futR

Kirstin Holsman

Contents

Repo maintained by: Kirstin Holsman
Alaska Fisheries Science Center
NOAA Fisheries, Seattle WA
kirstin.holsman@noaa.gov
Last updated: Apr 18, 2022

Overview

futR() is a generic Rpackage for fitting recruitment models to stock assessment estimates of spawning stock biomass and recruitment with or without climate covariates. The recruitment model is based on Template Model Builder (`TMB`) and formultations follow Maunder and Desriso (2011) using a generalized three parameter stock-recruitment model with environmental covariates (Deriso 1980; Schnute 1985). This includes Ricker (logistic), Beverton Holt, log-linear, and log-linear with biomass lagged by year 'y-1'. The model can be fit with and with out random effects on spawning stock biomass (SSB) and recruitment (R) (i.e., measurement error on SSB and rec) using the methods of Porch and Lauretta (2016) and with the optional unbiased estimate of sigma (sensu Ludwid and Walters 1981, Porch and Lauretta 2016). Environmental covariates are optional but can be included as main effects or as interactions.

For more information see Holsman et al. 2020 Climate and trophic controls on groundfish recruitment in Alaska.

Installing futR()

The package can be installed from github using the devtools package:

```
install.packages("devtools")
```

The projection package can then be installed to R directly:

```
devtools::install_github("kholsman/futR")
```

Setting up the package:

The base function for fitting recruitment requires a data.frame of recruitment and spawning biomass:

```
# rm(list=ls())

 #_____________________________________________
 # 1. Set things up
 #_____________________________________________

 # e.g.,
 main  <-  getwd()
 setwd(main)

 # load data, packages, setup, etc.
 source("R/make.R")

 #_____________________________________________
 # 2. Compile futR
 #_____________________________________________

 compile('src/futR.cpp') # this will generate warnings - they can be ignored if "0" is returned
```

Options

Now we can fit a set of models with and without covariates. There are various switches for fitting models:

Recruitment formulations (rectype):

1. Linear (gamma = 0)
2. Beverton Holt (gamma = -1)
3. Ricker (0 < gamma <1) ; gamma is estimated (tMethod):
   a. link = cloglog
   b. link = logit
4. Exponential (gamma=1, b<0)

Observation error options (sigMethod):

0. No observation error (tau = 0)
1. estimate sigma, random effects on SSB if tau >0, tau input

2. unbiased sigma estimate, tau input
3. as in 1 but with defined measurement error for rec (indep of random effects on Spawners/SSB)
4. as in 1 but with defined measurement error for rec and Spawners/SSB)

Link options (tMethod):

1. cloglog link (g = 1-exp(-exp(gamma)))
2. logit link (g = exp(gamma)/(1+exp(gamma)))

Environmental effects (if set to "TRUE"" in estparm):

- beta = effects on pre-lavarl/ effective number of spawners
- lamba = effects on post-spawning success (e.g., age 0+ survival)

Base model

Let's start by fitting based models (no climate covariates) with different options for observation error.

Observation error options (sigMethod):

0. No observation error (tau = 0)
1. estimate sigma, random effects on SSB if tau >0, tau input
2. unbiased sigma estimate, tau input
3. as in 1 but with defined measurement error for rec (indep of random effects on Spawners/SSB)
4. as in 1 but with defined measurement error for rec and Spawners/SSB)

