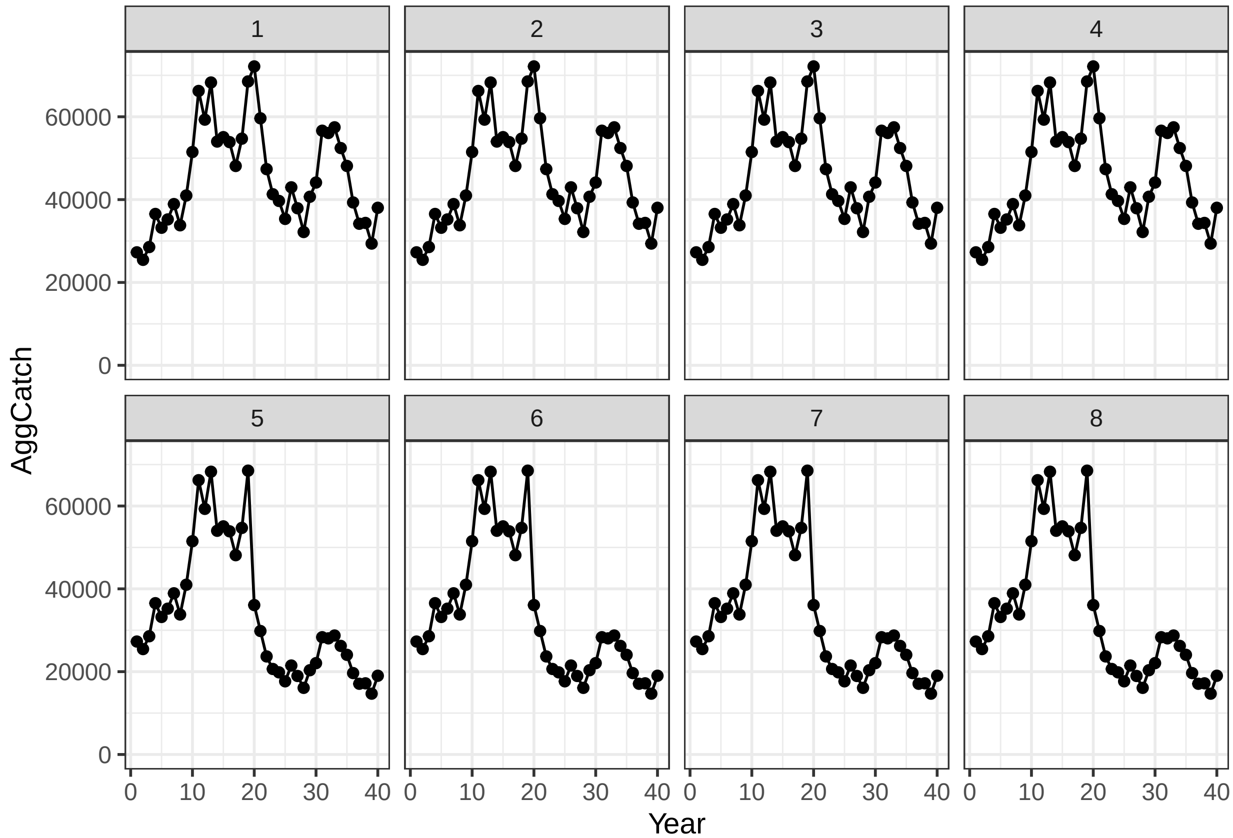
```
#simulate data from operating model
#all RE and data are simulated
sim_input <- list()

# sim_input[[1]] has no data modification and em1
```

```r
set.seed(14159265) #use same seed for all operating models
sim_input[[1]] = lapply(1:nsim, function(x) {
  input_i = em_input1
  sim = om$simulate(complete=TRUE)
  input_i$data = sim
  return(input_i)
})

# sim_input[[2]] uses the correct data and em2
sim_input[[2]] = lapply(1:nsim, function(x) {
  input_i = em_input2
  obs_names = c("agg_indices","agg_catch","catch_paa","index_paa", "Ecov_obs", "obsvec")
  input_i$data[obs_names] = sim_input[[1]][[x]]$data[obs_names]
  return(input_i)
})

# sim_input[[3]] uses the correct data and em3
sim_input[[3]] = lapply(1:nsim, function(x) {
  input_i = em_input3
  obs_names = c("agg_indices","agg_catch","catch_paa","index_paa", "Ecov_obs", "obsvec")
  input_i$data[obs_names] = sim_input[[1]][[x]]$data[obs_names]
  return(input_i)
})

# sim_input[[4]] uses the correct data and em4
sim_input[[4]] = lapply(1:nsim, function(x) {
  input_i = em_input4
  obs_names = c("agg_indices","agg_catch","catch_paa","index_paa", "Ecov_obs", "obsvec")
  input_i$data[obs_names] = sim_input[[1]][[x]]$data[obs_names]
  return(input_i)
})

# now create the under-reported catch data
# sim_input[[5]] has under-reported catch and em1
set.seed(14159265) #use same seed for all operating models
agg_catch_multiplier <- create_agg_catch_multiplier(input, multiplier=0.50)
sim_input[[5]] = lapply(1:nsim, function(x) {
  input_i = em_input1
  sim = om$simulate(complete=TRUE)
  sim <- bias_data(sim, multiply_agg_catch_flag=TRUE,
                   agg_catch_multiplier=agg_catch_multiplier)
  input_i$data = sim
  return(input_i)
})

# sim_input[[6]] has under-reported catch and em2
sim_input[[6]] = lapply(1:nsim, function(x) {
  input_i = em_input2
  obs_names = c("agg_indices","agg_catch","catch_paa","index_paa", "Ecov_obs", "obsvec")
  input_i$data[obs_names] = sim_input[[5]][[x]]$data[obs_names]
  return(input_i)
})
```

```
# sim_input[[7]] has under-reported catch and em3
sim_input[[7]] = lapply(1:nsim, function(x) {
  input_i = em_input3
  obs_names = c("agg_indices","agg_catch","catch_paa","index_paa", "Ecov_obs", "obsvec")
  input_i$data[obs_names] = sim_input[[5]][[x]]$data[obs_names]
  return(input_i)
})

# sim_input[[8]] has under-reported catch and em4
sim_input[[8]] = lapply(1:nsim, function(x) {
  input_i = em_input4
  obs_names = c("agg_indices","agg_catch","catch_paa","index_paa", "Ecov_obs", "obsvec")
  input_i$data[obs_names] = sim_input[[5]][[x]]$data[obs_names]
  return(input_i)
})
```

