Stock Assessments of the Bottomfish Management Unit Species of American Samoa, the Commonwealth of the Northern Mariana Islands, and Guam, 2019

Supplementary Material

Model code and data scripts used for the 2019 territorial bottomfish benchmark assessments are provided herein as reference. Each section (model code and data scripts) is divided into three parts for each of the three territories assessed. The model section (Supplement 1. R code for running JABBA case-base and projection models) contains the code needed to run JABBA base-case and projection models, which include a separate prime and JABBA file for each territory, and three plotting files that can be used for all territories. The prime file has two locations where a user can switch between running the base-case code and running the projection code. The JABBA and plotting files need no adjustment. The data scripts section (Supplement 2. Data scripts for calculating catch time series and CPUE index) contains the R scripts used to calculate total catch for each assessment, as well as the code used to filter CPUE, standardize CPUE, and calculate the CPUE index for each assessment.

Table of Contents

Supplement 1. R code for running JABBA case-base and projection models

1.1. Guam

1.1.1. Prime file to execute model in JABBA

```
##
#Script to start base case JABBA model.
##

rm(list=ls())

# required packages
library(gplots); library(coda); library(rjags); library(R2jags); library("fitdistrplus");
library(reshape)

# Set Working directory file, where assessments are stored
File = "Location where folder for assessment is stored"

# Set working directory for JABBA R source code
JABBA.file = "Folder where the JABBA script is located"

# JABBA version
version = "v1.2_catchCV_lbs_combined"
# Set Assessment file: Data file names must match
assessment = "Guam_base_FINAL"


#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Graphic, Output, Saving (.RData) settings
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
KOBE.plot = TRUE # Produces JABBA Kobe plot
KOBE.type = c("ICCAT","IOTC")[2] # ICCAT uses 3 colors; IOTC 4 (incl. orange)
Biplot= TRUE # Produces a "post-modern" biplot with buffer and target zones (Quinn & Collie
2005)
SP.plot = c("standard","phase")[2] # Produces standard or 'Kobe phase' SP plot
save.trajectories =TRUE # saves posteriors of P=B/K, B/Bmsy and H/Hmsy as .RData object
harvest.label = c("Hmsy","Fmsy")[1] # choose label preference H/Hmsy versus Fmsy
CPUE.plot= TRUE # Runs state-tool to produce "aligned" multi-CPUE plot
meanCPUE = FALSE # Uses averaged CPUE from state-space tool instead of individual indices
Projection = TRUE # Use Projections: requires to define TACs vectors
save.projections = TRUE # saves projection posteriors as .RData object
catch.metric = "(lb)" # Define catch input metric e.g. (tons) "000 t" etc
Reproduce.seed = TRUE # If FALSE a random seed assigned to each run, if TRUE set.seed(123)
# P_bound = c(0.02,1.2)  # Soft penalty bounds for P
# Save entire posterior as .RData object
```

```
save.all = TRUE # (if TRUE, a very large R object of entire posterior is saved)
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Optional: Note Scenarios
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Specify Scenario name for output file names
Scenarios = c("catchCV",NA,NA,NA)

# Execute multiple JABBA runs in loop

for(s in 1:1){

 Scenario = Scenarios[s]

  #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
  # Surplus Production model specifications
  #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

  # Choose model type:
  # 1: Schaefer
  # 2: Fox
  # 3: Pella-Tomlinson

  Model = c(4,4,4,4)[s]
  Mod.names = c("Schaefer","Fox","Pella","Pella_m")[Model]

  # Depensation option:
  # Set Plim = Blim/K where recruitment may become impaired (e.g. Plim = 0.25)
  # Choose Plim = 0 to reduce to conventional Schaefer, Fox, Pella models
  Plim = 0

  # Required specification for Pella-Tomlinson (Model = 3/4)
  BmsyK = 0.5 # Set Surplus Production curve inflection point
  shape.CV = sqrt(log(1+0.5^2)) #Need sd on log-scale, not CV # Must be defined if Model = 4!
  #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

  #--------------------------------------------------
  # Read csv files
  #--------------------------------------------------

  # Use SEs from csv file for abudance indices (TRUE/FALSE)
  SE.I = TRUE
  CV.C = TRUE
  # Load assessment data
  catch = read.csv(paste0(File,"/",assessment,"/catch",assessment,".csv"))
```

```
cpue = read.csv(paste0(File,"/",assessment,"/cpue",assessment,".csv"))

if(SE.I ==TRUE){
  se =  read.csv(paste0(File,"/",assessment,"/se",assessment,".csv"))
}
if(CV.C == TRUE){
  cv.c = read.csv(paste0(File,"/",assessment,"/catchCV",assessment,".csv"))
}
indices2 = names(cpue)[-1]
wink.colors = data.frame(idx = indices2,
                         cols = c('#e6194b', "#3cb44b", "#ffe119",
                                  "#0082c8","#f58231", "#911eb4",
                                  "#46f0f0", "#f032e6", "#d2f53c",
                                  "#fabebe", "#008080","#e6beff", "#aa6e28")[1:length(indices2)])


#-----------------------------------------------------
# Option use mean CPUE from state-space cpue averaging
#-----------------------------------------------------
meanCPUE = FALSE

#-----------------------------------------------------
# Starting value option for r and K
#-----------------------------------------------------

#-----------------------------------------------
# Prior for unfished biomass K
#-----------------------------------------------
# The option are:
# a) Specify as a lognormal prior with mean and CV
# b) Specify as range to be converted into lognormal prior

# ><> new objective K prior
K.dist <- c("lnorm","range")[1] # ><> to range
# Get low and upper r quantile 10th and 90th
qrs <- qlnorm(c(0.025,0.975),log(0.46),sqrt(log(1+0.5^2)))
#Apply CMSY Eq 3 by Froese et al, (2017)
Klow <- 2*max(catch[,2])/qrs[2]
Khigh <- 12*max(catch[,2])/qrs[1]
#  K.prior <- c(Klow,Khigh)
K.prior <- c(4*55/0.46,0.5) #actual CV needed here

#-----------------------------------------------------------
# mean and CV and sd for Initial depletion level P1= SB/SB0
#-----------------------------------------------------------
# Set the initial depletion prior B1/K
```

```
 # To be converted into a lognormal prior (with upper bound at 1.1)

 psi.dist = c("lnorm","beta")[1]
 # specify as mean and CV
 psi.prior.mean <- signif(as.numeric(3.6556/quantile(cpue[,2], na.rm = TRUE, #
                                        probs = 0.95)), digits = 2) ## uci over 95th
# psi.prior = c(psi.prior.mean,0.5)
 psi.prior = c(0.75,0.5) #actual CV needed here
 #--------------------------------------------------------------
 # Determine estimation for catchability q and observation error
 #--------------------------------------------------------------
 # Assign q to CPUE
 sets.q = 1:(ncol(cpue)-1)

 #----------------------------------------------------
 # Determine r prior
 #----------------------------------------------------
 # The option are:
 # a) Specifying a lognormal prior
 # b) Specifying a resilience category after Froese et al. (2017; CMSY)
 # Resilience: "Very low", "Low", "Medium", High" (requires r.range = TRUE)

 # use [1] lognormal(mean,stdev) or [2] range (min,max) or
 r.dist = c("lnorm","range")[1]
#  r.prior = c(0.2735,0.3)
 r.prior = c(0.46,sqrt(log(1+0.5^2))) #needs to be sd on log-scale
 #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
 # Observation  Error
 #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

 #To Estimate additional observation variance set sigma.add = TRUE
 sigma.est = TRUE

 # Series
 sets.var = 1:(ncol(cpue)-1) # estimate individual additional variance

 # As option for data-weighing
 # minimum fixed observation error for each variance set (optional choose 1 value for both)
 fixed.obsE = c(0.0) # Important if SE.I is not available

 # Total observation error: TOE = sqrt(SE^2+sigma.est^2+fixed.obsE^2)

 #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
 # Process Error
 #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
 #Estimate set sigma.proc == True
```

```
sigma.proc = TRUE
# Determines if process error deviation are estimated for all years (TRUE)
# or only from the point the first abundance index becomes available (FALSE)
proc.dev.all = TRUE
#------------------------------------------
if(sigma.proc == TRUE){
  igamma = c(0.2,0.1) #specify inv-gamma parameters  WAS 4,0.01

  # Process error check
  gamma.check = 1/rgamma(1000,igamma[1],igamma[2])
  # check mean process error + CV
  # mu.proc = sqrt(mean(gamma.check)); CV.proc =
sd(sqrt(gamma.check))/mean(sqrt(gamma.check))

  # check CV
  # round(c(mu.proc,CV.proc),3)
  # quantile(sqrt(gamma.check),c(0.1,0.9))
}else{
  sigma.proc = 0.1 #IF Fixed: typicallly 0.05-0.15 (see Ono et al. 2012)
}
#--------------------------------------------

#><><><><><><><><><><><><><><><><>
#Include variation in catch estimates
#><><><><><><><><><><><><><><><><>
Catch.CV=TRUE

#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Optional: Do TAC Projections
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
Projection = FALSE # Switch on by Projection = TRUE
#Projection = TRUE # <---- USE THIS ONE FOR PROJECTIONS

# Set range for alternative TAC projections
TACs = seq(0,200,1)

TACint = mean(catch[nrow(catch)-2,2]:catch[nrow(catch),2]) # avg last 3 years
TACint_17 <- catch[nrow(catch),2] ##

# Set number of projections years
pyrs = 8

#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Execute model and produce output
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
```

```
  # MCMC settings
  ni <- 300000 # Number of iterations
  nt <- 5 # steps saved
  #nt <- 50 #steps saved <----USE THIS ONE FOR PROJECTIONS
  nb <- 75000 # Burn-in
  nc <- 2 # number of chains
  nsaved = (ni-nb)/nt*nc

  source(paste0(JABBA.file,"/JABBA",version,".R"))

}# THE END
```

1.1.2. Source code for JABBA model and projections

```
cat(paste0("\n","><><><><><><><><><><>"))
cat(paste0("\n","><> Run Model ",Mod.names,"<><"))
cat(paste0("\n","><><><><><><><><><><>","\n","\n"))
# setwd(paste(File))
dir.create(paste0(File,"/",assessment),showWarnings = FALSE)
dir.create(paste0(File,"/",assessment,"/",Scenario,"_",Mod.names),showWarnings = FALSE)
dir.create(paste0(File,"/",assessment,"/",Scenario,"_",Mod.names,"/Input"),showWarnings =
FALSE)
input.dir = paste0(File,"/",assessment,"/",Scenario,"_",Mod.names,"/Input")

#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
>><>
# Define objects to make sure they exist if not included in Prime file
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
>><>
if(exists("igamma")==FALSE) igamma = c(0.2,0.1)  # Generic process error prior  WAS 4,0.01
if(exists("BmsyK")==FALSE) BmsyK = 0.4  # JABBA default for Pella model
if(exists("Model")==FALSE){ model = 1; Mod.names = c("Schaefer")} # Run Schaefer if model
is not specified
if(exists("proc.dev.all")==FALSE) proc.dev.all = FALSE # process error deviation if only catch
is available
if(exists("Plim")==FALSE) Plim = 0  # Standard non-compound model
if(exists("P_bound")==FALSE) P_bound = c(0.02,1)  # Soft penalty bounds for P
if(exists("KOBE.plot")==FALSE) KOBE.plot = TRUE # Produces JABBA Kobe plot
if(exists("KOBE.type")==FALSE) KOBE.type = c("ICCAT","IOTC")[2] # ICCAT uses 3
colors; IOTC 4 (incl. orange)
if(exists("SP.plot")==FALSE) SP.plot = c("standard","phase")[2] # Produces standard or 'Kobe
phase' SP plot
if(exists("Biplot")==FALSE) Biplot= TRUE # Produces a "post-modern" biplot with buffer and
target zones (Quinn & Collie 2005)
```

```
if(exists("save.trajectories")==FALSE) save.trajectories =FALSE # saves posteriors of P=B/K,
B/Bmsy and H/Hmsy as .RData object
if(exists("harvest.label")==FALSE) harvest.label = c("Hmsy","Fmsy")[2] # choose label
preference H/Hmsy versus Fmsy
if(exists("CPUE.plot")==FALSE) CPUE.plot= TRUE # Runs state-tool to produce "alligned"
multi-CPUE plot
if(exists("catch.metric")==FALSE) catch.metric = "(t)" # Runs state-tool to produce "alligned"
multi-CPUE plot
if(exists("meanCPUE")==FALSE) meanCPUE = FALSE # Uses averaged CPUE from state-
space tool instead of individual indices
if(exists("Projection")==FALSE) Projection = FALSE # Use Projections: requires to define
TACs vectors
if(exists("save.projections")==FALSE) save.projections = FALSE# saves projection posteriors
as .RData object
if(exists("Reproduce.seed")==FALSE) Reproduce.seed = FALSE # If FALSE a random seed
assigned to each run (default)
if(exists("TACint")==FALSE) TACint = mean(catch[nrow(catch)-3,2]:catch[nrow(catch),2]) #
use mean catch from last years
if(exists("imp.yr")==FALSE) imp.yr = as.numeric(format(Sys.Date(), "%Y"))+1 # use next year
from now
if(exists("init.values")==FALSE) init.values =FALSE # Allows to add manual starting values for
K, r, q
if(exists("sigmaobs_bound")==FALSE) sigmaobs_bound = 1 # Adds an upper bound to the
observation variance
if(exists("sigmaproc_bound")==FALSE) sigmaproc_bound = 0.2 # Adds an upper bound to the
process variance
if(exists("q_bounds")==FALSE) q_bounds= c(10^-10,10) # Defines lower and upper bounds for
q
if(exists("K_bounds")==FALSE) K_bounds= c(10,10^8) # Defines lower and upper bounds for q
# Save entire posterior as .RData object
if(exists("save.all")==FALSE) save.all = FALSE #
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
>

## Prepare input data ----

cat(paste0("\n","><> Prepare input data <><","\n"))
indices = names(cpue)[2:ncol(cpue)]
n.indices = max(length(indices),1)
catches = names(catch)[2:ncol(catch)]
n.catches = length(catches)

years=catch[,1]
styr = min(years)
endyr = max(years)
n.years = length(years)
```

```
styr.cpue = min(cpue[,1])
styr.I = styr.cpue-styr+1


# Convert input data to matrices for JAGS input
conv.cpue = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.indices),styr.I-
1,n.indices),as.matrix(cpue[,-1])))
CPUE=matrix(conv.cpue,nrow=n.years,ncol=n.indices)


if(SE.I==FALSE){
  se = cpue
  conv.se = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.indices),styr.I-
1,n.indices),as.matrix(cpue[,-1])))
  se2 = matrix(ifelse(fixed.obsE>0,fixed.obsE^2,10^-10),n.years,n.indices)#/2
} else{
  conv.se = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.indices),styr.I-
1,n.indices),as.matrix(se[,-1])))
  #conv.se = sqrt(conv.se^2+fixed.obsE^2)
  se2 = matrix(ifelse(is.na(conv.se),0.3^2,conv.se)^2,n.years,n.indices)+fixed.obsE^2#/2
}

if(Catch.CV==FALSE){
  conv.catch = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.catches),styr.I-
1,n.catches),as.matrix(catch[,-1])))
  Catch=matrix(conv.catch,nrow=n.years,ncol=n.catches)
  Catch[is.na(Catch)] = 0 # Replace any NA by zero
} else{
  conv.catch = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.catches),styr.I-
1,n.catches),as.matrix(catch[,-1])))
  Catch=matrix(conv.catch,nrow=n.years,ncol=n.catches)
  Catch[is.na(Catch)] = 0 # Replace any NA by zero

  conv.catch.cv=as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.catches),styr.I-
1,n.catches),as.matrix(cv.c[,-1]))) #assumes we have a cv for every catch
  catch.cv=matrix(conv.catch.cv,nrow=n.years,ncol=n.catches)
  catch.cv[is.na(catch.cv)] = 0 # Replace any NA by zero
}


# Total Catch
TC = apply(Catch,1,sum)

# Plot Catch

cat(paste0("\n","><> Plot Catch in Input subfolder <><","\n"))
```

```
Par = list(mfrow=c(1,1),mar = c(5, 5, 1, 1), mgp =c(3,1,0), tck = -0.02,cex=0.8)
png(file = paste0(input.dir,"/Catches_",assessment,".png"), width = 7, height = 5,