0. No observation error (tau = 0)

```
# set up some demo data:
rec        <-  rec_dat[[1]]
env        <-  env_covars

# z score the covariates:
env[1,]    <-  as.numeric(scale(env_covars[1,]))
env[2,]    <-  as.numeric(scale(env_covars[2,]))
ration     <-  ration_tmb[,1]


# 3.2 Set up data
PAR$phases
PAR$estparams

# which parameters to estimate with futR?
phases  = c(
  log_a         = 1,
  log_b         = 1,
  #logit_tau      = TRUE,
  beta          = 1,
```

```
  lambda       = 1,
  epsi_s       = 1,
  logsigma     = 1)

 estparams  = c(
  log_a        = TRUE,
  log_b        = TRUE,
  #logit_tau     = TRUE,
  beta         = FALSE,
  lambda       = TRUE,
  epsi_s       = FALSE,
  logsigma     = TRUE)

# rec_noerr        <- rec
# rec_noerr$sdRobs <- 0

# makeDat will make the input values, data, and phases for the model:
datlist  <-  makeDat(
                  tauIN      =  1,
                  sigMethod  =  1, #estimate sigma, random effects on SSB if tau >0, tau input
                  tMethod    =  1,
                  estparams  =  estparams,
                  typeIN     =  4,
                  rec_years  =  rec$years,
                  Rec        =  rec$Robs,
                  SSB        =  rec$SSB,
                  sdSSB      =  rec$sdSSB,
                  sdRec      =  rec$sdRobs,
                  covars     =  NULL,
                  covars_sd  =  NULL)

  # run the basic model
wd0 <- getwd()
setwd("../src")
mm1 <-  runmod(dlistIN=datlist,version='futR',recompile=T,simulate=TRUE,simnitr = 1000)

df1 <-  data.frame(model = "mm1",
                   estimate=as.vector(mm1$sim),
                   parameter=names( mm1$mle)[row(mm1$sim)])

mu   <- df1%>%group_by(model,parameter)%>%summarise(grp.mean=mean(estimate))
peak <- df1%>%group_by(model,parameter)%>%
  count(parameter,round(estimate,1))%>%
  slice(which.max(n))
names(peak)<- c("model","parameter","freq","n")
# now plot the denisty of each parm:
p <-
  ggplot(data=df1) +
  geom_density( aes(x=estimate, color=parameter))+
  facet_wrap(~parameter,scales="free")+
  geom_vline(data=peak,aes(xintercept=freq),
             color="blue", linetype="dashed", size=1)+
  theme_kir_EBM()
```

```
p
```

1. est. sigma, rand effects on SSB if tau > 0

```
datlist$rs_dat$tau  <-  0.000001
# recall that sigMethod == 1 when creating datlist: i.e.,
# estimate sigma, random effects on SSB if tau >0, tau input

# re-run the model with tau
mm1_t0 <- runmod(dlistIN=datlist,version="futR",recompile=F,simulate=TRUE)
df1_t0 <- data.frame(
  estimate = as.vector(mm1_t0$sim),
  parameter = names( mm1_t0$mle)[row(mm1_t0$sim)])



df1_t0 <-  data.frame(model = "mm1_t0",
                      estimate=as.vector(mm1_t0$sim),
                      parameter=names( mm1_t0$mle)[row(mm1_t0$sim)])
df <- rbind(df1, df1_t0)
mu   <- df1%>%group_by(model,parameter)%>%summarise(grp.mean=mean(estimate))
peak <- df1%>%group_by(model,parameter)%>%
  count(parameter,round(estimate,1))%>%
  slice(which.max(n))
names(peak)<- c("model","parameter","freq","n")
# now plot the denisty of each parm:
p <-
  ggplot(data=df) +
  geom_density( aes(x=estimate, color=model))+
  facet_wrap(~parameter,scales="free")+
  geom_vline(data=peak,aes(xintercept=freq, color = model), linetype="dashed", size=1)+
  theme_kir_EBM()
p
```

2. unbiased sigma estimate, tau input

```
datlist$rs_dat$tau  <-  0.000001
# recall that sigMethod == 1 when creating datlist: i.e.,
# estimate sigma, random effects on SSB if tau >0, tau input

# re-run the model with tau
mm1_t0 <- runmod(dlistIN=datlist,version="futR",recompile=F,simulate=TRUE)
df1_t0 <- data.frame(
  estimate = as.vector(mm1_t0$sim),
  parameter = names( mm1_t0$mle)[row(mm1_t0$sim)])



df1_t0 <-  data.frame(model = "mm1_t0",
```

```
                    estimate=as.vector(mm1_t0$sim),
                    parameter=names( mm1_t0$mle)[row(mm1_t0$sim)])
df <- rbind(df1, df1_t0)
mu   <- df1%>%group_by(model,parameter)%>%summarise(grp.mean=mean(estimate))
peak <- df1%>%group_by(model,parameter)%>%
  count(parameter,round(estimate,1))%>%
  slice(which.max(n))
names(peak)<- c("model","parameter","freq","n")
# now plot the denisty of each parm:
p <-
  ggplot(data=df) +
  geom_density( aes(x=estimate, color=model))+
  facet_wrap(~parameter,scales="free")+
  geom_vline(data=peak,aes(xintercept=freq, color = model), linetype="dashed", size=1)+
  theme_kir_EBM()
p
```