Check to see that the eight data sets have the correct aggregate catch. Models 1-4 have the true catch, while models 5-8 have the under-reported catch in recent years.

```
aggc <- tibble(Model = character(), Year = integer(), AggCatch = double())
for (m in 1:8){
  aggcdat <- lapply(1:nsim, function(x){
    out <- sim_input[[m]][[x]]$data$agg_catch
  return(out)
  })
  thisaggc <- tibble(Model = m, Year = 1:sim_input[[m]][[1]]$data$n_years_model, AggCatch = unlist(aggc
  aggc <- rbind(aggc, thisaggc)
}

p1 <- ggplot(aggc, aes(x=Year, y=AggCatch)) +
  geom_point() +
  geom_line() +
  facet_wrap(~Model, ncol = 4) +
  expand_limits(y = 0) +
  theme_bw()

print(p1)
```



Now run the eight models. This may take a few minutes. The Rmarkdown compilation checks to see if the file misspec_demo_fits.RDS is present in the misspecification_study directory. If it is, it simply reads that file. If it is not, then the model is run. So if you change any of the settings above, be sure to remove the RDS file before running the Rmarkdown script.

```
if(file.exists(file.path(here(), "misspecification_study", "misspec_demo_fits.RDS"))){
  em_fits <- readRDS(file.path(here(), "misspecification_study", "misspec_demo_fits.RDS"))
} else {
  # run the models
  em_fits = list()
  for(m in 1:8){
    em_fits[[m]] = lapply(1:nsim, function(x){
      out = fit_wham(sim_input[[m]][[x]], do.osa = FALSE, MakeADFun.silent = TRUE, retro.silent = TRUE,
      return(out)
    })
  }
  saveRDS(em_fits, file.path(here(), "misspecification_study", "misspec_demo_fits.RDS"))
}
```

Start by comparing the Mohn's rho for SSB. The misspecified models all have larger magnitude Mohn's rho for SSB than the models with the correct catch.

```
rhoSSB <- matrix(NA, nrow=8, ncol=2)
colnames(rhoSSB) <- c("Model", "rhoSSB")
for (m in 1:8){
  thisrho <- lapply(1:nsim, function(x){
```

```
    out <- mohns_rho(em_fits[[m]][[x]])[1]
  return(out)
  })
  rhoSSB[m,] <- c(m, unlist(thisrho))
}
knitr::kable(rhoSSB)
```

Model	rhoSSB
1	-0.0285103
2	-0.0417838
3	-0.0284903
4	-0.0339938
5	0.1482064
6	0.1355997
7	0.1489498
8	0.1134559

We can see which model has the best fit using the compare_wham_models function for the two sets of data.

```
correct_data_compare <- compare_wham_models(list(em_fits[[1]][[1]],em_fits[[2]][[1]],em_fits[[3]][[1]],e
```

```
##    dAIC    AIC  rho_R rho_SSB rho_Fbar
## m2  0.0 5008.4 0.2629 -0.0418   0.0573
## m1  2.4 5010.8 0.3136 -0.0285   0.0416
## m4  3.9 5012.3 0.2772 -0.0340   0.0506
## m3  4.4 5012.8 0.3136 -0.0285   0.0415
```

```
## Warning in sqrt(diag(cov)): NaNs produced
```

```
misspecified_data_compare <- out <- compare_wham_models(list(em_fits[[5]][[1]],em_fits[[6]][[1]],em_fit
```

```
##    dAIC    AIC  rho_R rho_SSB rho_Fbar
## m4  0.0 5065.0 0.3632  0.1135  -0.1131
## m2  5.2 5070.2 0.3751  0.1356  -0.1486
## m1  6.5 5071.5 0.4234  0.1482  -0.1585
## m3  8.5 5073.5 0.4239  0.1489  -0.1583
```