    res = 200, units = "in")
par(Par)
plot(catch[,1],catch[,1],ylim=c(0,max(catch[,2:ncol(catch)],na.rm=TRUE)),ylab=paste0("Catch
",catch.metric),xlab="Year",type="n")
for(i in 2:ncol(catch)) lines(catch[,1],catch[,i],lty=(i-1),lwd=2)
legend("topright",paste(names(catch)[2:ncol(catch)]),lty=1:(ncol(catch)-1),bty="n")
dev.off()

#---------------------
# Index color palette
#---------------------
jabba.colors = as.character(rep(c('#e6194b', "#3cb44b", "#ffe119",
                                  "#0082c8","#f58231", "#911eb4",
                                  "#46f0f0", "#f032e6", "#d2f53c",
                                  "#fabebe", "#008080","#e6beff", "#aa6e28"),2))
#--------------------
# Set seed
#--------------------
if(Reproduce.seed==FALSE){ set.seed(ceiling(runif(1,min=0,max=1e6))) } else {set.seed(123)}

#-------------------------------------------------------------------------
# CPUE run State-Space model for averaging CPUE
#-------------------------------------------------------------------------
if(CPUE.plot==TRUE){
  cat(paste0("\n","><> Run State-Space CPUE averaging tool","\n"))
  #find first time-series with first CPUE
  q1.y = c(1:n.years)[is.na(apply(CPUE,1,mean,na.rm=TRUE))==FALSE][1] #first year with
CPUE
  q1.I = which.max(CPUE[q1.y,])

  qs = c(q1.I,c(1:(ncol(cpue)-1))[-q1.I])


  sink("cpueAVG.jags")
  cat("
      model {

      # Prior specifications
      eps <- 0.0000000000001 # small constant

      iq[1] ~ dgamma(1000,1000)
      q[1] <-  pow(iq[1],-1)
      logq[1] <- log(1)
```

```
    for(i in 2:nI){
    iq[i] ~ dgamma(0.001,0.001)
    q[i] <- pow(iq[i],-1)
    logq[i] <-  log(q[i])
    }


    ")

if(sigma.proc==TRUE){
  cat("
      # Process variance
      isigma2 <- isigma2.est
      sigma2 <- pow(isigma2,-1)
      sigma <- sqrt(sigma2)
      fakesigma.fixed <- sigma.fixed # Prevent unused variable error msg
      ",append=TRUE)
}else{ cat("
    isigma2 <- pow(sigma.fixed+eps,-2)
          sigma2 <- pow(isigma2,-1)
          sigma <- sqrt(sigma2)

          ",append=TRUE)}

if(sigma.est==TRUE){
  cat("
      # Observation variance
      # Observation error
      itau2~ dgamma(0.001,0.001)
      tau2 <- 1/itau2


      for(i in 1:nI)
      {
      for(t in 1:N)
      {
      var.obs[t,i] <- SE2[t,i]+tau2
      ivar.obs[t,i] <- 1/var.obs[t,i]
      # note total observation error (TOE)
      TOE[t,i] <- sqrt(var.obs[t,i])

      }}
      ",append=TRUE)
}else{ cat("
    # Observation variance
          # Observation error
```

```
        itau2~ dgamma(2,2)
        tau2 <- 1/itau2


        for(i in 1:nI)
        {
        for(t in 1:N)
        {
        var.obs[t,i] <- SE2[t,i] # drop tau2
        fake.tau[t,i] <- tau2

        ivar.obs[t,i] <- 1/var.obs[t,i]
        # note total observation error (TOE)
        TOE[t,i] <- sqrt(var.obs[t,i])

        }}

        ",append=TRUE)}

# Run rest of code
cat("
   # Process variance prior
   isigma2.est ~ dgamma(0.001,0.001)


   # Priors and constraints
   logY.est[1] ~ dnorm(logY1, 1)      # Prior for initial population size

   mean.r ~ dnorm(1, 0.001)           # Prior for mean growth rate

   # Likelihood
   # State process
   for (t in 1:(N-1)){
   r[t] ~ dnorm(mean.r, isigma2)
   logY.est[t+1] <- logY.est[t] + r[t] }

   # Observation process
   for (t in 1:N) {
   for(i in 1:nI){
   y[t,i] ~ dnorm(logY.est[t]+logq[i], ivar.obs[t,i])
   }}

   # Population sizes on real scale
   for (t in 1:N) {
   Y.est[t] <- exp(logY.est[t])
   }
```

```
  }
     ",fill = TRUE)
  sink()



  q.init = 1
  mCPUE = as.matrix(CPUE[q1.y:n.years,qs])
  mSE2 = as.matrix(se2[q1.y:n.years,qs])
  if(n.indices>1) for(i in 2:n.indices){q.init[i] =
mean(mCPUE[,i],na.rm=TRUE)/mean(mCPUE[,1],na.rm=TRUE)}
  # Bundle data
  jags.data <- list(y = log(mCPUE),SE2=mSE2, logY1 = log(mCPUE[1,1]), N =
length(q1.y:n.years),nI=n.indices,sigma.fixed=ifelse(sigma.proc==TRUE,0,sigma.proc))

  # Initial values
  inits <- function(){list(isigma2.est=runif(1,20,100), itau2=runif(1,80,200), mean.r = rnorm(1),iq
= 1/q.init)}

  # Parameters monitored
  parameters <- c("mean.r", "sigma","r", "Y.est","q")


  # Call JAGS from R (BRT 3 min)
  mod.cpue <- jags(jags.data, inits, parameters, "cpueAVG.jags", n.chains = nc, n.thin =
max(nt,2), n.iter = max(ni/5,10000), n.burnin = nb/10)


  cat(paste0("\n","><> Plot State-Space CPUE fits in Input subfolder <><","\n"))
  # get individual trends
  fitted <- lower <- upper <- NULL
  cpue.yrs = years[q1.y:n.years]

  for (t in 1:nrow(mCPUE)){
    fitted[t] <- median(mod.cpue$BUGSoutput$sims.list$Y.est[,t])
    lower[t] <- quantile(mod.cpue$BUGSoutput$sims.list$Y.est[,t], 0.025)
    upper[t] <- quantile(mod.cpue$BUGSoutput$sims.list$Y.est[,t], 0.975)}


  q.adj = apply(mod.cpue$BUGSoutput$sims.list$q,2,median)


  Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
  png(file = paste0(input.dir,"/CPUE_",assessment,"_",Scenario,".png"), width = 5, height = 3.5,
      res = 200, units = "in")
```

```
  par(Par)
  u.ylim = NULL
  for(i in 1:n.indices){ u.ylim = c(u.ylim,exp(log(mCPUE[,i]/q.adj[i])+1.96*sqrt(mSE2[,i])))}
  ylim = c(0,max(u.ylim,na.rm=TRUE))
  plot(0, 0, ylim = ylim, xlim = range(cpue.yrs), ylab = "Expected CPUE", xlab = "Year", col =
"black", type = "n")
  legend("topright",paste(indices),lwd=2,col=(jabba.colors)[1:n.indices],bty="n")
  polygon(x = c(cpue.yrs,rev(cpue.yrs)), y = c(lower,rev(upper)), col = "gray", border =
"gray90")

  for(i in 1:n.indices)
  {
    shift = runif(1,-0.1,0.1)
    cols=jabba.colors[qs[i]]

plotCI(cpue.yrs+shift,mCPUE[,i]/q.adj[i],ui=exp(log(mCPUE[,i]/q.adj[i])+1.96*sqrt(mSE2[,i])),
li=exp(log(mCPUE[,i]/q.adj[i])-1.96*sqrt(mSE2[,i])),add=TRUE,col= cols,pt.bg =
cols,pch=21,gap=0)
    lines(cpue.yrs+shift,mCPUE[,i]/q.adj[i], col = cols,lwd=2)
    points(cpue.yrs+shift,mCPUE[,i]/q.adj[i], bg = cols,pch=21)
  }
  lines(cpue.yrs,fitted,lwd=2)

  dev.off()

  logSE = apply(log(mod.cpue$BUGSoutput$sims.list$Y.est),2,sd)


  if(nrow(mCPUE)<n.years) {
    fitted = c(rep(NA,q1.y-1),fitted)
    logSE = c(rep(0.2,q1.y-1),logSE)
  }
  avgCPUE = data.frame(Year=years,CPUE= fitted,logSE=logSE)

  write.csv(avgCPUE,paste0(input.dir,"/avgCPUE_",assessment,"_",Scenario,".csv"))

  if(meanCPUE==TRUE){
    cat(paste0("\n","><> Use average CPUE as input for JABBA <><","\n"))

    CPUE = as.matrix(avgCPUE[,2])
    cpue.check = cpue[,-1]
    cpue.check[is.na(cpue[,-1])]=0
    CPUE[,1] = ifelse(apply(cpue.check,1,sum)==0,rep(NA,length(CPUE[,1])),CPUE[,1])
    se2 =  as.matrix(avgCPUE[,3]^2)
    n.indices=1
    indices = "All"
```

```
    sets.q =1
    sets.var =1
  }

  }

#------------------------------------------------------------------------
# END of CPUE State-Space tool
#------------------------------------------------------------------------

#-----------
# FUNCTIONS
#-----------
cat(paste0("\n","><> Prepare JABBA prior inputs <><","\n"))

#-------------------------------------------------
# Function to get beta prior parameters
#-------------------------------------------------
get_beta <- function(mu,CV,Min=0,Prior="x"){
  a = seq(0.0001,1000,0.001)
  b= (a-mu*a)/mu
  s2 = a*b/((a+b)^2*(a+b+1))
  sdev = sqrt(s2)
  # find beta )parameter a
  CV.check = (sdev/mu-CV)^2
  a = a[CV.check==min(CV.check)]
  #find beta parameter b
  b = (a-mu*a)/mu
  x = seq(Min,1,0.001)
  pdf = dbeta(x,a,b)
  plot(x,pdf,type="l",xlim=range(x[pdf>0.01]),xlab=paste(Prior),ylab="",yaxt="n")
  polygon(c(x,rev(x)),c(rep(0,length(x)),rev(ifelse(pdf==Inf,100000,pdf))),col="grey")
  return(c(a,b))
}

#-------------------------------------------------
# Function to get gamma prior parameters
#-------------------------------------------------

get_gamma <- function(mu,CV,Prior="x"){
  a = seq(0.0001,1000,0.0001)
  b = a/mu
  s2 = (a/b^2)
  sdev = sqrt(s2)
  # find beta )parameter a
  CV.check = (sdev/mu-CV)^2
```

```
  a = a[CV.check==min(CV.check)]
  #find beta parameter b
  b = a/mu
  x = sort(rgamma(1000,a,b))
  pdf = dgamma(x,a,b)
  plot(x,pdf,type="l",xlim=range(x[pdf>0.01]),xlab=paste(Prior),ylab="",yaxt="n")
  polygon(c(x,rev(x)),c(rep(0,length(x)),rev(ifelse(pdf==Inf,100000,pdf))),col="grey")
  return(c(a,b))
}


#-------------------------------------------------
# Function to get lognormal prior parameters
#-------------------------------------------------
plot_lnorm <- function(mu,CV,Prior="x"){
  sdev= sqrt(log(CV^2+1))
  rand.pr = rlnorm(1000,log(mu),sdev)
  x = seq(min(rand.pr),quantile(rand.pr,0.995),max(rand.pr/500))
  pdf = dlnorm(x,log(mu),sdev)
  plot(x,pdf,type="l",xlim=range(x),xlab=paste(Prior),ylab="",yaxt="n")
  polygon(c(x,rev(x)),c(rep(0,length(x)),rev(ifelse(pdf==Inf,100000,pdf))),col="grey")
  return(c(mu,sdev))
}

#-----------------------------------
# Function kobeJabba for FLR
#-----------------------------------
kobeJabba<-function(x,minyear=1){

  out=cbind(melt(x[,,2]),c(x[,,3]))
  names(out)=c("iter","year","stock","harvest")
  out$year=out$year+minyear-1
  out}

#-------------------------------------------------
# Function kobeJabbaProj for projections with FLR
#-------------------------------------------------
kobeJabbaProj<-function(x,minyear=1,tac=NULL){

  out=cbind(melt(x[,,,2]),c(x[,,,3]),c(x[,,,4]))
  names(out)=c("iter","year","tac","stock","harvest","Hrate")
  out$year=out$year+minyear-1

  out}

#-------------------------------------------------
```

```
# Determine initial ranges for r
#-------------------------------------------------
if(r.dist=="range"){
  # initial range of r based on resilience (FishBase.org)
  if(length(r.prior)>1){ start.r = r.prior} else
    if(r.prior == "High") {
      start.r <- c(0.6,1.5)} else if(r.prior == "Medium") {
        start.r <- c(0.2,0.8)}     else if(r.prior == "Low") {
          start.r <- c(0.05,0.5)}  else { # i.e. res== "Very low"
            start.r <- c(0.015,0.1)}

  log.r = mean(log(start.r))
  sd.r = abs(log.r - log(start.r[1]))/2
  r.prior = c(exp(log.r),sd.r)
  CV.r = sqrt(exp(sd.r^2)-1)
} else {
  log.r = log(r.prior[1])
  sd.r = r.prior[2]
  CV.r = sqrt(exp(sd.r^2)-1)
}

#-------------------------------------------------
# Prepare K prior
#-------------------------------------------------
if(K.dist=="range"){
  log.K = mean(log(K.prior))
  sd.K= abs(log.K - log(K.prior[1]))/2
  CV.K = sqrt(exp(sd.K^2)-1)
} else {

  log.K = log(K.prior[1])
  CV.K = K.prior[2]
  sd.K=sqrt(log(CV.K^2+1))
}


#-----------------------------------------------------------
# Get JABBA parameterization and surplus production function
#-----------------------------------------------------------
# For Pella-Tomlinson
if (Model == 3 | Model == 4) {
  ## run thru sensitivities if given shape
  # find inflection point
  ishape = NULL
  # Find shape for  SBmsytoK
  ishape = seq(0.1, 10, 0.001)
```

```
  check.shape = ((ishape) ^ (-1 / (ishape - 1)) - BmsyK) ^ 2
  #  Set shape (> 0, with 1.001 ~ Fox and 2 = Schaefer)
  shape =  ishape[check.shape == min(check.shape)]

  if (exists("sensname")) {
    if (sensname == 'M') {
      m = shape = sensmean[s] ## becomes m.mu
      shape.CV = sensvar[s]
      cat(paste0(shape, " ", shape.CV), "\n")
    } ## end sensname == M
  } ## end sensname exists
} else {shape = FALSE}


#-----------------------------------------------


# Set shape m for Fox and Schaefer: Fox m ~1; Schaefer m =2
if(shape==FALSE){
  if(Model == 1){m=2} else {m = 1.001}}else{m=shape}

cat(paste0("\n","><> Plot Prior distributions in Input subfolder  <><","\n"))

Par = list(mfrow=c(1,3),mai=c(0.5,0.1,0,.1),omi = c(0.1,0.2,0.1,0) + 0.1,mgp=c(2,1,0), tck = -
0.02,cex=0.8)
png(file = paste0(input.dir,"/Priors_",assessment,"_",Scenario,".png"), width = 9, height = 3,
    res = 200, units = "in")
par(Par)
K.pr = plot_lnorm(exp(log.K),CV.K,Prior="K")

if(psi.dist=="beta"){
  psi.pr = get_beta(mu=psi.prior[1],CV=psi.prior[2],Min=0,Prior=paste0("Prior
B(",years[1],")/K"))} else {
    psi.pr = plot_lnorm(mu=psi.prior[1],CV=psi.prior[2],Prior=paste0("Prior B(",years[1],")/K"))
  }


r.pr = plot_lnorm(mu=exp(log.r),CV=CV.r,Prior="r")
mtext(paste("Density"), side=2, outer=TRUE, at=0.5,line=1,cex=0.9)
dev.off()


cat(paste0("\n","><> Plot assumed Surplus Production shape in Input subfolder  <><","\n"))


# Plot MSY
```

```
Par = list(mfrow=c(1,1),mai=c(0.6,0.3,0,.15),omi = c(0.1,0.2,0.2,0) + 0.1,mgp=c(2,1,0), tck = -
0.02,cex=0.8)
png(file = paste0(input.dir,"/Production",assessment,"_",Scenario,".png"), width = 6, height = 5,
    res = 200, units = "in")
par(Par)

# Get Bmsy/B0 as a function of M
Bmsy=(m)^(-1/(m-1))
P = seq(0.0001,1,0.001)
SP = ifelse(P>Plim,r.pr[1]/(m-1)*P*(1-P^(m-1)),r.pr[1]/(m-1)*P*(1-P^(m-1))*4*P)
#if(is.null(refBmsy)==TRUE) refBmsy = Bmsy
plot(P,SP/max(SP),type="l",ylab="Relative Yield",xlab="B/B0",lwd=2)
mtext(paste("Relative Yield"), side=2, outer=TRUE, at=0.6,line=1,cex=0.9)
legend("topright",c("SPM"),col=c(1),lwd=2,bty="n")


if(Model==4){
  # shape density
  # dm = dgamma(seq(0.001,5,0.1),5,5)*m
  dm = dlnorm((seq(0.001,5,0.1)),log(m),shape.CV)
  dm = dm/max(dm)
  bmsyk  = (seq(0.001,5,0.1))^(-1/(seq(0.001,5,0.1)-1))

  polygon(c(bmsyk,rev(bmsyk)),c(dm,rep(0,length(dm))),col="grey",border=0)
}
abline(v=Bmsy,lty=2)
mtext(paste("Relative Yield"), side=2, outer=TRUE, at=0.6,line=1,cex=0.9)
legend("topright",c("SPM"),col=c(1),lwd=2,bty="n")
abline(v=Bmsy,lty=2)
dev.off()


# Note PRIORS and save input subfolder
Priors =rbind(K.pr,psi.prior,c(r.pr[1],CV.r))
row.names(Priors) = c("K","Psi","r")
colnames(Priors) = c("Mean","CV")
write.csv(Priors,paste0(input.dir,"/Priors",assessment,"_",Scenario,".csv"))


#------------------------------------------------------------
# Set up JABBA
#------------------------------------------------------------
cat(paste0("\n","><> Set up JAGS input <><","\n"))
# Plot MSY
# remove scientific numbers
options(scipen=999)
```

```
#-------------------------------------------------------


# starting values
nq = length(unique(sets.q))
nvar = length(unique(sets.var))

## MDFitchett change to TAC setup
#-------------------------------------------------------
# Setup TAC projection
#-------------------------------------------------------
if(Projection==TRUE) {
  nTAC = length(TACs)
  TAC = mat.or.vec(pyrs, nTAC)
  yr.now=2019
  yr.last = max(years) # assessment year

  for (i in 1:nTAC) {
    TAC[, i] = c(rep(TACint, yr.now - yr.last), rep(TACs[i], pyrs - (yr.now -yr.last)))
  }

} else if(Projection == FALSE){
  nTAC = 1
  TAC = TC[n.years]
  pyrs = 1
}

#-------------------------------------------------------------
# JABBA Schaefer/Fox Models 1-2, Pella 3
#-------------------------------------------------------------

# Slope of hockey-stick
slope.HS = ifelse(Plim==0,1/10^-10,1/Plim)


nSel = 1 # setup for JABBA-SELECT version (in prep)
nI = ncol(CPUE) # number of CPUE series
stI = ifelse(proc.dev.all==TRUE,1,
c(1:n.years)[is.na(apply(CPUE,1,mean,na.rm=TRUE))==FALSE][1]) #first year with CPUE


# Initial starting values
if(Catch.CV==FALSE){
inits <- function(){list(K=
rlnorm(1,log(4*55/0.46),0.5),r=rlnorm(1,log(0.46),0.5),m=rlnorm(1,log(2),0.5),
```

```
psi=rlnorm(1,log(0.75),0.5), q = runif(1,0.0000000001,10), isigma2.est=rgamma(1,0.2,0.1),
itau2=rgamma(1,0.2,0.1))}
} else {
# inits <- function(){list(K=4*55/0.46,r=0.46,m=2,rCatch = TC, psi=0.75, q = 5,
isigma2.est=100, itau2=100)}
inits <- function(){list(K=
rlnorm(1,log(4*55/0.46),0.5),r=rlnorm(1,log(0.46),0.5),m=rlnorm(1,log(2),0.5),rCatch = TC,
psi=rlnorm(1,log(0.75),0.5), q = runif(1,0.0000000001,10), isigma2.est=rgamma(1,0.2,0.1),
itau2=rgamma(1,0.2,0.1))}
}

# starting value option
if(init.values==TRUE){
inits <- function(){list(K= K.init,r=r.init,q = q.init, isigma2.est=runif(1,20,100),
itau2=runif(nvar,80,200))}
}

# JABBA input data
if(Catch.CV==FALSE){
surplus.dat = list(N=n.years, TC = TC, I=CPUE,SE2=se2,mu.m=m,r.pr=r.pr,psi.pr=psi.pr,K.pr =
K.pr,
                nq=nq,nI = nI,nvar=nvar,sigma.fixed=ifelse(sigma.proc==TRUE,0,sigma.proc),
                sets.var=sets.var, sets.q=sets.q,pen.bk =
rep(0,n.years),Plim=Plim,slope.HS=slope.HS,
                nTAC=nTAC,pyrs=pyrs,TAC=TAC,igamma = igamma,stI=stI,TACint
=TACint,TACint_17=TACint_17,P_bound=P_bound,proc.pen=0,K.pen = 0,
                obs.pen =
rep(0,nvar),q_bounds=q_bounds,sigmaobs_bound=sigmaobs_bound,sigmaproc_bound=sigmapr
oc_bound,K_bounds=K_bounds)
} else {
surplus.dat = list(N=n.years, TC = TC, catch.cv = catch.cv,
I=CPUE,SE2=se2,mu.m=m,r.pr=r.pr,psi.pr=psi.pr,K.pr = K.pr,
                nq=nq,nI = nI,nvar=nvar,sigma.fixed=ifelse(sigma.proc==TRUE,0,sigma.proc),
                sets.var=sets.var, sets.q=sets.q,pen.bk =
rep(0,n.years),Plim=Plim,slope.HS=slope.HS,
                nTAC=nTAC,pyrs=pyrs,TAC=TAC,igamma = igamma,stI=stI,TACint
=TACint,TACint_17=TACint_17,P_bound=P_bound,proc.pen=0,K.pen = 0,
                obs.pen =
rep(0,nvar),q_bounds=q_bounds,sigmaobs_bound=sigmaobs_bound,sigmaproc_bound=sigmapr
oc_bound,K_bounds=K_bounds)
}




# If shape parameter is estimated (Model =4)
```

```
if(Model==4){
  surplus.dat$m.CV = shape.CV }


# JAGS model file
JABBA = "JABBA.jags"


# PARAMETERS TO MONITOR -
if(Catch.CV==FALSE){
params <- c("K","r", "q", "psi","sigma2", "tau2","m","Hmsy","SBmsy", "MSY",
"BtoBmsy","HtoHmsy","Overfishing_ind","CPUE","Proc.Dev","P","SB","H","prP","prBtoBms
y","prHtoHmsy","prOverfishing_ind", "prH","TOE")
} else {
params <- c("K","r", "q", "psi","sigma2", "tau2","m","Hmsy","SBmsy", "MSY",
"BtoBmsy","HtoHmsy","Overfishing_ind","CPUE","Proc.Dev","P","SB","H","prP","prBtoBms
y","prHtoHmsy","prOverfishing_ind", "prH","TOE")
}



cat(paste0("\n","><> RUN ",Mod.names," model for ",assessment," ",Scenario," in JAGS
<><","\n","\n"))

# JAGS MODEL Standard
sink("JABBA.jags")
cat("

   model {

   # Prior specifications
   eps <- 0.00000000000000000000000000000001 # small constant

   # Catchability coefficients
   for(i in 1:nq)
   {
   q[i] ~ dunif(q_bounds[1],q_bounds[2])
   }


   ")

if(Model==4){
  cat("
     # Shape m prior
     m ~ dlnorm(log(mu.m),pow(m.CV,-2))
```

```
    ",append=TRUE)
  }else{ cat("
    m <- mu.m
   ",append=TRUE)}

if(psi.dist =="beta"){
  cat("
    # Beta Prior for Biomass depletion at the start (deterministic)
    psi ~ dbeta(psi.pr[1],psi.pr[2])
    ",append=TRUE)
} else {
  cat("
    # Lognormal for Biomass depletion at the start (deterministic)
    psi ~ dlnorm(log(psi.pr[1]),pow(psi.pr[2],-2)) #I(0.1,1.1)
    ",append=TRUE)
}

if(sigma.proc==TRUE){
  cat("
    # Process variance
    isigma2 <- isigma2.est
    sigma2 <- pow(isigma2,-1)
    sigma <- sqrt(sigma2)
    fakesigma.fixed <- sigma.fixed # Prevent unused variable error msg
    ",append=TRUE)
}else{ cat("
    isigma2 <- pow(sigma.fixed+eps,-2)
        sigma2 <- pow(isigma2,-1)
        sigma <- sqrt(sigma2)

        ",append=TRUE)}

if(sigma.est==TRUE){
  cat("
    # Observation variance
    for(i in 1:nvar)
    {
    # Observation error
    itau2[i]~ dgamma(0.2,0.1)   #from D7, was 0.001,0.001
    tau2[i] <- 1/itau2[i]
    }

    for(i in 1:nI)
    {
    for(t in 1:N)
    {
```

```
    var.obs[t,i] <- SE2[t,i]+tau2[sets.var[i]]
    ivar.obs[t,i] <- 1/var.obs[t,i]
    # note total observation error (TOE)
    TOE[t,i] <- sqrt(var.obs[t,i]) # Total observation variance

    }}
    ",append=TRUE)
}else{ cat("
    # Observation variance
        for(i in 1:nvar)
        {
        # Observation error
        itau2[i]~ dgamma(0.2,0.1)
        tau2[i] <- 1/itau2[i]
        }

        for(i in 1:nI)
        {
        for(t in 1:N)
        {
        var.obs[t,i] <- SE2[t,i] # drop tau2
        fake.tau[t,i] <- tau2[sets.var[i]]

        ivar.obs[t,i] <- 1/var.obs[t,i]
        # note total observation error (TOE)
        TOE[t,i] <- sqrt(var.obs[t,i])

        }}

        ",append=TRUE)}

# Process section divided into 2 chunks, replicated for Catch.CV = FALSE/TRUE
if(Catch.CV==FALSE){
cat("
   # Process variance prior
   isigma2.est ~ dgamma(igamma[1],igamma[2])

   # Carrying Capacity SB0
   K ~ dlnorm(log(K.pr[1]),pow(K.pr[2], -2))


   # informative priors for Hmsy as a function of r
   r ~ dlnorm(log(r.pr[1]),pow(r.pr[2],-2))


   # Process equation
```

```
   Pmean[1] <- log(psi)
   iPV[1] <- ifelse(1<(stI),10000,isigma2) # inverse process variance
   P[1] ~ dlnorm(Pmean[1],iPV[1]) # set to small noise instead of isigma2
   penB[1]  <- ifelse(P[1]<P_bound[1],log(K*P[1])-
log(K*P_bound[1]),ifelse(P[1]>P_bound[2],log(K*P[1])-log(K*P_bound[2]),0)) # penalty if
Pmean is outside viable biomass

   # Process equation
   for (t in 2:N)
   {
   Pmean[t] <- ifelse(P[t-1] > Plim,
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1)) - TC[t-1]/K,0.005)),
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1))*P[t-1]*slope.HS - TC[t-1]/K,0.005)))
   iPV[t] <- ifelse(t<(stI),10000,isigma2) # inverse process variance
   P[t] ~ dlnorm(Pmean[t],iPV[t])
   penB[t]  <- ifelse(P[t]<(P_bound[1]),log(K*P[t])-
log(K*(P_bound[1])),ifelse(P[t]>P_bound[2],log(K*P[t])-log(K*(P_bound[2])),0)) # penalty if
Pmean is outside viable biomass
   }

   # Process error deviation
   for(t in 1:N){
   Proc.Dev[t] <- P[t]-exp(Pmean[t])}

   # Enforce soft penalties on bounds for P
   for(t in 1:N){
   pen.bk[t] ~ dnorm(penB[t],1000) # enforce penalty with CV = 0.1
   }
   ", append=TRUE)
} else{ cat("
   # Process variance prior
   isigma2.est ~ dgamma(igamma[1],igamma[2])

   # Carrying Capacity SB0
   K ~ dlnorm(log(K.pr[1]),pow(K.pr[2], -2))

   # informative priors for Hmsy as a function of r
   r ~ dlnorm(log(r.pr[1]),pow(r.pr[2],-2))


# Process equation
    Pmean[1] <- log(psi)
    iPV[1] <- ifelse(1<(stI),10000,isigma2) # inverse process variance
    P[1] ~ dlnorm(Pmean[1],iPV[1]) # set to small noise instead of isigma2
```

```
    penB[1]  <- ifelse(P[1]<P_bound[1],log(K*P[1])-
log(K*P_bound[1]),ifelse(P[1]>P_bound[2],log(K*P[1])-log(K*P_bound[2]),0)) # penalty if
Pmean is outside viable biomass
    Catch_precision[1]<-1/log(1+catch.cv[1,]*catch.cv[1,])
    rCatch[1] ~ dlnorm(log(TC[1]),Catch_precision[1])

# Process equation
   for (t in 2:N)
   {
   Pmean[t] <- ifelse(P[t-1] > Plim,
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1)) - rCatch[t-1]/K,0.005)),
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1))*P[t-1]*slope.HS - rCatch[t-
1]/K,0.005)))

    iPV[t] <- ifelse(t<(stI),10000,isigma2) # inverse process variance
    P[t] ~ dlnorm(Pmean[t],iPV[t])
    Catch_precision[t]<-1/log(1+catch.cv[t,]*catch.cv[t,])
    rCatch[t] ~ dlnorm(log(TC[t]),Catch_precision[t])
    penB[t]  <- ifelse(P[t]<(P_bound[1]),log(K*P[t])-
log(K*(P_bound[1])),ifelse(P[t]>P_bound[2],log(K*P[t])-log(K*(P_bound[2])),0)) # penalty if
Pmean is outside viable biomass
    }

  # Process error deviation
   for(t in 1:N){
   Proc.Dev[t] <- P[t]-exp(Pmean[t])}

   # Enforce soft penalties on bounds for P
  for(t in 1:N){
  pen.bk[t] ~ dnorm(penB[t],1000) # enforce penalty with CV = 0.1
   }
          ", append=TRUE)

}


# Run rest of code
   cat("
   Hmsy <- r*pow(m-1,-1)*(1-1/m)

   for (t in 1:N)
   {
   SB[t] <- K*P[t]
   }"
      , append=TRUE)
```

```
# Include calculation of H for random catch
if(Catch.CV==FALSE){
  cat("
for (t in 1:N)
{
H[t] <- TC[t]/SB[t]
}
 ", append=TRUE)} else{
  cat("
for (t in 1:N)
{
H[t] <- rCatch[t]/SB[t]
}
 ", append=TRUE)}

# Observation equation in related to EB
cat("
for(i in 1:nI)
{
for (t in 1:N)
{
Imean[t,i] <- log(q[sets.q[i]]*P[t]*K);
I[t,i] ~ dlnorm(Imean[t,i],(ivar.obs[t,i]));
CPUE[t,i] <- q[sets.q[i]]*P[t]*K
}}


# Management quantities
SBmsy_K <- (m)^(-1/(m-1))
SBmsy <- SBmsy_K*K

MSY <- SBmsy*Hmsy
for (t in 1:N)
{
# use x y to put them towards the end of the alphabetically sorted  mcmc object
# SP[t] <- pow(r.pella,-(m-1))*SB[t]*(1-pow(P[t],m-1))
BtoBmsy[t] <- SB[t]/SBmsy
HtoHmsy[t] <- H[t]/(Hmsy)

#modified for territorial HCR:
Overfishing_ind[t]<-ifelse(BtoBmsy[t]>0.7, H[t]/(Hmsy),  H[t]/((Hmsy*SB[t])/(0.7*SBmsy)))
}


# Enforce soft penalty on K if < K_bounds >
K.pen ~ dnorm(penK,1000) # enforce penalty
```

```
   penK  <- ifelse(K<(K_bounds[1]),log(K)-log(K_bounds[1]),ifelse(K>K_bounds[2],log(K)-
log(K_bounds[2]),0)) # penalty if Pmean is outside viable biomass


   # Enforce soft penalty on process deviance if sigma.proc > 0.2
   proc.pen ~ dnorm(penProc,1000) # enforce penalty
   penProc  <- ifelse(sigma>sigmaproc_bound,log(sigma)-log(sigmaproc_bound),0)


   # Enforce soft penalty on observation error if sigma.obs > sigma_bound
   for(i in 1:nvar){
   obs.pen[i] ~ dnorm(penObs[i],1000) # enforce penalty
   penObs[i]  <- ifelse(pow(tau2[i],0.5)>sigmaobs_bound,log(pow(tau2[i],0.5))-
log(sigmaobs_bound),0)
   }

   ", append=TRUE)

# PROJECTION
if(Projection==TRUE){
  cat("
     for(i in 1:nTAC){
     # Project first year into the future
     prPmean[1,i] <- ifelse(P[N] > Plim,
     log(max(P[N] +  Hmsy/(1-1/m)*P[N]*(1-pow(P[N],m-1)) - TACint_17/K,0.005)),
     log(max(P[N] +  Hmsy/(1-1/m)*P[N]*(1-pow(P[N],m-1))*4*P[N] - TACint_17/K,0.005)))
     prP[1,i] ~ dlnorm(prPmean[1,i],isigma2)
     # Project all following years
     for(t in 2:pyrs){
     prPmean[t,i] <- ifelse(prP[t-1,i] > Plim,
     log(max(prP[t-1,i] +  Hmsy/(1-1/m)*prP[t-1,i]*(1-pow(prP[t-1,i],m-1)) - TAC[t-
1,i]/K,0.001)),
     log(max(prP[t-1,i] +  Hmsy/(1-1/m)*prP[t-1,i]*(1-pow(prP[t-1,i],m-1))*slope.HS*prP[t-1,i] -
TAC[t-1,i]/K,0.005)))
     # process error (as monte-carlo simular)
     prP[t,i] ~ dlnorm(prPmean[t,i],isigma2)}
     for(t in 1:pyrs){
     prB[t,i] <- prP[t,i]*K
     prH[t,i] <- TAC[t,i]/prB[t,i]
     prHtoHmsy[t,i] <- prH[t,i]/Hmsy
     prBtoBmsy[t,i] <- prB[t,i]/SBmsy
     prOverfishing_ind[t,i]<-ifelse(prBtoBmsy[t,i]>0.7, prH[t,i]/(Hmsy),
prH[t,i]/((Hmsy*prB[t,i])/(0.7*SBmsy))) #JS 2-13-19
     }}
     ",append=TRUE)} else {
       cat("
```

```
        #Prevent error for unused input
        fakeTAC <-  TAC
        fakepyrs <- pyrs
        fakenTAC <- nTAC
        fakeTACint <- TACint
        prHtoHmsy <- 1
        prP <- 1
        prBtoBmsy <- 1
        prOverfishing_ind<-1
        ", append=TRUE)}

cat("

} # END OF MODEL
   ",append=TRUE,fill = TRUE)
sink()


ptm <- proc.time()

mod <- jags(surplus.dat, inits,params,paste(JABBA), n.chains = nc, n.thin = nt, n.iter = ni,
n.burnin = nb)  # adapt is burn-in

proc.time() - ptm
save.time = proc.time() - ptm

cat(paste0("\n",paste0("><> Scenario ",Scenario,"_",Mod.names," completed in
",as.integer(save.time[3]/60)," min and ",round((save.time[3]/60-
as.integer(save.time[3]/60))*100)," sec <><","\n")))

cat(paste0("\n","><> Produce results output of ",Mod.names," model for ",assessment,"
",Scenario," <><","\n"))

# if run with library(rjags)
posteriors = mod$BUGSoutput$sims.list



#------------------------------------------------------------
# <><<><<><<><<><<><<>< Outputs ><>><>><>><>><>><>><>><>><>
#------------------------------------------------------------
output.dir = paste0(File,"/",assessment,"/",Scenario,"_",Mod.names,"/Output")
dir.create(output.dir, showWarnings = FALSE)

# run some mcmc convergence tests
par.dat= data.frame(posteriors[params[c(1:7)]])
```

```
geweke1 = geweke.diag(data.frame(par.dat[1:(nsaved/nc),]))
geweke2 = geweke.diag(data.frame(par.dat[(nsaved/nc+1):nsaved,]))
pvalues1 <- 2*pnorm(-abs(geweke1$z))
pvalues2 <- 2*pnorm(-abs(geweke2$z))

heidle1 = heidel.diag(data.frame(par.dat[1:(nsaved/nc),]))
heidle2 = heidel.diag(data.frame(par.dat[(nsaved/nc+1):nsaved,]))

autocorr1=lapply(par.dat[1:(nsaved/nc),],FUN=acf,plot=F)
autocorr2=lapply(par.dat[(nsaved/nc+1):nsaved,],FUN=acf,plot=F)
max.acK=pmax(abs(autocorr1$K$acf),abs(autocorr1$K$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acr=pmax(abs(autocorr1$r$acf),abs(autocorr2$r$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acq=pmax(abs(autocorr1$q$acf),abs(autocorr2$q$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acpsi=pmax(abs(autocorr1$psi$acf),abs(autocorr2$psi$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acsigma2=pmax(abs(autocorr1$sigma2$acf),abs(autocorr2$sigma2$acf))[c(2,6)] #max lag1
and lag5 autocorrelation across chains
max.actau2=pmax(abs(autocorr1$tau2$acf),abs(autocorr2$tau2$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acm=pmax(abs(autocorr1$m$acf),abs(autocorr2$m$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.ac=rbind(max.acK,max.acr,max.acq,max.acpsi,max.acsigma2,max.actau2,max.acm)

# postrior means + 95% BCIs
#Model  parameter
apply(par.dat,2,quantile,c(0.025,0.5,0.975))

man.dat = data.frame(posteriors[params[8:10]])
#Management quantities
apply(man.dat,2,quantile,c(0.025,0.5,0.975))

# Depletion
Depletion = posteriors$P[,c(1,n.years)]
colnames(Depletion) = c(paste0("P",years[1]),paste0("P",years[n.years]))

# Current stock status (Kobe posterior)
H_Hmsy.cur = posteriors$HtoHmsy[,c(n.years)]
B_Bmsy.cur = posteriors$BtoBmsy[,c(n.years)]


# Prepare posterior quantities
man.dat = data.frame(man.dat,Depletion,B_Bmsy.cur,H_Hmsy.cur)
```

```
results = round(t(cbind(apply(par.dat,2,quantile,c(0.025,0.5,0.975)))),6)

results = data.frame(Median =
results[,2],LCI=results[,1],UCI=results[,3],Geweke.p1=round(pvalues1,3),Geweke.p2=round(pv
alues2,3),Heidel.p1 = round(heidle1[,3],3),Heidel.p2 =
round(heidle2[,3],3),Heidel.it1=heidle1[,2],Heidel.it2=heidle2[,2],Heidel.hw.pass1=heidle1[,4]=
=1,Heidel.hw.pass2=heidle2[,4]==1,lag1=max.ac[,1],lag5=max.ac[,2]) #added all heidleberger
and welch diagnostics 10-25-2018

ref.points = round(t(cbind(apply(man.dat,2,quantile,c(0.025,0.5,0.975)))),3)

ref.points = data.frame(Median = ref.points[,2],LCI=ref.points[,1],UCI=ref.points[,3])

# get number of parameters
npar = length(par.dat)
# number of years
N=n.years

# Save posteriors (Produces large object!)
if(save.all==TRUE) save(posteriors,file=paste0(output.dir,"/",Scenario,"_posteriors"))

#------------------------------------------------------------------------
# Save parameters, results table and current status posterior in csv files
#------------------------------------------------------------------------

# Save model estimates and convergence p-values
write.csv(data.frame(results),paste0(output.dir,"/Estimates_",assessment,"_",Scenario,".csv"))

# Make standard results table with parameter estimates and reference points
Table = rbind(data.frame(results)[,1:3],data.frame(ref.points))
Table[4,] = round(sqrt((Table[4,])),3)
rownames(Table)[4] = "sigma.proc"
write.csv(Table,paste0(output.dir,"/Results_",assessment,"_",Scenario,".csv"))
#Save posterior of recent assessment year (KOBE posterior)
write.csv(data.frame(BtoBmsy=B_Bmsy.cur,FtoFmsy=H_Hmsy.cur),paste0(output.dir,"/Status_
posterior",assessment,".csv"))

## source all plotting scripts
source(paste0(JABBA.file,'/plot_JABBA_report_combined.R'))


if(save.trajectories==TRUE){
  cat(paste0("\n","><> Saving Posteriors of FRP trajectories <><","\n"))

  # FRP trajectories
```

```
  trajectories = array(NA,c(nsaved,n.years,3))
  trajectories[,,1] = posteriors$P
  trajectories[,,2] = posteriors$BtoBmsy
  trajectories[,,3] = posteriors$Overfishing_ind


  kb=kobeJabba(trajectories,years[1])
  save(kb,file=paste0(output.dir,"/",Scenario,"_trajectories"))

}

cat(paste0("\n","><> Scenario ",Mod.names,"_",Scenario," for ",assessment," - DONE!
<><","\n"))
```

1.2. CNMI

1.2.1. Prime file to execute model in JABBA

```
##
#Script to start base case JABBA model.
##

rm(list=ls())

# required packages
library(gplots); library(coda); library(rjags); library(R2jags); library("fitdistrplus");
library(reshape)

# Set Working directory file, where assessments are stored
File = "Location where folder for assessment is stored"

# Set working directory for JABBA R source code
JABBA.file = "Folder where the JABBA script is located"

# JABBA version
version = "v1.2_catchCV_lbs_combined"
# Set Assessment file: Data file names must match
assessment = "CNMI_base_FINAL"


#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Graphic, Output, Saving (.RData) settings
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
KOBE.plot = TRUE # Produces JABBA Kobe plot
KOBE.type = c("ICCAT","IOTC")[2] # ICCAT uses 3 colors; IOTC 4 (incl. orange)
```

```
Biplot= TRUE # Produces a "post-modern" biplot with buffer and target zones (Quinn & Collie
2005)
SP.plot = c("standard","phase")[2] # Produces standard or 'Kobe phase' SP plot
save.trajectories =TRUE # saves posteriors of P=B/K, B/Bmsy and H/Hmsy as .RData object
harvest.label = c("Hmsy","Fmsy")[1] # choose label preference H/Hmsy versus Fmsy
CPUE.plot= TRUE # Runs state-tool to produce "aligned" multi-CPUE plot
meanCPUE = FALSE # Uses averaged CPUE from state-space tool instead of individual indices
Projection = TRUE # Use Projections: requires to define TACs vectors
save.projections = TRUE # saves projection posteriors as .RData object
catch.metric = "(lb)" # Define catch input metric e.g. (tons) "000 t" etc
Reproduce.seed = TRUE # If FALSE a random seed assigned to each run, if TRUE set.seed(123)
# P_bound = c(0.02,1.2)  # Soft penalty bounds for P
# Save entire posterior as .RData object
save.all = TRUE # (if TRUE, a very large R object of entire posterior is saved)
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Optional: Note Scenarios
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Specify Scenario name for output file names
Scenarios = c("catchCV",NA,NA,NA)

# Execute multiple JABBA runs in loop

for(s in 1:1){

 Scenario = Scenarios[s]

  #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
  # Surplus Production model specifications
  #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

  # Choose model type:
  # 1: Schaefer
  # 2: Fox
  # 3: Pella-Tomlinson

  Model = c(1,1,1,1)[s]
  Mod.names = c("Schaefer","Fox","Pella","Pella_m")[Model]

  # Depensation option:
  # Set Plim = Blim/K where recruitment may become impaired (e.g. Plim = 0.25)
  # Choose Plim = 0 to reduce to conventional Schaefer, Fox, Pella models
  Plim = 0

  # Required specification for Pella-Tomlinson (Model = 3/4)
```

```
BmsyK = 0.5 # Set Surplus Production curve inflection point
shape.CV = sqrt(log(1+0.5^2)) #Need sd on log-scale, not CV  # Must be defined if Model = 4!
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

#------------------------------------------------
# Read csv files
#------------------------------------------------

# Use SEs from csv file for abudance indices (TRUE/FALSE)
SE.I = TRUE
CV.C = TRUE
# Load assessment data
catch = read.csv(paste0(File,"/",assessment,"/catch",assessment,".csv"))
cpue = read.csv(paste0(File,"/",assessment,"/cpue",assessment,".csv"))

if(SE.I ==TRUE){
  se =  read.csv(paste0(File,"/",assessment,"/se",assessment,".csv"))
}
if(CV.C == TRUE){
  cv.c = read.csv(paste0(File,"/",assessment,"/catchCV",assessment,".csv"))
}
indices2 = names(cpue)[-1]
wink.colors = data.frame(idx = indices2,
                         cols = c('#e6194b', "#3cb44b", "#ffe119",
                                  "#0082c8","#f58231", "#911eb4",
                                  "#46f0f0", "#f032e6", "#d2f53c",
                                  "#fabebe", "#008080","#e6beff", "#aa6e28")[1:length(indices2)])


#----------------------------------------------------
# Option use mean CPUE from state-space cpue averaging
#----------------------------------------------------
meanCPUE = FALSE

#----------------------------------------------------
# Starting value option for r and K
#----------------------------------------------------

#-----------------------------------------------
# Prior for unfished biomass K
#-----------------------------------------------
# The option are:
# a) Specify as a lognormal prior with mean and CV
# b) Specify as range to be converted into lognormal prior

# ><> new objective K prior
```

```
  K.dist <- c("lnorm","range")[1] # ><> to range
  # Get low and upper r quantile for 95% CI
  qrs <- qlnorm(c(0.025,0.975),log(0.46),sqrt(log(1+0.5^2)))
  #Apply CMSY Eq 3 by Froese et al, (2017)
  Klow <- 2*max(catch[,2])/qrs[2]
  Khigh <- 12*max(catch[,2])/qrs[1]
#  K.prior <- c(Klow,Khigh)
  K.prior <- c(4*172/0.46,0.5) #actual CV needed here

  #---------------------------------------------------------
  # mean and CV and sd for Initial depletion level P1= SB/SB0
  #---------------------------------------------------------
  # Set the initial depletion prior B1/K
  # To be converted into a lognormal prior (with upper bound at 1.1)

  psi.dist = c("lnorm","beta")[1]
  # specify as mean and CV
  #psi.prior.mean <- signif(as.numeric(3.6556/quantile(cpue[,2], na.rm = TRUE, #
  #                                          probs = 0.95)), digits = 2) ## uci over 95th
 # psi.prior = c(psi.prior.mean,0.5)
  psi.prior = c(0.45,0.5) #actual CV needed here
  #------------------------------------------------------------
  # Determine estimation for catchability q and observation error
  #------------------------------------------------------------
  # Assign q to CPUE
  sets.q = 1:(ncol(cpue)-1)

  #---------------------------------------------------
  # Determine r prior
  #---------------------------------------------------
  # The option are:
  # a) Specifying a lognormal prior
  # b) Specifying a resiliance category after Froese et al. (2017; CMSY)
  # Resilience: "Very low", "Low", "Medium", High" (requires r.range = TRUE)

  # use [1] lognormal(mean,stdev) or [2] range (min,max) or
  r.dist = c("lnorm","range")[1]
#  r.prior = c(0.2735,0.3)
  r.prior = c(0.46,sqrt(log(1+0.5^2))) #needs to be sd on log-scale
  #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
  # Observation Error
  #><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>

  #To Estimate additional observation variance set sigma.add = TRUE
  sigma.est = TRUE
```

```
# Series
sets.var = 1:(ncol(cpue)-1) # estimate individual additional variance

# As option for data-weighing
# minimum fixed observation error for each variance set (optional choose 1 value for both)
fixed.obsE = c(0.0) # Important if SE.I is not available

# Total observation error: TOE = sqrt(SE^2+sigma.est^2+fixed.obsE^2)

#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Process Error
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
#Estimate set sigma.proc == True
sigma.proc = TRUE
# Determines if process error deviation are estimated for all years (TRUE)
# or only from the point the first abundance index becomes available (FALSE)
proc.dev.all = TRUE
#------------------------------------------
if(sigma.proc == TRUE){
  igamma = c(0.2,0.1) #specify inv-gamma parameters  WAS 4,0.01

  # Process error check
  gamma.check = 1/rgamma(1000,igamma[1],igamma[2])
  # check mean process error + CV
   mu.proc = sqrt(mean(gamma.check)); CV.proc =
sd(sqrt(gamma.check))/mean(sqrt(gamma.check))

  # check CV
  # round(c(mu.proc,CV.proc),3)
  # quantile(sqrt(gamma.check),c(0.1,0.9))
}else{
  sigma.proc = 0.1 #IF Fixed: typicallly 0.05-0.15 (see Ono et al. 2012)
}
#--------------------------------------------

#><><><><><><><><><><><><><><><><><><>
#Include variation in catch estimates
#><><><><><><><><><><><><><><><><><><>
Catch.CV=TRUE

#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
# Optional: Do TAC Projections
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><>
Projection = FALSE # Switch on by Projection = TRUE
#Projection = TRUE # <---- USE THIS ONE FOR PROJECTIONS
```

```
  # Set range for alternative TAC projections
  TACs = seq(0,500,1)

  TACint = mean(catch[nrow(catch)-2,2]:catch[nrow(catch),2]) # avg last 3 years
  TACint_17 <- catch[nrow(catch),2]

  # Set number of projections years
  pyrs = 8

  #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>
  # Execute model and produce output
  #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>

  # MCMC settings
  ni <- 300000 # Number of iterations
  nt <- 5 # Steps saved
  #nt <- 50 #steps saved <----USE THIS ONE FOR PROJECTIONS
  nb <- 60000 # Burn-in
  nc <- 2 # number of chains
  nsaved = (ni-nb)/nt*nc

  source(paste0(JABBA.file,"/JABBA",version,".R"))

}# THE END
```

1.2.2. Source code for JABBA model and projections

```
cat(paste0("\n","><>><>><>><>><>><>><>><>"))
cat(paste0("\n","><> Run Model ",Mod.names,"<><"))
cat(paste0("\n","><>><>><>><>><>><>><>><>","\n","\n"))
# setwd(paste(File))
dir.create(paste0(File,"/",assessment),showWarnings = FALSE)
dir.create(paste0(File,"/",assessment,"/",Scenario,"_",Mod.names),showWarnings = FALSE)
dir.create(paste0(File,"/",assessment,"/",Scenario,"_",Mod.names,"/Input"),showWarnings =
FALSE)
input.dir = paste0(File,"/",assessment,"/",Scenario,"_",Mod.names,"/Input")

#><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><
>><>
# Define objects to make sure they exist if not included in Prime file
#><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><
>><>
if(exists("igamma")==FALSE) igamma = c(0.2,0.1)  # Generic process error prior  WAS 4,0.01
if(exists("BmsyK")==FALSE) BmsyK = 0.4  # JABBA default for Pella model
```

```
if(exists("Model")==FALSE){ model = 1; Mod.names = c("Schaefer")} # Run Schaefer if model
is not specified
if(exists("proc.dev.all")==FALSE) proc.dev.all = FALSE # process error deviation if only catch
is available
if(exists("Plim")==FALSE) Plim = 0  # Standard non-compound model
if(exists("P_bound")==FALSE) P_bound = c(0.02,1)  # Soft penalty bounds for P
if(exists("KOBE.plot")==FALSE) KOBE.plot = TRUE # Produces JABBA Kobe plot
if(exists("KOBE.type")==FALSE) KOBE.type = c("ICCAT","IOTC")[2] # ICCAT uses 3
colors; IOTC 4 (incl. orange)
if(exists("SP.plot")==FALSE) SP.plot = c("standard","phase")[2] # Produces standard or 'Kobe
phase' SP plot
if(exists("Biplot")==FALSE) Biplot= TRUE # Produces a "post-modern" biplot with buffer and
target zones (Quinn & Collie 2005)
if(exists("save.trajectories")==FALSE) save.trajectories =FALSE # saves posteriors of P=B/K,
B/Bmsy and H/Hmsy as .RData object
if(exists("harvest.label")==FALSE) harvest.label = c("Hmsy","Fmsy")[2] # choose label
preference H/Hmsy versus Fmsy
if(exists("CPUE.plot")==FALSE) CPUE.plot= TRUE # Runs state-tool to produce "aligned"
multi-CPUE plot
if(exists("catch.metric")==FALSE) catch.metric = "(t)" # Runs state-tool to produce "aligned"
multi-CPUE plot
if(exists("meanCPUE")==FALSE) meanCPUE = FALSE # Uses averaged CPUE from state-
space tool instead of individual indices
if(exists("Projection")==FALSE) Projection = FALSE # Use Projections: requires to define
TACs vectors
if(exists("save.projections")==FALSE) save.projections = FALSE# saves projection posteriors
as .RData object
if(exists("Reproduce.seed")==FALSE) Reproduce.seed = FALSE # If FALSE a random seed
assigned to each run (default)
if(exists("TACint")==FALSE) TACint = mean(catch[nrow(catch)-3,2]:catch[nrow(catch),2]) #
use mean catch from last years
if(exists("imp.yr")==FALSE) imp.yr = as.numeric(format(Sys.Date(), "%Y"))+1 # use next year
from now
if(exists("init.values")==FALSE) init.values =FALSE # Allows to add manual starting values for
K, r, q
if(exists("sigmaobs_bound")==FALSE) sigmaobs_bound = 1 # Adds an upper bound to the
observation variance
if(exists("sigmaproc_bound")==FALSE) sigmaproc_bound = 0.2 # Adds an upper bound to the
process variance
if(exists("q_bounds")==FALSE) q_bounds= c(10^-10,10) # Defines lower and upper bounds for
q
if(exists("K_bounds")==FALSE) K_bounds= c(10,10^8) # Defines lower and upper bounds for q
# Save entire posterior as .RData object
if(exists("save.all")==FALSE) save.all = FALSE #
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><<
>
```

```
## Prepare input data ----

cat(paste0("\n","><> Prepare input data <><","\n"))
indices = names(cpue)[2:ncol(cpue)]
n.indices = max(length(indices),1)
catches = names(catch)[2:ncol(catch)]
n.catches = length(catches)

years=catch[,1]
styr = min(years)
endyr = max(years)
n.years = length(years)
styr.cpue = min(cpue[,1])
styr.I = styr.cpue-styr+1


# Convert input data to matrices for JAGS input
conv.cpue = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.indices),styr.I-
1,n.indices),as.matrix(cpue[,-1])))
CPUE=matrix(conv.cpue,nrow=n.years,ncol=n.indices)


if(SE.I==FALSE){
  se = cpue
  conv.se = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.indices),styr.I-
1,n.indices),as.matrix(cpue[,-1])))
  se2 = matrix(ifelse(fixed.obsE>0,fixed.obsE^2,10^-10),n.years,n.indices)#/2
} else{
  conv.se = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.indices),styr.I-
1,n.indices),as.matrix(se[,-1])))
  #conv.se = sqrt(conv.se^2+fixed.obsE^2)
  se2 = matrix(ifelse(is.na(conv.se),0.3^2,conv.se)^2,n.years,n.indices)+fixed.obsE^2#/2
}

if(Catch.CV==FALSE){
  conv.catch = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.catches),styr.I-
1,n.catches),as.matrix(catch[,-1])))
  Catch=matrix(conv.catch,nrow=n.years,ncol=n.catches)
  Catch[is.na(Catch)] = 0 # Replace any NA by zero
} else{
  conv.catch = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.catches),styr.I-
1,n.catches),as.matrix(catch[,-1])))
  Catch=matrix(conv.catch,nrow=n.years,ncol=n.catches)
  Catch[is.na(Catch)] = 0 # Replace any NA by zero
```

```
  conv.catch.cv=as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.catches),styr.I-
1,n.catches),as.matrix(cv.c[,-1]))) #assumes we have a cv for every catch
  catch.cv=matrix(conv.catch.cv,nrow=n.years,ncol=n.catches)
  catch.cv[is.na(catch.cv)] = 0 # Replace any NA by zero
}


# Total Catch
TC = apply(Catch,1,sum)

# Plot Catch

cat(paste0("\n","><> Plot Catch in Input subfolder <><","\n"))

Par = list(mfrow=c(1,1),mar = c(5, 5, 1, 1), mgp =c(3,1,0), tck = -0.02,cex=0.8)
png(file = paste0(input.dir,"/Catches_",assessment,".png"), width = 7, height = 5,
    res = 200, units = "in")
par(Par)
plot(catch[,1],catch[,1],ylim=c(0,max(catch[,2:ncol(catch)],na.rm=TRUE)),ylab=paste0("Catch
",catch.metric),xlab="Year",type="n")
for(i in 2:ncol(catch)) lines(catch[,1],catch[,i],lty=(i-1),lwd=2)
legend("topright",paste(names(catch)[2:ncol(catch)]),lty=1:(ncol(catch)-1),bty="n")
dev.off()

#---------------------
# Index color palette
#---------------------
jabba.colors = as.character(rep(c('#e6194b', "#3cb44b", "#ffe119",
                                  "#0082c8","#f58231", "#911eb4",
                                  "#46f0f0", "#f032e6", "#d2f53c",
                                  "#fabebe", "#008080","#e6beff", "#aa6e28"),2))
#--------------------
# Set seed
#--------------------
if(Reproduce.seed==FALSE){ set.seed(ceiling(runif(1,min=0,max=1e6))) } else {set.seed(123)}

#------------------------------------------------------------------------
# CPUE run State-Space model for averaging CPUE
#------------------------------------------------------------------------
if(CPUE.plot==TRUE){
  cat(paste0("\n","><> Run State-Space CPUE averaging tool","\n"))
  #find first time-series with first CPUE
  q1.y = c(1:n.years)[is.na(apply(CPUE,1,mean,na.rm=TRUE))==FALSE][1] #first year with
CPUE
  q1.I = which.max(CPUE[q1.y,])
```

```
qs = c(q1.I,c(1:(ncol(cpue)-1))[-q1.I])


sink("cpueAVG.jags")
cat("
   model {

   # Prior specifications
   eps <- 0.0000000000001 # small constant

   iq[1] ~ dgamma(1000,1000)
   q[1] <-  pow(iq[1],-1)
   logq[1] <- log(1)
   for(i in 2:nI){
   iq[i] ~ dgamma(0.001,0.001)
   q[i] <- pow(iq[i],-1)
   logq[i] <-  log(q[i])
   }


   ")

if(sigma.proc==TRUE){
  cat("
     # Process variance
     isigma2 <- isigma2.est
     sigma2 <- pow(isigma2,-1)
     sigma <- sqrt(sigma2)
     fakesigma.fixed <- sigma.fixed # Prevent unused variable error msg
     ",append=TRUE)
}else{ cat("
   isigma2 <- pow(sigma.fixed+eps,-2)
        sigma2 <- pow(isigma2,-1)
        sigma <- sqrt(sigma2)

        ",append=TRUE)}

if(sigma.est==TRUE){
  cat("
     # Observation variance
     # Observation error
     itau2~ dgamma(0.001,0.001)
     tau2 <- 1/itau2


     for(i in 1:nI)
```

```
    {
    for(t in 1:N)
    {
    var.obs[t,i] <- SE2[t,i]+tau2
    ivar.obs[t,i] <- 1/var.obs[t,i]
    # note total observation error (TOE)
    TOE[t,i] <- sqrt(var.obs[t,i])

    }}
    ",append=TRUE)
}else{ cat("
   # Observation variance
        # Observation error
        itau2~ dgamma(2,2)
        tau2 <- 1/itau2


        for(i in 1:nI)
        {
        for(t in 1:N)
        {
        var.obs[t,i] <- SE2[t,i] # drop tau2
        fake.tau[t,i] <- tau2

        ivar.obs[t,i] <- 1/var.obs[t,i]
        # note total observation error (TOE)
        TOE[t,i] <- sqrt(var.obs[t,i])

        }}

        ",append=TRUE)}

# Run rest of code
cat("
   # Process variance prior
   isigma2.est ~ dgamma(0.001,0.001)


   # Priors and constraints
   logY.est[1] ~ dnorm(logY1, 1)      # Prior for initial population size

   mean.r ~ dnorm(1, 0.001)           # Prior for mean growth rate

   # Likelihood
   # State process
   for (t in 1:(N-1)){
```

```
    r[t] ~ dnorm(mean.r, isigma2)
    logY.est[t+1] <- logY.est[t] + r[t] }

    # Observation process
    for (t in 1:N) {
    for(i in 1:nI){
    y[t,i] ~ dnorm(logY.est[t]+logq[i], ivar.obs[t,i])
    }}

    # Population sizes on real scale
    for (t in 1:N) {
    Y.est[t] <- exp(logY.est[t])
    }

  }
    ",fill = TRUE)
  sink()



  q.init = 1
  mCPUE = as.matrix(CPUE[q1.y:n.years,qs])
  mSE2 = as.matrix(se2[q1.y:n.years,qs])
  if(n.indices>1) for(i in 2:n.indices){q.init[i] =
mean(mCPUE[,i],na.rm=TRUE)/mean(mCPUE[,1],na.rm=TRUE)}
  # Bundle data
  jags.data <- list(y = log(mCPUE),SE2=mSE2, logY1 = log(mCPUE[1,1]), N =
length(q1.y:n.years),nI=n.indices,sigma.fixed=ifelse(sigma.proc==TRUE,0,sigma.proc))

  # Initial values
  inits <- function(){list(isigma2.est=runif(1,20,100), itau2=runif(1,80,200), mean.r = rnorm(1),iq
= 1/q.init)}

  # Parameters monitored
  parameters <- c("mean.r", "sigma","r", "Y.est","q")


  # Call JAGS from R (BRT 3 min)
  mod.cpue <- jags(jags.data, inits, parameters, "cpueAVG.jags", n.chains = nc, n.thin =
max(nt,2), n.iter = max(ni/5,10000), n.burnin = nb/10)


  cat(paste0("\n","><> Plot State-Space CPUE fits in Input subfolder <><","\n"))
  # get individual trends
  fitted <- lower <- upper <- NULL
  cpue.yrs = years[q1.y:n.years]
```

```
  for (t in 1:nrow(mCPUE)){
    fitted[t] <- median(mod.cpue$BUGSoutput$sims.list$Y.est[,t])
    lower[t] <- quantile(mod.cpue$BUGSoutput$sims.list$Y.est[,t], 0.025)
    upper[t] <- quantile(mod.cpue$BUGSoutput$sims.list$Y.est[,t], 0.975)}


  q.adj = apply(mod.cpue$BUGSoutput$sims.list$q,2,median)


  Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
  png(file = paste0(input.dir,"/CPUE_",assessment,"_",Scenario,".png"), width = 5, height = 3.5,
      res = 200, units = "in")
  par(Par)
  u.ylim = NULL
  for(i in 1:n.indices){ u.ylim = c(u.ylim,exp(log(mCPUE[,i]/q.adj[i])+1.96*sqrt(mSE2[,i])))}
  ylim = c(0,max(u.ylim,na.rm=TRUE))
  plot(0, 0, ylim = ylim, xlim = range(cpue.yrs), ylab = "Expected CPUE", xlab = "Year", col =
"black", type = "n")
  legend("topright",paste(indices),lwd=2,col=(jabba.colors)[1:n.indices],bty="n")
  polygon(x = c(cpue.yrs,rev(cpue.yrs)), y = c(lower,rev(upper)), col = "gray", border =
"gray90")

  for(i in 1:n.indices)
  {
    shift = runif(1,-0.1,0.1)
    cols=jabba.colors[qs[i]]

plotCI(cpue.yrs+shift,mCPUE[,i]/q.adj[i],ui=exp(log(mCPUE[,i]/q.adj[i])+1.96*sqrt(mSE2[,i])),
li=exp(log(mCPUE[,i]/q.adj[i])-1.96*sqrt(mSE2[,i])),add=TRUE,col= cols,pt.bg =
cols,pch=21,gap=0)
    lines(cpue.yrs+shift,mCPUE[,i]/q.adj[i], col = cols,lwd=2)
    points(cpue.yrs+shift,mCPUE[,i]/q.adj[i], bg = cols,pch=21)
  }
  lines(cpue.yrs,fitted,lwd=2)

  dev.off()

  logSE = apply(log(mod.cpue$BUGSoutput$sims.list$Y.est),2,sd)


  if(nrow(mCPUE)<n.years) {
    fitted = c(rep(NA,q1.y-1),fitted)
    logSE = c(rep(0.2,q1.y-1),logSE)
  }
  avgCPUE = data.frame(Year=years,CPUE= fitted,logSE=logSE)
```

```
  write.csv(avgCPUE,paste0(input.dir,"/avgCPUE_",assessment,"_",Scenario,".csv"))

  if(meanCPUE==TRUE){
    cat(paste0("\n","><> Use average CPUE as input for JABBA <><","\n"))

    CPUE = as.matrix(avgCPUE[,2])
    cpue.check = cpue[,-1]
    cpue.check[is.na(cpue[,-1])]=0
    CPUE[,1] = ifelse(apply(cpue.check,1,sum)==0,rep(NA,length(CPUE[,1])),CPUE[,1])
    se2 =  as.matrix(avgCPUE[,3]^2)
    n.indices=1
    indices = "All"
    sets.q =1
    sets.var =1
  }

  }

#------------------------------------------------------------------------
# END of CPUE State-Space tool
#------------------------------------------------------------------------

#-----------
# FUNCTIONS
#-----------
cat(paste0("\n","><> Prepare JABBA prior inputs <><","\n"))

#-------------------------------------------------
# Function to get beta prior parameters
#-------------------------------------------------
get_beta <- function(mu,CV,Min=0,Prior="x"){
  a = seq(0.0001,1000,0.001)
  b= (a-mu*a)/mu
  s2 = a*b/((a+b)^2*(a+b+1))
  sdev = sqrt(s2)
  # find beta parameter a
  CV.check = (sdev/mu-CV)^2
  a = a[CV.check==min(CV.check)]
  #find beta parameter b
  b = (a-mu*a)/mu
  x = seq(Min,1,0.001)
  pdf = dbeta(x,a,b)
  plot(x,pdf,type="l",xlim=range(x[pdf>0.01]),xlab=paste(Prior),ylab="",yaxt="n")
  polygon(c(x,rev(x)),c(rep(0,length(x)),rev(ifelse(pdf==Inf,100000,pdf))),col="grey")
  return(c(a,b))
```

```
}

#------------------------------------------------
# Function to get gamma prior parameters
#------------------------------------------------

get_gamma <- function(mu,CV,Prior="x"){
  a = seq(0.0001,1000,0.0001)
  b = a/mu
  s2 = (a/b^2)
  sdev = sqrt(s2)
  # find beta parameter a
  CV.check = (sdev/mu-CV)^2
  a = a[CV.check==min(CV.check)]
  #find beta parameter b
  b = a/mu
  x = sort(rgamma(1000,a,b))
  pdf = dgamma(x,a,b)
  plot(x,pdf,type="l",xlim=range(x[pdf>0.01]),xlab=paste(Prior),ylab="",yaxt="n")
  polygon(c(x,rev(x)),c(rep(0,length(x)),rev(ifelse(pdf==Inf,100000,pdf))),col="grey")
  return(c(a,b))
}


#------------------------------------------------
# Function to get lognormal prior parameters
#------------------------------------------------
plot_lnorm <- function(mu,CV,Prior="x"){
  sdev= sqrt(log(CV^2+1))
  rand.pr = rlnorm(1000,log(mu),sdev)
  x = seq(min(rand.pr),quantile(rand.pr,0.995),max(rand.pr/500))
  pdf = dlnorm(x,log(mu),sdev)
  plot(x,pdf,type="l",xlim=range(x),xlab=paste(Prior),ylab="",yaxt="n")
  polygon(c(x,rev(x)),c(rep(0,length(x)),rev(ifelse(pdf==Inf,100000,pdf))),col="grey")
  return(c(mu,sdev))
}

#-----------------------------------
# Function kobeJabba for FLR
#-----------------------------------
kobeJabba<-function(x,minyear=1){

  out=cbind(melt(x[,,2]),c(x[,,3]))
  names(out)=c("iter","year","stock","harvest")
  out$year=out$year+minyear-1
  out}
```

```
#-------------------------------------------------
# Function kobeJabbaProj for projections with FLR
#-------------------------------------------------
kobeJabbaProj<-function(x,minyear=1,tac=NULL){

  out=cbind(melt(x[,,,2]),c(x[,,,3]),c(x[,,,4]))
  names(out)=c("iter","year","tac","stock","harvest","Hrate")
  out$year=out$year+minyear-1

  out}

#-------------------------------------------------
# Determine initial ranges for r
#-------------------------------------------------
if(r.dist=="range"){
  # initial range of r based on resilience (FishBase.org)
  if(length(r.prior)>1){ start.r = r.prior} else
    if(r.prior == "High") {
      start.r <- c(0.6,1.5)} else if(r.prior == "Medium") {
        start.r <- c(0.2,0.8)}    else if(r.prior == "Low") {
          start.r <- c(0.05,0.5)}  else { # i.e. res== "Very low"
            start.r <- c(0.015,0.1)}

  log.r = mean(log(start.r))
  sd.r = abs(log.r - log(start.r[1]))/2
  r.prior = c(exp(log.r),sd.r)
  CV.r = sqrt(exp(sd.r^2)-1)
} else {
  log.r = log(r.prior[1])
  sd.r = r.prior[2]
  CV.r = sqrt(exp(sd.r^2)-1)
}

#-------------------------------------------------
# Prepare K prior
#-------------------------------------------------
if(K.dist=="range"){
  log.K = mean(log(K.prior))
  sd.K= abs(log.K - log(K.prior[1]))/2
  CV.K = sqrt(exp(sd.K^2)-1)
} else {

  log.K = log(K.prior[1])
  CV.K = K.prior[2]
  sd.K=sqrt(log(CV.K^2+1))
```

```
}


#-------------------------------------------------------
# Get JABBA parameterization and surplus production function
#-------------------------------------------------------
# For Pella-Tomlinson
if (Model == 3 | Model == 4) {
  ## run thru sensitivities if given shape
  # find inflection point
  ishape = NULL
  # Find shape for  SBmsytoK
  ishape = seq(0.1, 10, 0.001)
  check.shape = ((ishape) ^ (-1 / (ishape - 1)) - BmsyK) ^ 2
  #  Set shape (> 0, with 1.001 ~ Fox and 2 = Schaefer)
  shape =  ishape[check.shape == min(check.shape)]

  if (exists("sensname")) {
    if (sensname == 'M') {
      m = shape = sensmean[s] ## becomes m.mu
      shape.CV = sensvar[s]
      cat(paste0(shape, " ", shape.CV), "\n")
    } ## end sensname == M
  } ## end sensname exists
} else {shape = FALSE}


#-----------------------------------------------


# Set shape m for Fox and Schaefer: Fox m ~1; Schaefer m =2
if(shape==FALSE){
  if(Model == 1){m=2} else {m = 1.001}}else{m=shape}

cat(paste0("\n","><> Plot Prior distributions in Input subfolder  <><","\n"))

Par = list(mfrow=c(1,3),mai=c(0.5,0.1,0,.1),omi = c(0.1,0.2,0.1,0) + 0.1,mgp=c(2,1,0), tck = -
0.02,cex=0.8)
png(file = paste0(input.dir,"/Priors_",assessment,"_",Scenario,".png"), width = 9, height = 3,
    res = 200, units = "in")
par(Par)
K.pr = plot_lnorm(exp(log.K),CV.K,Prior="K")

if(psi.dist=="beta"){
  psi.pr = get_beta(mu=psi.prior[1],CV=psi.prior[2],Min=0,Prior=paste0("Prior
B(",years[1],")/K"))} else {
```

```
  psi.pr = plot_lnorm(mu=psi.prior[1],CV=psi.prior[2],Prior=paste0("Prior B(",years[1],")/K"))
 }


r.pr = plot_lnorm(mu=exp(log.r),CV=CV.r,Prior="r")
mtext(paste("Density"), side=2, outer=TRUE, at=0.5,line=1,cex=0.9)
dev.off()


cat(paste0("\n","><> Plot assumed Surplus Production shape in Input subfolder  <><","\n"))


# Plot MSY
Par = list(mfrow=c(1,1),mai=c(0.6,0.3,0,.15),omi = c(0.1,0.2,0.2,0) + 0.1,mgp=c(2,1,0), tck = -
0.02,cex=0.8)
png(file = paste0(input.dir,"/Production",assessment,"_",Scenario,".png"), width = 6, height = 5,
   res = 200, units = "in")
par(Par)

# Get Bmsy/B0 as a function of M
Bmsy=(m)^(-1/(m-1))
P = seq(0.0001,1,0.001)
SP = ifelse(P>Plim,r.pr[1]/(m-1)*P*(1-P^(m-1)),r.pr[1]/(m-1)*P*(1-P^(m-1))*4*P)
#if(is.null(refBmsy)==TRUE) refBmsy = Bmsy
plot(P,SP/max(SP),type="l",ylab="Relative Yield",xlab="B/B0",lwd=2)
mtext(paste("Relative Yield"), side=2, outer=TRUE, at=0.6,line=1,cex=0.9)
legend("topright",c("SPM"),col=c(1),lwd=2,bty="n")


if(Model==4){
 # shape density
 #dm = dgamma(seq(0.001,5,0.1),5,5)*m
 dm = dlnorm((seq(0.001,5,0.1)),log(m),shape.CV)
 dm = dm/max(dm)
 bmsyk  = (seq(0.001,5,0.1))^(-1/(seq(0.001,5,0.1)-1))

 polygon(c(bmsyk,rev(bmsyk)),c(dm,rep(0,length(dm))),col="grey",border=0)
}
abline(v=Bmsy,lty=2)
mtext(paste("Relative Yield"), side=2, outer=TRUE, at=0.6,line=1,cex=0.9)
legend("topright",c("SPM"),col=c(1),lwd=2,bty="n")
abline(v=Bmsy,lty=2)
dev.off()


# Note PRIORS and save input subfolder
```

```
Priors =rbind(K.pr,psi.prior,c(r.pr[1],CV.r))
row.names(Priors) = c("K","Psi","r")
colnames(Priors) = c("Mean","CV")
write.csv(Priors,paste0(input.dir,"/Priors",assessment,"_",Scenario,".csv"))


#------------------------------------------------------------
# Set up JABBA
#------------------------------------------------------------
cat(paste0("\n","><> Set up JAGS input <><","\n"))
# Plot MSY
# remove scientific numbers
options(scipen=999)
#------------------------------------------------------------


# starting values
nq = length(unique(sets.q))
nvar = length(unique(sets.var))

## MDFitchett change to TAC setup
#------------------------------------------------------------
# Setup TAC projection
#------------------------------------------------------------
if(Projection==TRUE) {
  nTAC = length(TACs)
  TAC = mat.or.vec(pyrs, nTAC)
  yr.now = 2019
  yr.last = max(years) # assessment year

  for (i in 1:nTAC) {
    TAC[, i] = c(rep(TACint, yr.now - yr.last), rep(TACs[i], pyrs - (yr.now -yr.last)))
  }

} else if(Projection == FALSE){
  nTAC = 1
  TAC = TC[n.years]
  pyrs = 1
}

#------------------------------------------------------------
# JABBA Schaefer/Fox Models 1-2, Pella 3
#------------------------------------------------------------

# Slope of hockey-stick
slope.HS = ifelse(Plim==0,1/10^-10,1/Plim)
```

```
nSel = 1 # setup for JABBA-SELECT version (in prep)
nI = ncol(CPUE) # number of CPUE series
stI = ifelse(proc.dev.all==TRUE,1,
c(1:n.years)[is.na(apply(CPUE,1,mean,na.rm=TRUE))==FALSE][1]) #first year with CPUE


# Initial starting values
if(Catch.CV==FALSE){
inits <- function(){list(K=
rlnorm(1,log(4*172/0.46),0.5),r=rlnorm(1,log(0.46),0.5),psi=rlnorm(1,log(0.45),0.5), q =
runif(1,0.0000000001,10), isigma2.est=rgamma(1,0.2,0.1), itau2=rgamma(1,0.2,0.1))}
} else {
#inits <- function(){list(K=4*172/0.46,r=0.46,rCatch = TC, psi=0.45, q = 5, isigma2.est=100,
itau2=100)}
inits <- function(){list(K= rlnorm(1,log(4*172/0.46),0.5),r=rlnorm(1,log(0.46),0.5),rCatch = TC,
psi=rlnorm(1,log(0.45),0.5), q = runif(1,0.0000000001,10), isigma2.est=rgamma(1,0.2,0.1),
itau2=rgamma(1,0.2,0.1))}
}

# starting value option
if(init.values==TRUE){
inits <- function(){list(K= K.init,r=r.init,q = q.init, isigma2.est=runif(1,20,100),
itau2=runif(nvar,80,200))}
}

# JABBA input data
if(Catch.CV==FALSE){
surplus.dat = list(N=n.years, TC = TC, I=CPUE,SE2=se2,mu.m=m,r.pr=r.pr,psi.pr=psi.pr,K.pr =
K.pr,
            nq=nq,nI = nI,nvar=nvar,sigma.fixed=ifelse(sigma.proc==TRUE,0,sigma.proc),
            sets.var=sets.var, sets.q=sets.q,pen.bk =
rep(0,n.years),Plim=Plim,slope.HS=slope.HS,
            nTAC=nTAC,pyrs=pyrs,TAC=TAC,igamma =
igamma,stI=stI,TACint=TACint,TACint_17=TACint_17,P_bound=P_bound,proc.pen=0,K.pen
= 0,
            obs.pen =
rep(0,nvar),q_bounds=q_bounds,sigmaobs_bound=sigmaobs_bound,sigmaproc_bound=sigmapr
oc_bound,K_bounds=K_bounds)
} else {
surplus.dat = list(N=n.years, TC = TC, catch.cv = catch.cv,
I=CPUE,SE2=se2,mu.m=m,r.pr=r.pr,psi.pr=psi.pr,K.pr = K.pr,
            nq=nq,nI = nI,nvar=nvar,sigma.fixed=ifelse(sigma.proc==TRUE,0,sigma.proc),
            sets.var=sets.var, sets.q=sets.q,pen.bk =
rep(0,n.years),Plim=Plim,slope.HS=slope.HS,
```

```
            nTAC=nTAC,pyrs=pyrs,TAC=TAC,igamma =
igamma,stI=stI,TACint=TACint,TACint_17=TACint_17,P_bound=P_bound,proc.pen=0,K.pen
= 0,
            obs.pen =
rep(0,nvar),q_bounds=q_bounds,sigmaobs_bound=sigmaobs_bound,sigmaproc_bound=sigmapr
oc_bound,K_bounds=K_bounds)
}




# If shape parameter is estimated (Model =4)
if(Model==4){
  surplus.dat$m.CV = shape.CV }


# JAGS model file
JABBA = "JABBA.jags"


# PARAMETERS TO MONITOR -
if(Catch.CV==FALSE){
params <- c("K","r", "q", "psi","sigma2", "tau2","m","Hmsy","SBmsy", "MSY",
"BtoBmsy","HtoHmsy","Overfishing_ind","CPUE","Proc.Dev","P","SB","H","prP","prBtoBms
y","prHtoHmsy", "prOverfishing_ind", "prH","TOE")
} else {
params <- c("K","r", "q", "psi","sigma2", "tau2","m","Hmsy","SBmsy", "MSY",
"BtoBmsy","HtoHmsy","Overfishing_ind","CPUE","Proc.Dev","P","SB","H","prP","prBtoBms
y","prHtoHmsy", "prOverfishing_ind", "prH","TOE")
}



cat(paste0("\n","><> RUN ",Mod.names," model for ",assessment," ",Scenario," in JAGS
<><","\n","\n"))

# JAGS MODEL Standard
sink("JABBA.jags")
cat("

   model {

   # Prior specifications
   eps <- 0.00000000000000000000000000000001 # small constant

   #Catchability coefficients
```

```
    for(i in 1:nq)
    {
    q[i] ~ dunif(q_bounds[1],q_bounds[2])
    }


    ")

if(Model==4){
  cat("
      # Shape m prior
      m ~ dlnorm(log(mu.m),pow(m.CV,-2))
      ",append=TRUE)
  }else{ cat("
      m <- mu.m
    ",append=TRUE)}

if(psi.dist =="beta"){
  cat("
      # Beta Prior for Biomass depletion at the start (deterministic)
      psi ~ dbeta(psi.pr[1],psi.pr[2])
      ",append=TRUE)
} else {
  cat("
      # Lognormal for Biomass depletion at the start (deterministic)
      psi ~ dlnorm(log(psi.pr[1]),pow(psi.pr[2],-2)) #I(0.1,1.1)
      ",append=TRUE)
}

if(sigma.proc==TRUE){
  cat("
      # Process variance
      isigma2 <- isigma2.est
      sigma2 <- pow(isigma2,-1)
      sigma <- sqrt(sigma2)
      fakesigma.fixed <- sigma.fixed # Prevent unused variable error msg
      ",append=TRUE)
}else{ cat("
      isigma2 <- pow(sigma.fixed+eps,-2)
          sigma2 <- pow(isigma2,-1)
          sigma <- sqrt(sigma2)

          ",append=TRUE)}

if(sigma.est==TRUE){
  cat("
```

```
    # Observation variance
    for(i in 1:nvar)
    {
    # Observation error
    itau2[i]~ dgamma(0.2,0.1)   #from D7, was 0.001,0.001
    tau2[i] <- 1/itau2[i]
    }

    for(i in 1:nI)
    {
    for(t in 1:N)
    {
    var.obs[t,i] <- SE2[t,i]+tau2[sets.var[i]]
    ivar.obs[t,i] <- 1/var.obs[t,i]
    # note total observation error (TOE)
    TOE[t,i] <- sqrt(var.obs[t,i]) # Total observation variance

    }}
    ",append=TRUE)
}else{ cat("
    # Observation variance
        for(i in 1:nvar)
        {
        # Observation error
        itau2[i]~ dgamma(0.2,0.1)
        tau2[i] <- 1/itau2[i]
        }

        for(i in 1:nI)
        {
        for(t in 1:N)
        {
        var.obs[t,i] <- SE2[t,i] # drop tau2
        fake.tau[t,i] <- tau2[sets.var[i]]

        ivar.obs[t,i] <- 1/var.obs[t,i]
        # note total observation error (TOE)
        TOE[t,i] <- sqrt(var.obs[t,i])

        }}

        ",append=TRUE)}

# Process section divided into 2 chunks, replicated for Catch.CV = FALSE/TRUE
if(Catch.CV==FALSE){
cat("
```

```
   # Process variance prior
   isigma2.est ~ dgamma(igamma[1],igamma[2])

   # Carrying Capacity SB0
   K ~ dlnorm(log(K.pr[1]),pow(K.pr[2], -2))


   # informative priors for Hmsy as a function of r
   r ~ dlnorm(log(r.pr[1]),pow(r.pr[2],-2))


   #Process equation
   Pmean[1] <- log(psi)
   iPV[1] <- ifelse(1<(stI),10000,isigma2) # inverse process variance
   P[1] ~ dlnorm(Pmean[1],iPV[1]) # set to small noise instead of isigma2
   penB[1]  <- ifelse(P[1]<P_bound[1],log(K*P[1])-
log(K*P_bound[1]),ifelse(P[1]>P_bound[2],log(K*P[1])-log(K*P_bound[2]),0)) # penalty if
Pmean is outside viable biomass

   # Process equation
   for (t in 2:N)
   {
   Pmean[t] <- ifelse(P[t-1] > Plim,
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1)) - TC[t-1]/K,0.005)),
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1))*P[t-1]*slope.HS - TC[t-1]/K,0.005)))
   iPV[t] <- ifelse(t<(stI),10000,isigma2) # inverse process variance
   P[t] ~ dlnorm(Pmean[t],iPV[t])
   penB[t]  <- ifelse(P[t]<(P_bound[1]),log(K*P[t])-
log(K*(P_bound[1])),ifelse(P[t]>P_bound[2],log(K*P[t])-log(K*(P_bound[2])),0)) # penalty if
Pmean is outside viable biomass
   }

   # Process error deviation
   for(t in 1:N){
   Proc.Dev[t] <- P[t]-exp(Pmean[t])}

   # Enforce soft penalties on bounds for P
   for(t in 1:N){
   pen.bk[t] ~ dnorm(penB[t],1000) # enforce penalty with CV = 0.1
   }
   ", append=TRUE)
} else{ cat("
   # Process variance prior
   isigma2.est ~ dgamma(igamma[1],igamma[2])

   # Carrying Capacity SB0
```

```
   K ~ dlnorm(log(K.pr[1]),pow(K.pr[2], -2))

   # informative priors for Hmsy as a function of r
   r ~ dlnorm(log(r.pr[1]),pow(r.pr[2],-2))


#Process equation
    Pmean[1] <- log(psi)
    iPV[1] <- ifelse(1<(stI),10000,isigma2) # inverse process variance
    P[1] ~ dlnorm(Pmean[1],iPV[1]) # set to small noise instead of isigma2
    penB[1]  <- ifelse(P[1]<P_bound[1],log(K*P[1])-
log(K*P_bound[1]),ifelse(P[1]>P_bound[2],log(K*P[1])-log(K*P_bound[2]),0)) # penalty if
Pmean is outside viable biomass
    Catch_precision[1]<-1/log(1+catch.cv[1,]*catch.cv[1,])
    rCatch[1] ~ dlnorm(log(TC[1]),Catch_precision[1])

# Process equation
   for (t in 2:N)
   {
   Pmean[t] <- ifelse(P[t-1] > Plim,
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1)) - rCatch[t-1]/K,0.005)),
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1))*P[t-1]*slope.HS - rCatch[t-
1]/K,0.005)))

    iPV[t] <- ifelse(t<(stI),10000,isigma2) # inverse process variance
    P[t] ~ dlnorm(Pmean[t],iPV[t])
    Catch_precision[t]<-1/log(1+catch.cv[t,]*catch.cv[t,])
    rCatch[t] ~ dlnorm(log(TC[t]),Catch_precision[t])
    penB[t]  <- ifelse(P[t]<(P_bound[1]),log(K*P[t])-
log(K*(P_bound[1])),ifelse(P[t]>P_bound[2],log(K*P[t])-log(K*(P_bound[2])),0)) # penalty if
Pmean is outside viable biomass
    }

  # Process error deviation
   for(t in 1:N){
   Proc.Dev[t] <- P[t]-exp(Pmean[t])}

   # Enforce soft penalties on bounds for P
  for(t in 1:N){
  pen.bk[t] ~ dnorm(penB[t],1000) # enforce penalty with CV = 0.1
   }
          ", append=TRUE)

}
```

```
# Run rest of code
   cat("
   Hmsy <- r*pow(m-1,-1)*(1-1/m)

   for (t in 1:N)
   {
   SB[t] <- K*P[t]
   }"
      , append=TRUE)

   #Include calculation of H for random catch
   if(Catch.CV==FALSE){
     cat("
   for (t in 1:N)
   {
   H[t] <- TC[t]/SB[t]
   }
    ", append=TRUE)} else{
     cat("
   for (t in 1:N)
   {
   H[t] <- rCatch[t]/SB[t]
   }
    ", append=TRUE)}

   # Observation equation in related to EB
   cat("
   for(i in 1:nI)
   {
   for (t in 1:N)
   {
   Imean[t,i] <- log(q[sets.q[i]]*P[t]*K);
   I[t,i] ~ dlnorm(Imean[t,i],(ivar.obs[t,i]));
   CPUE[t,i] <- q[sets.q[i]]*P[t]*K
   }}


   #Management quantities
   SBmsy_K <- (m)^(-1/(m-1))
   SBmsy <- SBmsy_K*K

   MSY <- SBmsy*Hmsy
   for (t in 1:N)
   {
   # use x y to put them towards the end of the alphabetically sorted  mcmc object
   #SP[t] <- pow(r.pella,-(m-1))*SB[t]*(1-pow(P[t],m-1))
```

```
   BtoBmsy[t] <- SB[t]/SBmsy
   HtoHmsy[t] <- H[t]/(Hmsy)

   #modified for territorial HCR:
   Overfishing_ind[t]<-ifelse(BtoBmsy[t]>0.7, H[t]/(Hmsy),  H[t]/((Hmsy*SB[t])/(0.7*SBmsy)))
   }


   # Enforce soft penalty on K if < K_bounds >
   K.pen ~ dnorm(penK,1000) # enforce penalty
   penK  <- ifelse(K<(K_bounds[1]),log(K)-log(K_bounds[1]),ifelse(K>K_bounds[2],log(K)-
log(K_bounds[2]),0)) # penalty if Pmean is outside viable biomass


   # Enforce soft penalty on process deviance if sigma.proc > 0.2
   proc.pen ~ dnorm(penProc,1000) # enforce penalty
   penProc  <- ifelse(sigma>sigmaproc_bound,log(sigma)-log(sigmaproc_bound),0)


   # Enforce soft penalty on observation error if sigma.obs > sigma_bound
   for(i in 1:nvar){
   obs.pen[i] ~ dnorm(penObs[i],1000) # enforce penalty
   penObs[i]  <- ifelse(pow(tau2[i],0.5)>sigmaobs_bound,log(pow(tau2[i],0.5))-
log(sigmaobs_bound),0)
   }

   ", append=TRUE)

# PROJECTION
if(Projection==TRUE){
  cat("
     for(i in 1:nTAC){
     # Project first year into the future
     prPmean[1,i] <- ifelse(P[N] > Plim,
     log(max(P[N] +  Hmsy/(1-1/m)*P[N]*(1-pow(P[N],m-1)) - TACint_17/K,0.005)),
     log(max(P[N] +  Hmsy/(1-1/m)*P[N]*(1-pow(P[N],m-1))*4*P[N] - TACint_17/K,0.005)))
     prP[1,i] ~ dlnorm(prPmean[1,i],isigma2)
     # Project all following years
     for(t in 2:pyrs){
     prPmean[t,i] <- ifelse(prP[t-1,i] > Plim,
     log(max(prP[t-1,i] +  Hmsy/(1-1/m)*prP[t-1,i]*(1-pow(prP[t-1,i],m-1)) - TAC[t-
1,i]/K,0.001)),
     log(max(prP[t-1,i] +  Hmsy/(1-1/m)*prP[t-1,i]*(1-pow(prP[t-1,i],m-1))*slope.HS*prP[t-1,i] -
TAC[t-1,i]/K,0.005)))
     # process error (as monte-carlo simular)
     prP[t,i] ~ dlnorm(prPmean[t,i],isigma2)}
```

```
      for(t in 1:pyrs){
      prB[t,i] <- prP[t,i]*K
      prH[t,i] <- TAC[t,i]/prB[t,i]
      prHtoHmsy[t,i] <- prH[t,i]/Hmsy
      prBtoBmsy[t,i] <- prB[t,i]/SBmsy
      prOverfishing_ind[t,i]<-ifelse(prBtoBmsy[t,i]>0.7, prH[t,i]/(Hmsy),
prH[t,i]/((Hmsy*prB[t,i])/(0.7*SBmsy))) #JS 2-13-19
      }}
      ",append=TRUE)} else {
       cat("
          #Prevent error for unused input
          fakeTAC <-  TAC
          fakepyrs <- pyrs
          fakenTAC <- nTAC
          fakeTACint <- TACint
          prHtoHmsy <- 1
          prP <- 1
          prBtoBmsy <- 1
          prOverfishing_ind<-1
          ", append=TRUE)}

cat("

} # END OF MODEL
    ",append=TRUE,fill = TRUE)
sink()


ptm <- proc.time()

mod <- jags(surplus.dat, inits,params,paste(JABBA), n.chains = nc, n.thin = nt, n.iter = ni,
n.burnin = nb)  # adapt is burn-in

proc.time() - ptm
save.time = proc.time() - ptm

cat(paste0("\n",paste0("><> Scenario ",Scenario,"_",Mod.names," completed in
",as.integer(save.time[3]/60)," min and ",round((save.time[3]/60-
as.integer(save.time[3]/60))*100)," sec <><","\n")))

cat(paste0("\n","><> Produce results output of ",Mod.names," model for ",assessment,"
",Scenario," <><","\n"))

# if run with library(rjags)
posteriors = mod$BUGSoutput$sims.list
```

```
#-------------------------------------------------------
# <><><><><><><><><><>< Outputs ><><><><><><><><><><><><>
#-------------------------------------------------------
output.dir = paste0(File,"/",assessment,"/",Scenario,"_",Mod.names,"/Output")
dir.create(output.dir, showWarnings = FALSE)

# run some mcmc convergence tests
par.dat= data.frame(posteriors[params[c(1:7)]])

geweke1 = geweke.diag(data.frame(par.dat[1:(nsaved/nc),]))
geweke2 = geweke.diag(data.frame(par.dat[(nsaved/nc+1):nsaved,]))
pvalues1 <- 2*pnorm(-abs(geweke1$z))
pvalues2 <- 2*pnorm(-abs(geweke2$z))

heidle1 = heidel.diag(data.frame(par.dat[1:(nsaved/nc),]))
heidle2 = heidel.diag(data.frame(par.dat[(nsaved/nc+1):nsaved,]))

autocorr1=lapply(par.dat[1:(nsaved/nc),],FUN=acf,plot=F)
autocorr2=lapply(par.dat[(nsaved/nc+1):nsaved,],FUN=acf,plot=F)
max.acK=pmax(abs(autocorr1$K$acf),abs(autocorr1$K$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acr=pmax(abs(autocorr1$r$acf),abs(autocorr2$r$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acq=pmax(abs(autocorr1$q$acf),abs(autocorr2$q$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acpsi=pmax(abs(autocorr1$psi$acf),abs(autocorr2$psi$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acsigma2=pmax(abs(autocorr1$sigma2$acf),abs(autocorr2$sigma2$acf))[c(2,6)] #max lag1
and lag5 autocorrelation across chains
max.actau2=pmax(abs(autocorr1$tau2$acf),abs(autocorr2$tau2$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acm=pmax(abs(autocorr1$m$acf),abs(autocorr2$m$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.ac=rbind(max.acK,max.acr,max.acq,max.acpsi,max.acsigma2,max.actau2,max.acm)

# postrior means + 95% BCIs
#Model  parameter
apply(par.dat,2,quantile,c(0.025,0.5,0.975))

man.dat = data.frame(posteriors[params[8:10]])
#Management quantities
apply(man.dat,2,quantile,c(0.025,0.5,0.975))

# Depletion
```

```
Depletion = posteriors$P[,c(1,n.years)]
colnames(Depletion) = c(paste0("P",years[1]),paste0("P",years[n.years]))

# Current stock status (Kobe posterior)
H_Hmsy.cur = posteriors$HtoHmsy[,c(n.years)]
B_Bmsy.cur = posteriors$BtoBmsy[,c(n.years)]


# Prepare posterior quantities
man.dat = data.frame(man.dat,Depletion,B_Bmsy.cur,H_Hmsy.cur)

results = round(t(cbind(apply(par.dat,2,quantile,c(0.025,0.5,0.975)))),6)

results = data.frame(Median =
results[,2],LCI=results[,1],UCI=results[,3],Geweke.p1=round(pvalues1,3),Geweke.p2=round(pv
alues2,3),Heidel.p1 = round(heidle1[,3],3),Heidel.p2 =
round(heidle2[,3],3),Heidel.it1=heidle1[,2],Heidel.it2=heidle2[,2],Heidel.hw.pass1=heidle1[,4]=
=1,Heidel.hw.pass2=heidle2[,4]==1,lag1=max.ac[,1],lag5=max.ac[,2]) #added all heidleberger
and welch diagnostics 10-25-2018

ref.points = round(t(cbind(apply(man.dat,2,quantile,c(0.025,0.5,0.975)))),3)

ref.points = data.frame(Median = ref.points[,2],LCI=ref.points[,1],UCI=ref.points[,3])

# get number of parameters
npar = length(par.dat)
# number of years
N=n.years

# Save posteriors (Produces large object!)
if(save.all==TRUE) save(posteriors,file=paste0(output.dir,"/",Scenario,"_posteriors"))

#------------------------------------------------------------------------
# Save parameters, results table and current status posterior in csv files
#------------------------------------------------------------------------

# Save model estimates and convergence p-values
write.csv(data.frame(results),paste0(output.dir,"/Estimates_",assessment,"_",Scenario,".csv"))

# Make standard results table with parameter estimates and reference points
Table = rbind(data.frame(results)[,1:3],data.frame(ref.points))
Table[4,] = round(sqrt((Table[4,])),3)
rownames(Table)[4] = "sigma.proc"
write.csv(Table,paste0(output.dir,"/Results_",assessment,"_",Scenario,".csv"))
#Save posterior of recent assessment year (KOBE posterior)
```

```
write.csv(data.frame(BtoBmsy=B_Bmsy.cur,FtoFmsy=H_Hmsy.cur),paste0(output.dir,"/Status_
posterior",assessment,".csv"))

## source all plotting scripts
source(paste0(JABBA.file,'/plot_JABBA_report_combined.R'))


if(save.trajectories==TRUE){
  cat(paste0("\n","><> Saving Posteriors of FRP trajectories <><","\n"))

  # FRP trajectories
  trajectories = array(NA,c(nsaved,n.years,3))
  trajectories[,,1] = posteriors$P
  trajectories[,,2] = posteriors$BtoBmsy
  trajectories[,,3] = posteriors$Overfishing_ind


  kb=kobeJabba(trajectories,years[1])
  save(kb,file=paste0(output.dir,"/",Scenario,"_trajectories"))

}

cat(paste0("\n","><> Scenario ",Mod.names,"_",Scenario," for ",assessment," - DONE!
<><","\n"))
```

1.3. American Samoa

1.3.1. Prime file to execute model in JABBA

```
##
#Script to start base case JABBA model.
##

rm(list=ls())

# required packages
library(gplots); library(coda); library(rjags); library(R2jags); library("fitdistrplus");
library(reshape)

# Set Working directory file, where assessments are stored
File = "Location where folder for assessment is stored"

# Set working directory for JABBA R source code
JABBA.file = "Folder where the JABBA script is located"
```

```
# JABBA version
version = "v1.2_catchCV_lbs_combined"
# Set Assessment file: Data file names must match
assessment = "AmSam_base_FINAL"


#><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>
# Graphic, Output, Saving (.RData) settings
#><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>
KOBE.plot = TRUE # Produces JABBA Kobe plot
KOBE.type = c("ICCAT","IOTC")[2] # ICCAT uses 3 colors; IOTC 4 (incl. orange)
Biplot= TRUE # Produces a "post-modern" biplot with buffer and target zones (Quinn & Collie
2005)
SP.plot = c("standard","phase")[2] # Produces standard or 'Kobe phase' SP plot
save.trajectories =TRUE # saves posteriors of P=B/K, B/Bmsy and H/Hmsy as .RData object
harvest.label = c("Hmsy","Fmsy")[1] # choose label preference H/Hmsy versus Fmsy
CPUE.plot= FALSE # Runs state-tool to produce "aligned" multi-CPUE plot
meanCPUE = FALSE # Uses averaged CPUE from state-space tool instead of individual indices
Projection = TRUE # Use Projections: requires to define TACs vectors
save.projections = TRUE # saves projection posteriors as .RData object
catch.metric = "(lb)" # Define catch input metric e.g. (tons) "000 t" etc
Reproduce.seed = TRUE # If FALSE a random seed assigned to each run, if TRUE set.seed(123)
# P_bound = c(0.02,1.2)  # Soft penalty bounds for P
# Save entire posterior as .RData object
save.all = TRUE # (if TRUE, a very large R object of entire posterior is saved)
#><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>

#><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>
# Optional: Note Scenarios
#><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>
# Specify Scenario name for output file names
Scenarios = c("catchCV",NA,NA,NA)

# Execute multiple JABBA runs in loop

for(s in 1:1){

 Scenario = Scenarios[s]

  #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>
  # Surplus Production model specifications
  #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>

  # Choose model type:
  # 1: Schaefer
  # 2: Fox
```

```
 # 3: Pella-Tomlinson

 Model = c(4,4,4,4)[s]
 Mod.names = c("Schaefer","Fox","Pella","Pella_m")[Model]

 # Depensation option:
 # Set Plim = Blim/K where recruitment may become impaired (e.g. Plim = 0.25)
 # Choose Plim = 0 to reduce to conventional Schaefer, Fox, Pella models
 Plim = 0

 # Required specification for Pella-Tomlinson (Model = 3/4)
 BmsyK = 0.5 # Set Surplus Production curve inflection point
 shape.CV = sqrt(log(1+0.5^2)) # Must be defined if Model = 4!  i used 0.83 for 100%      1.04 =
140%
 #shape.CV = sqrt(log(1+1^2)) - Brian used == 0.83
 #><><><><><><><><><><><><><><><><><><><><><><><>

 #--------------------------------------------------
 # Read csv files
 #--------------------------------------------------

 # Use SEs from csv file for abudance indices (TRUE/FALSE)
 SE.I = TRUE
 CV.C = TRUE
 # Load assessment data
 catch = read.csv(paste0(File,"/",assessment,"/catch",assessment,".csv"))
 cpue = read.csv(paste0(File,"/",assessment,"/cpue",assessment,".csv"))
# cpue = read.csv(paste0(File,"/",assessment,"/cpue_gear_",assessment,".csv")) #fake cpue after
2003 - rnorm with no trend
 if(SE.I ==TRUE){
  se =  read.csv(paste0(File,"/",assessment,"/se",assessment,".csv"))
 }
 if(CV.C == TRUE){
  cv.c = read.csv(paste0(File,"/",assessment,"/catchCV",assessment,".csv"))
 }
 indices2 = names(cpue)[-1]
 wink.colors = data.frame(idx = indices2,
                          cols = c('#e6194b', "#3cb44b", "#ffe119",
                                   "#0082c8","#f58231", "#911eb4",
                                   "#46f0f0", "#f032e6", "#d2f53c",
                                   "#fabebe", "#008080","#e6beff", "#aa6e28")[1:length(indices2)])


 #------------------------------------------------------
 # Option use mean CPUE from state-space cpue averaging
 #-----------------------------------------------------
```

```
  meanCPUE = FALSE

  #-------------------------------------------------
  # Starting value option for r and K
  #-------------------------------------------------

  #-----------------------------------------------
  # Prior for unfished biomass K
  #-----------------------------------------------
  # The options are:
  # a) Specify as a lognormal prior with mean and CV
  # b) Specify as range to be converted into lognormal prior

  # ><> new objective K prior
  K.dist <- c("lnorm","range")[1] # ><> to range
  # Get low and upper r quantile 10th and 90th
  qrs <- qlnorm(c(0.1,0.9),log(0.27),0.3)
  #Apply CMSY Eq 3 by Froese et al. (2017)
  Klow <- max(catch[,2])/qrs[2]
  Khigh <- 4*max(catch[,2])/qrs[1]
#  K.prior <- c(Klow,Khigh)
  K.prior <- c(4*75/0.46,0.5)#650,000 for r = 0.46; 909000 for r = 0.33
  #-------------------------------------------------------
  # mean and CV and sd for Initial depletion level P1= SB/SB0
  #-------------------------------------------------------
  # Set the initial depletion prior B1/K
  # To be converted into a lognormal prior (with upper bound at 1.1)

  psi.dist = c("lnorm","beta")[1]
  # specify as mean and CV
  psi.prior.mean <- signif(as.numeric(3.6556/quantile(cpue[,2], na.rm = TRUE, #
                                        probs = 0.95)), digits = 2) ## uci over 95th
 # psi.prior = c(psi.prior.mean,0.5)
  psi.prior = c(0.8,0.5)
  #-------------------------------------------------------
  # Determine estimation for catchability q and observation error
  #-------------------------------------------------------
  # Assign q to CPUE
  sets.q = 1:(ncol(cpue)-1)

  #-------------------------------------------------
  # Determine r prior
  #-------------------------------------------------
  # The options are:
  # a) Specifying a lognormal prior
  # b) Specifying a resilience category after Froese et al. (2017; CMSY)
```

```
  # Resilience: "Very low", "Low", "Medium", High" (requires r.range = TRUE)

  # use [1] lognormal(mean,stdev) or [2] range (min,max) or
  r.dist = c("lnorm","range")[1]
#  r.prior = c(0.2735,0.3)
   r.prior =c(0.46,sqrt(log(1+0.5^2))) #needs to be sd on log-scale
#  r.prior = c(0.16,0.5) #range

  #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>>
  # Observation Error
  #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>>

  #To Estimate additional observation variance set sigma.add = TRUE
  sigma.est = TRUE

  # Series
  sets.var = 1:(ncol(cpue)-1) # estimate individual additional variance

  # As option for data-weighing
  # minimum fixed observation error for each variance set (optional choose 1 value for both)
  fixed.obsE = c(0.0) # Important if SE.I is not available

  # Total observation error: TOE = sqrt(SE^2+sigma.est^2+fixed.obsE^2)

  #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>>
  # Process Error
  #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>>
  #Estimate set sigma.proc == True
  sigma.proc = TRUE
  # Determines if process error deviation are estimated for all years (TRUE)
  # or only from the point the first abundance index becomes available (FALSE)
  proc.dev.all = TRUE
  #------------------------------------------
  if(sigma.proc == TRUE){
    igamma = c(0.2,0.1) #specify inv-gamma parameters  WAS 4,0.01

    # Process error check
    gamma.check = 1/rgamma(1000,igamma[1],igamma[2])
    # check mean process error + CV
    # mu.proc = sqrt(mean(gamma.check)); CV.proc =
sd(sqrt(gamma.check))/mean(sqrt(gamma.check))

    # check CV
    # round(c(mu.proc,CV.proc),3)
    # quantile(sqrt(gamma.check),c(0.1,0.9))
  }else{
```

```
  sigma.proc = 0.1 #IF Fixed: typicallly 0.05-0.15 (see Ono et al. 2012)
 }
 #-------------------------------------------

 #><><><><><><><><><><><><><><><><><><><><><><><><><><>
 #Include variation in catch estimates
 #><><><><><><><><><><><><><><><><><><><><><><><><><><>
 Catch.CV=TRUE

 #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>>
 # Optional: Do TAC Projections
 #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>>
 Projection = FALSE # Switch on by Projection = TRUE
 #Projection = TRUE # <---- USE THIS ONE FOR PROJECTIONS

 # Set range for alternative TAC projections
 TACs = seq(0,200,1)

 TACint = mean(catch[nrow(catch)-2,2]:catch[nrow(catch),2]) # avg last 3 years  also change to
-2.  was -3
 TACint_17 <- catch[nrow(catch),2]

 # Set number of projections years
 pyrs = 8

 #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>
 # Execute model and produce output
 #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>

 # MCMC settings
 ni <- 300000 # Number of iterations  42500
 nt <- 5 # Steps saved
 #nt <- 50 #steps saved <----USE THIS ONE FOR PROJECTIONS
 nb <- 75000 # Burn-in  17500
 nc <- 2 # number of chains
 nsaved = (ni-nb)/nt*nc

 source(paste0(JABBA.file,"/JABBA",version,".R"))

}# THE END
```

1.3.2. Source code for JABBA model and projections

```
cat(paste0("\n","><>><>><>><>><>><>><>><>"))
cat(paste0("\n","><> Run Model ",Mod.names,"<><"))
```

```
cat(paste0("\n","><><><><><><><><><><>","\n","\n"))
# setwd(paste(File))
dir.create(paste0(File,"/",assessment),showWarnings = FALSE)
dir.create(paste0(File,"/",assessment,"/",Scenario,"_",Mod.names),showWarnings = FALSE)
dir.create(paste0(File,"/",assessment,"/",Scenario,"_",Mod.names,"/Input"),showWarnings =
FALSE)
input.dir = paste0(File,"/",assessment,"/",Scenario,"_",Mod.names,"/Input")

#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
>><>
# Define objects to make sure they exist if not included in Prime file
#><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><><
>><>
if(exists("igamma")==FALSE) igamma = c(0.2,0.1)  # Generic process error prior  WAS 4,0.01
if(exists("BmsyK")==FALSE) BmsyK = 0.4  # JABBA default for Pella model
if(exists("Model")==FALSE){ model = 1; Mod.names = c("Schaefer")} # Run Schaefer if model
is not specified
if(exists("proc.dev.all")==FALSE) proc.dev.all = FALSE # process error deviation if only catch
is available
if(exists("Plim")==FALSE) Plim = 0  # Standard non-compound model
if(exists("P_bound")==FALSE) P_bound = c(0.02,1)  # Soft penalty bounds for P
if(exists("KOBE.plot")==FALSE) KOBE.plot = TRUE # Produces JABBA Kobe plot
if(exists("KOBE.type")==FALSE) KOBE.type = c("ICCAT","IOTC")[2] # ICCAT uses 3
colors; IOTC 4 (incl. orange)
if(exists("SP.plot")==FALSE) SP.plot = c("standard","phase")[2] # Produces standard or 'Kobe
phase' SP plot
if(exists("Biplot")==FALSE) Biplot= TRUE # Produces a "post-modern" biplot with buffer and
target zones (Quinn & Collie 2005)
if(exists("save.trajectories")==FALSE) save.trajectories =FALSE # saves posteriors of P=B/K,
B/Bmsy and H/Hmsy as .RData object
if(exists("harvest.label")==FALSE) harvest.label = c("Hmsy","Fmsy")[2] # choose label
preference H/Hmsy versus Fmsy
if(exists("CPUE.plot")==FALSE) CPUE.plot= TRUE # Runs state-tool to produce "aligned"
multi-CPUE plot
if(exists("catch.metric")==FALSE) catch.metric = "(t)" # Runs state-tool to produce "aligned"
multi-CPUE plot
if(exists("meanCPUE")==FALSE) meanCPUE = FALSE # Uses averaged CPUE from state-
space tool instead of individual indices
if(exists("Projection")==FALSE) Projection = FALSE # Use Projections: requires to define
TACs vectors
if(exists("save.projections")==FALSE) save.projections = FALSE# saves projection posteriors
as .RData object
if(exists("Reproduce.seed")==FALSE) Reproduce.seed = FALSE # If FALSE a random seed
assigned to each run (default)
if(exists("TACint")==FALSE) TACint = mean(catch[nrow(catch)-3,2]:catch[nrow(catch),2]) #
use mean catch from last years
```

```
if(exists("imp.yr")==FALSE) imp.yr = as.numeric(format(Sys.Date(), "%Y"))+1 # use next year
from now
if(exists("init.values")==FALSE) init.values =FALSE # Allows to add manual starting values for
K, r, q
if(exists("sigmaobs_bound")==FALSE) sigmaobs_bound = 1 # Adds an upper bound to the
observation variance
if(exists("sigmaproc_bound")==FALSE) sigmaproc_bound = 0.2 # Adds an upper bound to the
process variance
if(exists("q_bounds")==FALSE) q_bounds= c(10^-10,10) # Defines lower and upper bounds for
q
if(exists("K_bounds")==FALSE) K_bounds= c(10,10^8) # Defines lower and upper bounds for q
# Save entire posterior as .RData object
if(exists("save.all")==FALSE) save.all = FALSE #
#><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><
>

## Prepare input data ----

cat(paste0("\n","><> Prepare input data <><","\n"))
indices = names(cpue)[2:ncol(cpue)]
n.indices = max(length(indices),1)
catches = names(catch)[2:ncol(catch)]
n.catches = length(catches)

years=catch[,1]
styr = min(years)
endyr = max(years)
n.years = length(years)
styr.cpue = min(cpue[,1])
styr.I = styr.cpue-styr+1


# Convert input data to matrices for JAGS input
conv.cpue = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.indices),styr.I-
1,n.indices),as.matrix(cpue[,-1])))
CPUE=matrix(conv.cpue,nrow=n.years,ncol=n.indices)


if(SE.I==FALSE){
  se = cpue
  conv.se = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.indices),styr.I-
1,n.indices),as.matrix(cpue[,-1])))
  se2 = matrix(ifelse(fixed.obsE>0,fixed.obsE^2,10^-10),n.years,n.indices)#/2
} else{
  conv.se = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.indices),styr.I-
1,n.indices),as.matrix(se[,-1])))
```

```
  #conv.se = sqrt(conv.se^2+fixed.obsE^2)
  se2 = matrix(ifelse(is.na(conv.se),0.3^2,conv.se)^2,n.years,n.indices)+fixed.obsE^2#/2
}

if(Catch.CV==FALSE){
  conv.catch = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.catches),styr.I-
1,n.catches),as.matrix(catch[,-1])))
  Catch=matrix(conv.catch,nrow=n.years,ncol=n.catches)
  Catch[is.na(Catch)] = 0 # Replace any NA by zero
} else{
  conv.catch = as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.catches),styr.I-
1,n.catches),as.matrix(catch[,-1])))
  Catch=matrix(conv.catch,nrow=n.years,ncol=n.catches)
  Catch[is.na(Catch)] = 0 # Replace any NA by zero

  conv.catch.cv=as.numeric(rbind(matrix(rep(NA,(styr.I-1)*n.catches),styr.I-
1,n.catches),as.matrix(cv.c[,-1]))) #assumes we have a cv for every catch
  catch.cv=matrix(conv.catch.cv,nrow=n.years,ncol=n.catches)
  catch.cv[is.na(catch.cv)] = 0 # Replace any NA by zero
}


# Total Catch
TC = apply(Catch,1,sum)

# Plot Catch

cat(paste0("\n","><> Plot Catch in Input subfolder <><","\n"))

Par = list(mfrow=c(1,1),mar = c(5, 5, 1, 1), mgp =c(3,1,0), tck = -0.02,cex=0.8)
png(file = paste0(input.dir,"/Catches_",assessment,".png"), width = 7, height = 5,
    res = 200, units = "in")
par(Par)
plot(catch[,1],catch[,1],ylim=c(0,max(catch[,2:ncol(catch)],na.rm=TRUE)),ylab=paste0("Catch
",catch.metric),xlab="Year",type="n")
for(i in 2:ncol(catch)) lines(catch[,1],catch[,i],lty=(i-1),lwd=2)
legend("topright",paste(names(catch)[2:ncol(catch)]),lty=1:(ncol(catch)-1),bty="n")
dev.off()

#---------------------
# Index color palette
#---------------------
jabba.colors = as.character(rep(c('#e6194b', "#3cb44b", "#ffe119",
                            "#0082c8","#f58231", "#911eb4",
                            "#46f0f0", "#f032e6", "#d2f53c",
                            "#fabebe", "#008080","#e6beff", "#aa6e28"),2))
```

```
#--------------------
# Set seed
#--------------------
if(Reproduce.seed==FALSE){ set.seed(ceiling(runif(1,min=0,max=1e6))) } else {set.seed(123)}

#-------------------------------------------------------------------------
# CPUE run State-Space model for averaging CPUE
#-------------------------------------------------------------------------
if(CPUE.plot==TRUE){
  cat(paste0("\n","><> Run State-Space CPUE averaging tool","\n"))
  #find first time-series with first CPUE
  q1.y = c(1:n.years)[is.na(apply(CPUE,1,mean,na.rm=TRUE))==FALSE][1] #first year with
CPUE
  q1.I = which.max(CPUE[q1.y,])

  qs = c(q1.I,c(1:(ncol(cpue)-1))[-q1.I])


  sink("cpueAVG.jags")
  cat("
      model {

      # Prior specifications
      eps <- 0.0000000000001 # small constant

      iq[1] ~ dgamma(1000,1000)
      q[1] <-  pow(iq[1],-1)
      logq[1] <- log(1)
      for(i in 2:nI){
      iq[i] ~ dgamma(0.001,0.001)
      q[i] <- pow(iq[i],-1)
      logq[i] <-  log(q[i])
      }


      ")

  if(sigma.proc==TRUE){
    cat("
        # Process variance
        isigma2 <- isigma2.est
        sigma2 <- pow(isigma2,-1)
        sigma <- sqrt(sigma2)
        fakesigma.fixed <- sigma.fixed # Prevent unused variable error msg
        ",append=TRUE)
  }else{ cat("
```

```
   isigma2 <- pow(sigma.fixed+eps,-2)
        sigma2 <- pow(isigma2,-1)
        sigma <- sqrt(sigma2)

        ",append=TRUE)}

if(sigma.est==TRUE){
  cat("
     # Observation variance
     # Observation error
     itau2~ dgamma(0.001,0.001)
     tau2 <- 1/itau2


     for(i in 1:nI)
     {
     for(t in 1:N)
     {
     var.obs[t,i] <- SE2[t,i]+tau2
     ivar.obs[t,i] <- 1/var.obs[t,i]
     # note total observation error (TOE)
     TOE[t,i] <- sqrt(var.obs[t,i])

     }}
     ",append=TRUE)
}else{ cat("
   # Observation variance
        # Observation error
        itau2~ dgamma(2,2)
        tau2 <- 1/itau2


        for(i in 1:nI)
        {
        for(t in 1:N)
        {
        var.obs[t,i] <- SE2[t,i] # drop tau2
        fake.tau[t,i] <- tau2

        ivar.obs[t,i] <- 1/var.obs[t,i]
        # note total observation error (TOE)
        TOE[t,i] <- sqrt(var.obs[t,i])

        }}

        ",append=TRUE)}
```

```
  # Run rest of code
  cat("
     # Process variance prior
     isigma2.est ~ dgamma(0.001,0.001)


     # Priors and constraints
     logY.est[1] ~ dnorm(logY1, 1)       # Prior for initial population size

     mean.r ~ dnorm(1, 0.001)            # Prior for mean growth rate

     # Likelihood
     # State process
     for (t in 1:(N-1)){
     r[t] ~ dnorm(mean.r, isigma2)
     logY.est[t+1] <- logY.est[t] + r[t] }

     # Observation process
     for (t in 1:N) {
     for(i in 1:nI){
     y[t,i] ~ dnorm(logY.est[t]+logq[i], ivar.obs[t,i])
     }}

     # Population sizes on real scale
     for (t in 1:N) {
     Y.est[t] <- exp(logY.est[t])
     }

  }
     ",fill = TRUE)
  sink()



  q.init = 1
  mCPUE = as.matrix(CPUE[q1.y:n.years,qs])
  mSE2 = as.matrix(se2[q1.y:n.years,qs])
  if(n.indices>1) for(i in 2:n.indices){q.init[i] =
mean(mCPUE[,i],na.rm=TRUE)/mean(mCPUE[,1],na.rm=TRUE)}
  # Bundle data
  jags.data <- list(y = log(mCPUE),SE2=mSE2, logY1 = log(mCPUE[1,1]), N =
length(q1.y:n.years),nI=n.indices,sigma.fixed=ifelse(sigma.proc==TRUE,0,sigma.proc))

  # Initial values
```

```
  inits <- function(){list(isigma2.est=runif(1,20,100), itau2=runif(1,80,200), mean.r = rnorm(1),iq
= 1/q.init)}

  # Parameters monitored
  parameters <- c("mean.r", "sigma","r", "Y.est","q")


  # Call JAGS from R (BRT 3 min)
  mod.cpue <- jags(jags.data, inits, parameters, "cpueAVG.jags", n.chains = nc, n.thin =
max(nt,2), n.iter = max(ni/5,10000), n.burnin = nb/10)


  cat(paste0("\n","><> Plot State-Space CPUE fits in Input subfolder <><","\n"))
  # get individual trends
  fitted <- lower <- upper <- NULL
  cpue.yrs = years[q1.y:n.years]

  for (t in 1:nrow(mCPUE)){
    fitted[t] <- median(mod.cpue$BUGSoutput$sims.list$Y.est[,t])
    lower[t] <- quantile(mod.cpue$BUGSoutput$sims.list$Y.est[,t], 0.025)
    upper[t] <- quantile(mod.cpue$BUGSoutput$sims.list$Y.est[,t], 0.975)}


  q.adj = apply(mod.cpue$BUGSoutput$sims.list$q,2,median)


  Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
  png(file = paste0(input.dir,"/CPUE_",assessment,"_",Scenario,".png"), width = 5, height = 3.5,
      res = 200, units = "in")
  par(Par)
  u.ylim = NULL
  for(i in 1:n.indices){ u.ylim = c(u.ylim,exp(log(mCPUE[,i]/q.adj[i])+1.96*sqrt(mSE2[,i])))}
  ylim = c(0,max(u.ylim,na.rm=TRUE))
  plot(0, 0, ylim = ylim, xlim = range(cpue.yrs), ylab = "Expected CPUE", xlab = "Year", col =
"black", type = "n")
  legend("topright",paste(indices),lwd=2,col=(jabba.colors)[1:n.indices],bty="n")
  polygon(x = c(cpue.yrs,rev(cpue.yrs)), y = c(lower,rev(upper)), col = "gray", border =
"gray90")

  for(i in 1:n.indices)
  {
    shift = runif(1,-0.1,0.1)
    cols=jabba.colors[qs[i]]

plotCI(cpue.yrs+shift,mCPUE[,i]/q.adj[i],ui=exp(log(mCPUE[,i]/q.adj[i])+1.96*sqrt(mSE2[,i])),
```

```
li=exp(log(mCPUE[,i]/q.adj[i])-1.96*sqrt(mSE2[,i])),add=TRUE,col= cols,pt.bg =
cols,pch=21,gap=0)
   lines(cpue.yrs+shift,mCPUE[,i]/q.adj[i], col = cols,lwd=2)
   points(cpue.yrs+shift,mCPUE[,i]/q.adj[i], bg = cols,pch=21)
  }
  lines(cpue.yrs,fitted,lwd=2)

  dev.off()

  logSE = apply(log(mod.cpue$BUGSoutput$sims.list$Y.est),2,sd)


  if(nrow(mCPUE)<n.years) {
   fitted = c(rep(NA,q1.y-1),fitted)
   logSE = c(rep(0.2,q1.y-1),logSE)
  }
  avgCPUE = data.frame(Year=years,CPUE= fitted,logSE=logSE)

  write.csv(avgCPUE,paste0(input.dir,"/avgCPUE_",assessment,"_",Scenario,".csv"))

  if(meanCPUE==TRUE){
   cat(paste0("\n","><> Use average CPUE as input for JABBA <><","\n"))

   CPUE = as.matrix(avgCPUE[,2])
   cpue.check = cpue[,-1]
   cpue.check[is.na(cpue[,-1])]=0
   CPUE[,1] = ifelse(apply(cpue.check,1,sum)==0,rep(NA,length(CPUE[,1])),CPUE[,1])
   se2 =  as.matrix(avgCPUE[,3]^2)
   n.indices=1
   indices = "All"
   sets.q =1
   sets.var =1
  }

  }

#------------------------------------------------------------------------
# END of CPUE State-Space tool
#------------------------------------------------------------------------

#-----------
# FUNCTIONS
#-----------
cat(paste0("\n","><> Prepare JABBA prior inputs <><","\n"))

#-----------------------------------------------
```

```
# Function to get beta prior parameters
#-------------------------------------------------
get_beta <- function(mu,CV,Min=0,Prior="x"){
  a = seq(0.0001,1000,0.001)
  b= (a-mu*a)/mu
  s2 = a*b/((a+b)^2*(a+b+1))
  sdev = sqrt(s2)
  # find beta )parameter a
  CV.check = (sdev/mu-CV)^2
  a = a[CV.check==min(CV.check)]
  #find beta parameter b
  b = (a-mu*a)/mu
  x = seq(Min,1,0.001)
  pdf = dbeta(x,a,b)
  plot(x,pdf,type="l",xlim=range(x[pdf>0.01]),xlab=paste(Prior),ylab="",yaxt="n")
  polygon(c(x,rev(x)),c(rep(0,length(x)),rev(ifelse(pdf==Inf,100000,pdf))),col="grey")
  return(c(a,b))
}

#-------------------------------------------------
# Function to get gamma prior parameters
#-------------------------------------------------

get_gamma <- function(mu,CV,Prior="x"){
  a = seq(0.0001,1000,0.0001)
  b = a/mu
  s2 = (a/b^2)
  sdev = sqrt(s2)
  # find beta )parameter a
  CV.check = (sdev/mu-CV)^2
  a = a[CV.check==min(CV.check)]
  #find beta parameter b
  b = a/mu
  x = sort(rgamma(1000,a,b))
  pdf = dgamma(x,a,b)
  plot(x,pdf,type="l",xlim=range(x[pdf>0.01]),xlab=paste(Prior),ylab="",yaxt="n")
  polygon(c(x,rev(x)),c(rep(0,length(x)),rev(ifelse(pdf==Inf,100000,pdf))),col="grey")
  return(c(a,b))
}


#-------------------------------------------------
# Function to get lognormal prior parameters
#-------------------------------------------------
plot_lnorm <- function(mu,CV,Prior="x"){
  sdev= sqrt(log(CV^2+1))
```

```
  rand.pr = rlnorm(1000,log(mu),sdev)
  x = seq(min(rand.pr),quantile(rand.pr,0.995),max(rand.pr/500))
  pdf = dlnorm(x,log(mu),sdev)
  plot(x,pdf,type="l",xlim=range(x),xlab=paste(Prior),ylab="",yaxt="n")
  polygon(c(x,rev(x)),c(rep(0,length(x)),rev(ifelse(pdf==Inf,100000,pdf))),col="grey")
  return(c(mu,sdev))
}

#-----------------------------------
# Function kobeJabba for FLR
#-----------------------------------
kobeJabba<-function(x,minyear=1){

  out=cbind(melt(x[,,2]),c(x[,,3]))
  names(out)=c("iter","year","stock","harvest")
  out$year=out$year+minyear-1
  out}

#-------------------------------------------------
# Function kobeJabbaProj for projections with FLR
#-------------------------------------------------
kobeJabbaProj<-function(x,minyear=1,tac=NULL){

  out=cbind(melt(x[,,,2]),c(x[,,,3]),c(x[,,,4]))
  names(out)=c("iter","year","tac","stock","harvest","Hrate")
  out$year=out$year+minyear-1

  out}

#-------------------------------------------------
# Determine initial ranges for r
#-------------------------------------------------
if(r.dist=="range"){
  # initial range of r based on resilience (FishBase.org)
  if(length(r.prior)>1){ start.r = r.prior} else
    if(r.prior == "High") {
      start.r <- c(0.6,1.5)} else if(r.prior == "Medium") {
        start.r <- c(0.2,0.8)}    else if(r.prior == "Low") {
          start.r <- c(0.05,0.5)}  else { # i.e. res== "Very low"
            start.r <- c(0.015,0.1)}

  log.r = mean(log(start.r))
  sd.r = abs(log.r - log(start.r[1]))/2
  r.prior = c(exp(log.r),sd.r)
  CV.r = sqrt(exp(sd.r^2)-1)
} else {
```

```
  log.r = log(r.prior[1])
  sd.r = r.prior[2]
  CV.r = sqrt(exp(sd.r^2)-1)
}

#-------------------------------------------------
# Prepare K prior
#-------------------------------------------------
if(K.dist=="range"){
  log.K = mean(log(K.prior))
  sd.K= abs(log.K - log(K.prior[1]))/2
  CV.K = sqrt(exp(sd.K^2)-1)
} else {

  log.K = log(K.prior[1])
  CV.K = K.prior[2]
  sd.K=sqrt(log(CV.K^2+1))
}


#-----------------------------------------------------------
# Get JABBA parameterization and surplus production function
#-----------------------------------------------------------
# For Pella-Tomlinson
if (Model == 3 | Model == 4) {
  ## run thru sensitivities if given shape
  # find inflection point
  ishape = NULL
  # Find shape for  SBmsytoK
  ishape = seq(0.1, 10, 0.001)
  check.shape = ((ishape) ^ (-1 / (ishape - 1)) - BmsyK) ^ 2
  #  Set shape (> 0, with 1.001 ~ Fox and 2 = Schaefer)
  shape =  ishape[check.shape == min(check.shape)]

  if (exists("sensname")) {
    if (sensname == 'M') {
      m = shape = sensmean[s] ## becomes m.mu
      shape.CV = sensvar[s]
      cat(paste0(shape, " ", shape.CV), "\n")
    } ## end sensname == M
  } ## end sensname exists
} else {shape = FALSE}


#-----------------------------------------------
```

```
# Set shape m for Fox and Schaefer: Fox m ~1; Schaefer m =2
if(shape==FALSE){
  if(Model == 1){m=2} else {m = 1.001}}else{m=shape}

cat(paste0("\n","><> Plot Prior distributions in Input subfolder  <><","\n"))

Par = list(mfrow=c(1,3),mai=c(0.5,0.1,0,.1),omi = c(0.1,0.2,0.1,0) + 0.1,mgp=c(2,1,0), tck = -
0.02,cex=0.8)
png(file = paste0(input.dir,"/Priors_",assessment,"_",Scenario,".png"), width = 9, height = 3,
    res = 200, units = "in")
par(Par)
K.pr = plot_lnorm(exp(log.K),CV.K,Prior="K")

if(psi.dist=="beta"){
  psi.pr = get_beta(mu=psi.prior[1],CV=psi.prior[2],Min=0,Prior=paste0("Prior
B(",years[1],")/K"))} else {
    psi.pr = plot_lnorm(mu=psi.prior[1],CV=psi.prior[2],Prior=paste0("Prior B(",years[1],")/K"))
  }


r.pr = plot_lnorm(mu=exp(log.r),CV=CV.r,Prior="r")
mtext(paste("Density"), side=2, outer=TRUE, at=0.5,line=1,cex=0.9)
dev.off()


cat(paste0("\n","><> Plot assumed Surplus Production shape in Input subfolder  <><","\n"))


# Plot MSY
Par = list(mfrow=c(1,1),mai=c(0.6,0.3,0,.15),omi = c(0.1,0.2,0.2,0) + 0.1,mgp=c(2,1,0), tck = -
0.02,cex=0.8)
png(file = paste0(input.dir,"/Production",assessment,"_",Scenario,".png"), width = 6, height = 5,
    res = 200, units = "in")
par(Par)

# Get Bmsy/B0 as a function of M
Bmsy=(m)^(-1/(m-1))
P = seq(0.0001,1,0.001)
SP = ifelse(P>Plim,r.pr[1]/(m-1)*P*(1-P^(m-1)),r.pr[1]/(m-1)*P*(1-P^(m-1))*4*P)
#if(is.null(refBmsy)==TRUE) refBmsy = Bmsy
plot(P,SP/max(SP),type="l",ylab="Relative Yield",xlab="B/B0",lwd=2)
mtext(paste("Relative Yield"), side=2, outer=TRUE, at=0.6,line=1,cex=0.9)
legend("topright",c("SPM"),col=c(1),lwd=2,bty="n")
```

```
if(Model==4){
  # shape density
  #dm = dgamma(seq(0.001,5,0.1),5,5)*m
  dm = dlnorm((seq(0.001,5,0.1)),log(m),shape.CV)
  dm = dm/max(dm)
  bmsyk  = (seq(0.001,5,0.1))^(-1/(seq(0.001,5,0.1)-1))

  polygon(c(bmsyk,rev(bmsyk)),c(dm,rep(0,length(dm))),col="grey",border=0)
}
abline(v=Bmsy,lty=2)
mtext(paste("Relative Yield"), side=2, outer=TRUE, at=0.6,line=1,cex=0.9)
legend("topright",c("SPM"),col=c(1),lwd=2,bty="n")
abline(v=Bmsy,lty=2)
dev.off()


# Note PRIORS and save input subfolder
Priors =rbind(K.pr,psi.prior,c(r.pr[1],CV.r))
row.names(Priors) = c("K","Psi","r")
colnames(Priors) = c("Mean","CV")
write.csv(Priors,paste0(input.dir,"/Priors",assessment,"_",Scenario,".csv"))


#-------------------------------------------------------------
# Set up JABBA
#-------------------------------------------------------------
cat(paste0("\n","><> Set up JAGS input <><","\n"))
# Plot MSY
# remove scientific numbers
options(scipen=999)
#-------------------------------------------------------------


# starting values
nq = length(unique(sets.q))
nvar = length(unique(sets.var))

## MDFitchett change to TAC setup
#-------------------------------------------------------------
# Setup TAC projection
#-------------------------------------------------------------
if(Projection==TRUE) {
  nTAC = length(TACs)
  TAC = mat.or.vec(pyrs, nTAC)
  yr.now=2019
  yr.last = max(years) # assessment year
```

```
  for (i in 1:nTAC) {
    TAC[, i] = c(rep(TACint, yr.now - yr.last), rep(TACs[i], pyrs - (yr.now -yr.last)))
  }

} else if(Projection == FALSE){
  nTAC = 1
  TAC = TC[n.years]
  pyrs = 1
}

#-----------------------------------------------------------------
# JABBA Schaefer/Fox Models 1-2, Pella 3
#-----------------------------------------------------------------

# Slope of hockey-stick
slope.HS = ifelse(Plim==0,1/10^-10,1/Plim)


nSel = 1 # setup for JABBA-SELECT version (in prep)
nI = ncol(CPUE) # number of CPUE series
stI = ifelse(proc.dev.all==TRUE,1,
c(1:n.years)[is.na(apply(CPUE,1,mean,na.rm=TRUE))==FALSE][1]) #first year with CPUE


# