# Florian Diekert (Uni Oslo)
# NFA cod simulation model (parameter values given, except recruitment from ICES2010 Table 3.25)
# fixed selectivity
# version of 2013-03-16

```r
topologies <- list(s=all, T=TRUE)
```

# clear workspace!

```r
rm(list = ls(all = TRUE))
```

# setwd("...")

### PARAMETERS
###-----------------###
## biological parameters
Nagecl <- 11  # Number of different age-classes
w <- c(0.274625, 0.658875, 1.288125, 2.15275, 3.291875, 4.764, 6.705125, 9.24725, 10.854, 12.731, 14.311)
lowQuart <- c(355536.75, 266694.75, 180205.50, 109323.25, 50394.50, 20899.50, 7440.75, 2602.00, 925.50, 260.25, 172.25)
upQuart <- c(728188.75, 569061.00, 401538.00, 236720.75, 138221.25, 81520.25, 45641.50, 24631.75, 12618.75, 5687.50, 6671.00)
W.LOW <- c(0.3431579, 0.7247368, 1.3414737, 2.2091053, 3.4078947, 4.8084737, 6.6426316, 9.2278421, 10.9737895, 11.9896316, 14.1471579)  # LOW refers to low abundance (which gives large weight values)
W.MID <- c(0.3333947, 0.6444737, 1.1728421, 1.8188158, 2.8082632, 4.1282105, 5.6433158, 7.4938158, 9.0431316, 10.7897895, 12.8810789)
W.HI <- c(0.3440526, 0.6695263, 1.1615789, 1.7615263, 2.3604211, 3.0888421, 3.9307368, 4.7911053, 5.5721053, 6.5447368, 9.2048421)
# maturity
mat <- c(0.0014375, 0.008375, 0.06725, 0.3065625, 0.6415625, 0.846625, 0.9596875, 0.992875, 1, 1, 1)
# Phi: Survival probability matrix
nat.mort <- 0.2  # survival probability from age-class s to s+1; s < S
nat.mort.old <- 0.5  # survival probability of oldest age-class S
Phi <- matrix(0, Nagecl, Nagecl)
Phi[seq(2, Nagecl * Nagecl, by = (Nagecl + 1))] <- 1 - nat.mort
Phi[Nagecl, Nagecl] <- 1 - nat.mort.old
###-----------------###
## economic parameters
delta <- 1/(1+0.05)  # discount factor
price <- c(11, 11, 14.5, 14.5, 16, 16, 18.5, 18.5, 18.5, 18.5, 18.5) * 1000
cost <- 63.46109
Numberboats <- 200  # N boats
q <- 5.582687 * 88
alpha <- 0.9023827
gamma <- c(0.7325028, 0.8746363, 0.9320400, 0.9494212, 0.9563457, 0.9568038, 0.9480402, 0.9366465, 0.9220681, 0.9156223, 0.9277637)
# clear mQv (not.mort, nat.mort.old, BM, isec, linear, logspec, logSSB, Ricker, SSB)
## read in average status quo profit for comparison (for all 7 biomodels)
```

# Annuity <- read.table(file="datafiles/cod_barents_Numbers2.txt",header=T)
AbunTS <- c(554414, 246213, 361020, 153619, 93689, 41351, 11183, 4252, 1052, 317, 165)
# recruitment, data taken from ICES 2010 table 3.25, p.209
ices <- read.table("datafiles/ICES_VPA.txt",header=FALSE)
colnames(ices) <- c("year", "recruits", "totalbio", "totspbio", "landings", "yield.ssb", "fbar5-10")
# the list of observed recruitments for random Recruitment scenario
obsRecr <- ices$recruits
# estimate the recruitment relationships
SSB <- ices$totspbio[1946:2006-1946+1]/1000
Recruits <- ices$recruits[1949:2009-1946+1]/1000
logSSB <- log(SSB)
logrec <- log(Recruits)
# first: linear model (straight line going through the origin)
linear <- lm(Recruits ~ 0 + SSB)
# these is the coefficient for the linear function
lin_slope=coef(linear)
# second: Beverton-Holt model
BH <- nls(Recruits~(exp(a)*SSB/(1+exp(b)*SSB)), start=list(a=2,b=-5))
# these are the coefficients for the BH function
aBH <- as.double(coef(BH)[1])
bBH <- as.double(coef(BH)[2])
resBH <- residuals(BH)
# third: Ricker model
Ricker <- nls(Recruits~(a*SSB*exp(b*SSB)), start=list(a=2,b=0))
# these are the coefficients for the Ricker function
aRicker <- as.double(coef(Ricker)[1])
bRicker <- as.double(coef(Ricker)[2])
resRicker <- residuals(Ricker)
###-----------------###
# WLD <- c(0.804437, 0.7644737, 0.794987, 0.794987, 0.794987, 0.794987, 0.794987, 0.794987, 0.794987, 0.794987, 0.794987)
WLD <- c(0.804437, 0.6444737, 0.1728421, 0.8188158, 4.1282105, 5.6433158, 7.4938158, 9.0431316, 10.7897895, 12.8810789)
# Mat: maturity matrix
mat <- c(0.0014375, 0.008375, 0.06725, 0.3065625, 0.6415625, 0.846625, 0.9596875, 0.992875, 1, 1, 1)
# Phi: Survival probability matrix
nat.mort <- 0.2  # survival probability from age-class s to s+1; s < S
nat.mort.old <- 0.5  # survival probability of oldest age-class S
Phi <- matrix(0, Nagecl, Nagecl)
Phi[seq(2, Nagecl * Nagecl, by = (Nagecl + 1))] <- 1 - nat.mort
Phi[Nagecl, Nagecl] <- 1 - nat.mort.old
###-----------------###
## alternative source this file to get parameters from scratch:
#source('code/parameters_model_mac2.R')
## remove superfluous parameters from wd
rm(Qv, nat.mort, nat.mort.old, BH, isec, linear, logspec, logSSB, Ricker, SSB)
## read in average status quo profit for comparison (for all 7 biomodels)
SQprof <- c(160230189316, 1.82458e+11, 188311112579, 166101229949, 1.25455e+11 , 143963165016, 139222221810)

SQprof <- SQprof/( sum(delta^(0:99)))

# END PARAMETERS

### SIMULATIONS
## Technical parameters for simulation
T <- 50 #Length of time horizon
Nagecl <- 11  #Number of different age-classes
Simul <-500 #N simulations

###-----------------###
ttbe.store <- matrix(NA,7,Simul) # number of columns is number of biological models simulated
ttss.store <- matrix(NA,7,Simul)
sq.ttbe.store <- matrix(NA,7,Simul) # number of columns is number of biological models simulated
sq.ttss.store <- matrix(NA,7,Simul)

for(mod in c(1,7)){
  modelstore <- c("ran","lin","BH","Ric","randdw","BHddw","Ricddw")
  # different simulation scenarios
  biomodel <- modelstore[mod] # or "lin","BH","Ric","randdw","BHddw","Ricddw"
  harvestrule <- "feedback" # or "HCR","fixed","escap"
  # control space:
  A  <- 51 #N of different policies to evaluate for a given rule
  Selpol <- 1:Nagecl # different selectivity policies
  CS  <- A*length(Selpol) # control-space dimension
  # define f, the set of different policies to evaluate "feedback" # or "HCR","fixed","escap"
  if(harvestrule=="feedback"){
    a.min <- .8 #Min effort proportion
    a.max <- .12 #Max effort proportion
  }
  if(harvestrule == "HCR"){
    a.min <- .1 #Min fishing mortality
    a.max <- .3 #Max fishing mortality
  }
  if(harvestrule == "fixed"){
    a.min <- 50000 #Min effort
    a.max <- 500000 #Max effort
  }
  if(harvestrule == "escap"){
    a.min <- 4000000 #Min escapement level
    a.max <- 14000000 #Max escapement level
  }
  f <- seq(a.min,a.max,length=A)
  count <- 1 #indexing the policy scenarios for later reference
  # creating the matrix in which the policy scenarios will be recorded
  SRef <- matrix(NA,(length=CS),7)
  colnames(SRef) <- c("mean NPV", "(std.dev)", "epol", "effort", "select", "harvest","biomass")
  # creating the matrix in which profit path is stored
  ProfPath <- matrix(NA,Nagecl,T)
  # initial values for recording best and specific Biomass, Harvest
  maxNPV  <- 1
  localmaxNPV <- rep(1,Nagecl)
  specific  <- sample(Simul,1)  # which specific simulation will be recorded
  ### For each effort policy,
  for (a in 1:A) {
    ### For each selectivity scenario,
    for (i in Selpol) {
      sel <- Sel[i,]
      # initial values for on-line calculation of mean and variance
      meanNPV  <-  0
      meanSelect <- 0
      meanEffort <- 0
      meanBiomass <- matrix(0,Nagecl,T)
      meanHarvest <- matrix(0,Nagecl,T)
      meanProfits <- rep(0,T)
      ttbe <- rep(NA,length=Simul)
      ttss <- rep(NA,length=Simul)
      M2  <-  0
      online  <- 0
      ### For each simulation
      for (s in 1:Simul) {
        # creating space for the local slices
        Numbers <- matrix(NA,Nagecl,T)
        Biomass <- matrix(NA,Nagecl,T)
        Harvest <- matrix(NA,Nagecl,T)
        Profits <- rep(NA,length=T)
        discPrf <- rep(NA,length=T)
        Effortpath <- rep(NA,length=T)
      }
    }
  }
}
# initial population randomly drawn from observed stock
n <- AbunTS  ## here only one vector, in full simu:
n <- as.double(AbunTS[sample(62,1),2:12])

### For each time step
for (t in 1:T) {
### ABUNDANCE - survival(graduation) and recruitment
Numbers[,t] <- n  # record numbers
n.begin <- as.vector(Phi%*%n)  # calculate the new n-vector

### recruitment
if(biomodel=="ran"|biomodel=="randdw") n.begin[1] <- sample(obsRecr,1)
if(biomodel=="lin")
if(t < 4){n.spawn <- n*mat}else{n.spawn  <- Numbers[,t-3]*mat}
   if(out2 < 1){out2 <- 1}else{out2 <- out2}
n.begin[1] <- as.double(lin_slope)*spawn

### Beverton-Holt recruitment
if(biomodel=="BH"|biomodel=="BHddw")
   if(t < 4){n.spawn <- n*mat}else{n.spawn  <- Numbers[,t-3]*mat}
   SSB <- spawn/1000
   if(out2 < 1){out2 <- 1}else{out2 <- out2}
   n.begin[1] <- out2*1000

### Ricker recruitment
if(biomodel=="Ric"|biomodel=="Ricddw")
   if(t < 4){n.spawn <- n*mat}else{n.spawn  <- Numbers[,t-3]*mat}
   SSB <- spawn/1000
   if(out2 < 1){out2 <- 1}else{out2 <- out2}
   n.begin[1] <- out2*1000

### halftime growth
if(biomodel=="randdw"|biomodel=="BHddw"|biomodel=="Ricddw")
for(n.age in 1:11){
   if(n.begin[n.age] > lowQuart[n.age])
      w[n.age] <- W.MID[n.age]
   else{w[n.age] <- W.HI[n.age]}
}  
x <- n.begin*w  # calculates biomass-vector in tonnes
Biomass[,t] <- x  # record the age-specific at beginning of t

### harvest
if(harvestrule=="feedback")
effort <- f[a]*sum(x)
if(harvestrule=="HCR")
e.min <- 1500 # min effort in sample, used as lower bound for optimization
   e.mean <- 90000 # mean effort in sample, used as ini value for optimization
   e.max <- 2*500000 # 2 times max effort in sample, used as upper bound
   if(t < 4){n.spawn <- n*mat}else{n.spawn  <- Numbers[,t-3]*mat}
   x.spawn <- n.spawn*w
   spawn <- sum(x.spawn)
   if(spawn > Bpa){FM <- f[a]*spawn/Bpa}
   else{FM <- f[a]*spawn/Bpa}
   H.policy <- sum( x*(1-exp(-FM)) )
   ef <- e.mean
   fobj <- function(ef, q, sel, alpha, x, gamma, H.policy, Numberboats) {
      sqrt <- Numberboats*(q*sel*(ef^alpha))*(x^gamma)
      z <- (H.policy - sum(sqrt))^2
      return(z)
   }
   fit <- optimize(fobj, c(e.min,e.max), q, sel, alpha, x, gamma, H.policy, Numberboats)
   effort <- fit$minimum
if(harvestrule=="fixed")
effort <- f[a]
if(harvestrule=="escap")
e.min <- 1500 # min effort in sample, used as lower bound for optimization
   e.mean <- 90000 # mean effort in sample, used as ini value for optimization
   e.max <- 2*500000 # 2 times max effort in sample, used as upper bound
   if(sum(x)>f[a])
      H.policy  <- sum(x)-f[a]
   ef <- e.mean
   fobj <- function(ef, q, sel, alpha, x, gamma, H.policy, Numberboats) {
      sqrt <- Numberboats*(q*sel*(ef^alpha))*(x^gamma)
      z <- (H.policy - sum(sqrt))^2
      return(z)
   }
   fit <- optimize(fobj, c(e.min,e.mean), q, sel, alpha, x, gamma, H.policy, Numberboats)
   effort <- fit$minimum
}
fit <- optimize(fobj, c(e.min, e.max), q, sel, alpha, x, gamma, H.policy, Numberboats)

Effortpath(t) <- effort # recording effort

h <- Numberboats*(q*effort)^alpha # Harvest function
# check that harvest for each age-class is between zero and cohort biomass
h[which(h<0)] <- 0
h[which(h>x)] <- x[which(h>x)] # record age-specific harvest at time t

Harvest[t] <- h

# PROFITS
Profits[t] <- sum(price*h) - Numberboats*cost*effort # record instant profits
discPrf[t] <- delta^t*Profits[t] # recording discounted profits

### NEW YEAR
x.begin <- x-h
n <- x.begin/n # update the n-vector

# calculating the averages on-line
online = online + 1

# select
meanSelect = i

# Effort
del1 = Effort - meanEffort
meanEffort = meanEffort + del1/online

# NPV
del1 = NPV - meanNPV
meanNPV = meanNPV + del1/online
M2 = M2 + del*(NPV - meanNPV) # This expression uses the new value of mean
varNPV = M2/(online - 1)

# average Biomass
del1 = Biomass - meanBiomass
meanBiomass = meanBiomass + del1/online

# average Harvest
del1 = Harvest - meanHarvest
meanHarvest = meanHarvest + del1/online

# average Profitpath
del1 = Profits - meanProfits
meanProfits = meanProfits + del1/online

if (s == specific){
    specBiomass <- Biomass
    specHarvest <- Harvest
}

if (i == 1){
    if(meanNPV > maxNPV){
        maxNPV <- meanNPV
        max.avBiom <- meanBiomass
        max.avHarv <- meanHarvest
        max.specBiom <- specBiomass
        max.specHarv <- specHarvest
    }
}

if (i == 1){
    if(meanNPV > maxNPV){
        maxNPV <- meanNPV
        max.avBiom <- meanBiomass
        max.avHarv <- meanHarvest
        max.specBiom <- specBiomass
        max.specHarv <- specHarvest
    }
}

# keeping the best and one specific for status-quo select
if(maxNPV > localmaxNPV[i]){[
    localmaxNPV[i] <- meanNPV
    ProfPath[i,] <- meanProfits
}

if (i == 1){
    if(meanNPV > maxNPV){
        maxNPV <- meanNPV
        max.avBiom <- meanBiomass
        max.avHarv <- meanHarvest
        max.specBiom <- specBiomass
        max.specHarv <- specHarvest
    }
    specBiomass <- Biomass
    specHarvest <- Harvest
}

if (s == specific){
    specBiomass <- Biomass
    specHarvest <- Harvest
}

if (i == 1){
    if(meanNPV > maxNPV){
        maxNPV <- meanNPV
        max.avBiom <- meanBiomass
        max.avHarv <- meanHarvest
        max.specBiom <- specBiomass
        max.specHarv <- specHarvest
    }
}

# keeping the best and one specific for status-quo select
if(maxNPV > localmaxNPV[i]){[
    localmaxNPV[i] <- meanNPV
    ProfPath[i,] <- meanProfits
}

if (i == 1){
    if(meanNPV > maxNPV){
        maxNPV <- meanNPV
        max.avBiom <- meanBiomass
        max.avHarv <- meanHarvest
        max.specBiom <- specBiomass
        max.specHarv <- specHarvest
    }
}
# record effort- and selectivity values of the scenario
SRef[count,1] <- meanNPV
SRef[count,2] <- sqrt(varNPV)
SRef[count,3] <- f(a) #pol
SRef[count,4] <- meanEffort
SRef[count,5] <- meanSelect # note that a select value of 1 refers to a first age at capture of 3
SRef[count,6] <- mean(meanHarvest)
SRef[count,7] <- mean(meanBiomass)

# go to the next policy scenario:
count <- count + 1

} # end a-loop

} # end model loop

###-----------------###
### END SIMULATIONS
###-----------------###

### FIND BEST POLICY

top <- order(SRef[,1], decreasing=TRUE)
sq.results <- which(SRef[,5] == 1)
sq.Ref <- SRef[sq.results,]
sq.maxIndi <- which(sq.Ref == max(sq.Ref[,1]))
Outcome <- cbind(SRef[top[1],],NA,SRef[top[2],],NA,SRef[top[3],],NA,sq.Ref[sq.maxIndi,])
colnames(Outcome) <- c("best","NA", "second best", "NA","third best", "NA","sq select.")
Outcome