centered_noncentered_RTMB

Overview

This document summarizes model comparisons between centered and non-centered parameterizations of AR(1) recruitment deviations. We explore models with and without an estimated mean recruitment parameter (as a fixed effect); models with and without the intercept give slightly different results, however our focus is in comparing the centered versus non-centered parameterization of random effects for each model type.

Data and model overview

We embedded the centered and non-centered recruitment deviations within a RTMB version of SAM, extending code originally written by Anders Nielsen. The case study is built off of an existing SAM model and the data list (`dat`) can be updated for RTMB as follows

```
library(RTMB)
library(stockassessment)
library(tictoc)

load("../fit.RData")
dat <- list()
dat$logobs <- fit$data$logobs
dat$aux <- fit$data$aux
dat$idx1 <- fit$data$idx1
dat$idx2 <- fit$data$idx2
dat$minYear <- min(fit$data$years)
dat$minAge <- min(fit$data$minAgePerFleet)
dat$fleetTypes <- fit$data$fleetTypes
dat$sampleTimes <- fit$data$sampleTimes
dat$year <- fit$data$years
dat$age <-  min(fit$data$minAgePerFleet):max(fit$data$maxAgePerFleet)
dat$M <- fit$data$natMor
```

```
dat$SW <- fit$data$stockMeanWeight
dat$MO <- fit$data$propMat
dat$PF <- fit$data$propF
dat$PM <- fit$data$propM

dat$srmode <- 0
dat$fcormode <- 2
dat$keyF <- fit$conf$keyLogFsta[1,]
dat$keyQ <- fit$conf$keyLogFpar
dat$keySd <- fit$conf$keyVarObs
dat$keySd[dat$keySd<(-.1)] <- NA
dat$covType <- c(0,1,2)
dat$keyIGAR <- fit$conf$keyCorObs
dat$keyIGAR[fit$conf$keyCorObs==-1] <- NA
dat$keyIGAR[is.na(fit$conf$keyCorObs)] <- -1
dat$keyIGAR[2, 1:4]<-0
dat$noParUS <- sapply(1:length(dat$fleetTypes),
                      function(f){
                        A <- sum(!is.na(dat$keySd[f,]))
                        ifelse(dat$covType[f]==2, (A*A-A)/2, 0)
                      })
```

Parameter section

The parameters estimated in the RTMB code appear below; the parameters that matter most between implementations of the centered and non-centered recruitment deviations are the last 2 (`z` and `rec_intercept`).

```
# Parameter section
par <- list()
par$logsdR <- 0
par$logsdS <- 0
par$logsdF <- numeric(max(dat$keyF)+1)
par$rickerpar <- if(dat$srmode==1){c(1,1)}else{numeric(0)}
par$transRhoF <- if(dat$fcormode==0){numeric(0)}else{0.1}
par$bhpar <- if(dat$srmode==2){c(1,1)}else{numeric(0)}
par$logQ <- numeric(max(dat$keyQ, na.rm=TRUE)+1)
par$logsd <- numeric(max(dat$keySd, na.rm=TRUE)+1)
par$logIGARdist <- numeric(max(dat$keyIGAR, na.rm=TRUE)+1)
par$parUS <- numeric(sum(dat$noParUS))
par$logN <- matrix(0, nrow=length(dat$year), ncol=length(dat$age))
```

```
par$logF <- matrix(0, nrow=length(dat$year), ncol=max(dat$keyF)+1)
par$missing <- numeric(sum(is.na(dat$logobs)))
par$tPhi <- 1  # AR phi for recruitment
par$z <- rep(0, length(dat$year))  # Standard normal innovations
par$rec_intercept <- 0 # mapped off if not estimated
```

Code

RTMB helper functions and code for the main model appear below. This script is general so that the recruitment intercept and centered / non-centered parameterization can be turned on / off with flags.

```
itrans <- function(x){
  2/(1 + exp(-2 * x)) - 1;
}

ssbFUN <- function(logN, logFF, M, SW, MO, PF, PM){
  nrow <- nrow(logN)
  ncol <- ncol(logN)
  ret <- numeric(nrow)
  for(y in 1:nrow){
    for(a in 1:ncol){
      ret[y] = ret[y]+
        SW[y,a]*MO[y,a]*exp(logN[y,a])*exp(-PF[y,a]*exp(logFF[y,a])-
                                             PM[y,a]*M[y,a])
    }
  }
  return(ret);
}

jnll <- function(par){
  getAll(par, dat)

  logobs <- OBS(logobs)

  nobs <- length(logobs)
  nrow <- nrow(M)
  ncol <- ncol(M)

  sdR <- exp(logsdR)
  sdS <- exp(logsdS)
```

```
sdF <- exp(logsdF)
sd <- exp(logsd)

phi <- 2*plogis(tPhi)-1 # AR(1) parameter

logobs <- logobs+numeric(1) ## hack to make advector
logobs[is.na(logobs)] <- missing  ##patch missing

logFF <- logF[,keyF+1] ## expand F


ssb <- ssbFUN(logN,logFF,M,SW,MO,PF,PM)

jnll <- 0

# NON-CENTERED PARAMETERIZATION
if(ar1code == -1){
  # likelihood for standard normal innovations
  jnll <- jnll - dnorm(z[1], 0,
                       sqrt(sdR*sdR/(1-phi*phi)),
                       log=TRUE)
  jnll <- jnll - sum(dnorm(z[-1], 0,
                           sdR,
                           log=TRUE))

  # transform innovations (z) to get recruitment deviations
  rec_dev <- numeric(nrow)
  rec_dev[1] <- z[1]  # initial condition
  for(i in 2:nrow){
    rec_dev[i] <- phi * rec_dev[i-1] + z[i]
  }

  # set recruitment as random walk + transformed deviations
  for(y in 1:nrow){
    if(y > 1) {
      thisSSB <- ifelse((y-minAge-1)>(-.5),
                        ssb[y-minAge],
                        ssb[1])
    }

    if(srmode==0){
      # random walk with optional intercept
```

```
      logN[y,1] <- rec_intercept + rec_dev[y]
    }
    if(srmode==1){
      # Ricker: predicted recruitment + AR(1) deviation + intercept
      predN <- rickerpar[1] + log(thisSSB) - exp(rickerpar[2])*thisSSB
      logN[y,1] <- predN + rec_intercept + rec_dev[y]
    }
    if(srmode==2){
      # Beverton-Holt: predicted recruitment + AR(1) deviation + intercept
      predN <- bhpar[1] + log(thisSSB) - log(1.0 + exp(bhpar[2])*thisSSB)
      logN[y,1] <- predN + rec_intercept + rec_dev[y]
    }
  }
}

# CENTERED PARAMETERIZATION
if(ar1code == 0){
  # AR1 init conditions
  jnll <- jnll - dnorm(logN[1,1], rec_intercept, sdR/sqrt(1-phi^2), log=TRUE)

  for(y in 2:nrow){
    thisSSB <- ifelse((y-minAge-1)>(-.5),
                      ssbFUN(logN,logFF,M,SW,MO,PF,PM)[y-minAge],
                      ssbFUN(logN,logFF,M,SW,MO,PF,PM)[1])

    if(srmode==0){
      predN <- logN[y-1,1]
    }
    if(srmode==1){
      predN <- rickerpar[1] + log(thisSSB) - exp(rickerpar[2])*thisSSB
    }
    if(srmode==2){
      predN <- bhpar[1] + log(thisSSB) - log(1.0 + exp(bhpar[2])*thisSSB)
    }

    # Include intercept in the AR process
    jnll <- jnll - dnorm(logN[y,1],
                         rec_intercept + phi*(predN - rec_intercept),
                         sdR, TRUE)
  }
}
```

```
ssb <- ssbFUN(logN, logFF, M, SW, MO, PF, PM)

# Remaining N matrix
for(y in 2:nrow){
  for(a in 2:ncol){
    predN <- logN[y-1,a-1] - exp(logFF[y-1,a-1]) - M[y-1,a-1]
    if(a==ncol){
      predN <- log(exp(predN) + exp(logN[y-1,a] - exp(logFF[y-1,a]) - M[y-1,a]))
    }
    jnll <- jnll - dnorm(logN[y,a], predN, sdS, TRUE)
  }
}

# F part
SigmaF <- matrix(0, ncol(logF), ncol(logF))

if(fcormode==0){
  diag(SigmaF) <- sdF*sdF
}

if(fcormode==1){
  diag(SigmaF) <- sdF*sdF
  rhoF <- itrans(transRhoF[1])
  for(i in 2:ncol(logF)){
    for(j in 1:(i-1)){
      SigmaF[i,j] <- rhoF*sdF[i]*sdF[j]
      SigmaF[j,i] <- SigmaF[i,j]
    }
  }
}

if(fcormode==2){
  diag(SigmaF) <- sdF*sdF
  rhoF <- itrans(transRhoF[1])
  for(i in 2:ncol(logF)){
    for(j in 1:(i-1)){
      SigmaF[i,j] <- sdF[i]*sdF[j]*(rhoF^(i-j))
      SigmaF[j,i] <- SigmaF[i,j]
    }
  }
}
```

```
for(y in 2:nrow){
  jnll <- jnll - dmvnorm(logF[y,], logF[y-1,], SigmaF, log=TRUE)
}

logPred <- numeric(nobs)
for(i in 1:nobs){
  y <- aux[i,1] - minYear + 1
  f <- aux[i,2]
  a <- aux[i,3] - minAge + 1
  Z <- exp(logFF[y,a]) + M[y,a]
  if(fleetTypes[f]==0){
    logPred[i] <- logN[y,a] - log(Z) + log(1-exp(-Z)) + logFF[y,a]
  }
  if(fleetTypes[f]==2){
    logPred[i] <- logQ[keyQ[f,a]+1] + logN[y,a] - Z*sampleTimes[f]
  }
}

Svec <- list()
for(f in 1:nrow(idx1)){
  thisdim <- sum(!is.na(keySd)[f,])
  S <- matrix(0, thisdim, thisdim)
  if(covType[f]==0){
    diag(S) <- sd[na.omit(keySd[f,])+1]^2
  }
  if(covType[f]==1){
    dist <- numeric(thisdim)
    d=2;
    for(a in 1:ncol(keyIGAR)){
      if(!is.na(keyIGAR[f,a])){
        dist[d] <- dist[d-1] + exp(logIGARdist[keyIGAR[f,a]+1])
        d <- d+1
      }
    }
    sdvec <- sd[na.omit(keySd[f,])+1]
    for(i in 1:nrow(S)){
      for(j in 1:(i-1)){
        S[i,j] <- sdvec[i]*sdvec[j]*(0.5^(dist[i]-dist[j]));
        S[j,i] <- S[i,j];
      }
      S[i,i] <- sdvec[i]^2;
    }
```

```
    }
    if(covType[f]==2){
      sdvec <- numeric(thisdim);
      d <- 1;
      for(a in 1:ncol(keySd)){
        if(!is.na(keySd[f,a])){
          sdvec[d] <- sd[keySd[f,a]+1]
          d <- d+1
        }
      }
      from <- 1
      ii <- 1
      while(ii<f){from=from+noParUS[ii]; ii<-ii+1}
      thispar <- parUS[from:(from+noParUS[f]-1)]
      U <- diag(thisdim)
      U[upper.tri(U)] <- thispar
      R <- cov2cor(t(U)%*%(U))
      D <- diag(sdvec)
      S <- D%*%R%*%D
    }
    Svec[[f]] <- S;
  }

  for(f in 1:nrow(idx1)){
    for(y in 1:ncol(idx1)){
      if(!is.na(idx1[f,y])){
        idx <- (idx1[f,y]+1):(idx2[f,y]+1)
        jnll <- jnll - dmvnorm(logobs[idx], logPred[idx], Svec[[f]], log=TRUE)
      }
    }
  }

  REPORT(logPred)
  ADREPORT(ssb)
  if(ar1code == -1) {
    ADREPORT(rec_dev)  # recruitment deviations
  }
  ADREPORT(logN[,1])  # report recruitment
  ADREPORT(rec_intercept)  # report intercept
  jnll
}
```

Mapping

In addition to the recruitment intercept being turned on / off, the first column of `logN` is not needed for the non-centered model.

```
create_map_list <- function(est_rec_intercept = FALSE, ar1code = 0) {
  map_list <- list(
    logsdF = as.factor(rep(0, length(par$logsdF)))
  )

  # Map off rec_intercept if not estimated
  if (!est_rec_intercept) {
    map_list$rec_intercept <- as.factor(NA)
  }

  # For non-centered version, map off rec column of logN
  if (ar1code == -1) {
    nyr <- length(dat$year)
    nage <- length(dat$age)
    logN_map <- matrix(1:(nyr * nage), nyr, nage)
    logN_map[,1] <- NA  # Map off recruitment
    map_list$logN <- as.factor(logN_map)
  }

  return(map_list)
}
```

Models

1. Centered model without intercept

```
dat$ar1code <- 0
n_iter <- 50
dat$est_rec_intercept <- FALSE
par_centered <- par
par_centered$z <- NULL  # remove z for centered version

obj_centered_no_int <- MakeADFun(jnll, par_centered,
                                 random=c("logN", "logF", "missing"),
                                 map=create_map_list(est_rec_intercept = FALSE,
                                                     ar1code = 0),
```

```
                                  silent=TRUE)

opt_centered_no_int <- nlminb(obj_centered_no_int$par,
                              obj_centered_no_int$fn,
                              obj_centered_no_int$gr,
                              control=list(eval.max=1000, iter.max=1000))
sdr_centered_no_int <- sdreport(obj_centered_no_int)
```

2. Centered model without intercept

```
dat$est_rec_intercept <- TRUE

obj_centered_int <- MakeADFun(jnll, par_centered,
                              random=c("logN", "logF", "missing"),
                              map=create_map_list(est_rec_intercept = TRUE,
                                                  ar1code = 0),
                              silent=TRUE)

opt_centered_int <- nlminb(obj_centered_int$par,
                           obj_centered_int$fn,
                           obj_centered_int$gr,
                           control=list(eval.max=1000, iter.max=1000))
sdr_centered_int <- sdreport(obj_centered_int)
```

3. Non-centered version without intercept

```
dat$ar1code <- -1
dat$est_rec_intercept <- FALSE

obj_noncentered_no_int <- MakeADFun(jnll, par,
                                    random=c("z", "logN", "logF", "missing"),
                                    map=create_map_list(est_rec_intercept = FALSE,
                                                        ar1code = -1),
                                    silent=TRUE)
opt_noncentered_no_int <- nlminb(obj_noncentered_no_int$par,
                                 obj_noncentered_no_int$fn,
                                 obj_noncentered_no_int$gr,
                                 control=list(eval.max=1000,
```

```
                                                iter.max=1000))
sdr_noncentered_noint <- sdreport(obj_noncentered_no_int)
```

4. Non-centered version with intercept

```
dat$est_rec_intercept <- TRUE

obj_noncentered_int <- MakeADFun(jnll, par,
                                 random=c("z", "logN", "logF", "missing"),
                                 map=create_map_list(est_rec_intercept = TRUE,
                                                     ar1code = -1),
                                 silent=TRUE)

opt_noncentered_int <- nlminb(obj_noncentered_int$par,
                              obj_noncentered_int$fn,
                              obj_noncentered_int$gr,
                              control=list(eval.max=1000, iter.max=1000))
sdr_noncentered_int <- sdreport(obj_noncentered_int)
```

Comparison

For each group of models (with and without the fixed effect recruitment mean), we can compare models with the centered and non-centered parameterization. We compared (1) likelihoods, (2) fixed effect parameter means, (3) fixed effect parameter variances, and (4) estimated recruitment deviation mean and variances. For all comparisons, we found parameters to be perfectly correlated between the centered and non-centered parameterizations.

The only noticeable difference between these formulations appeared to be in the speed. From the base case we generated new datasets (n = 50) to better understand whether the centered or non-centered parameterization was faster. We calculated the mean and standard deviation of computation time for each set of iterations (summarized in table below). This indicates that in general, the non-centered parameterizations are faster than the centered equivalents.

Intercept	Model	Avg..time..s.	SD
Y	Centered	4.206	0.554
N	Centered	4.510	0.584
Y	Non-centered	2.664	1.310
N	Non-centered	3.216	1.020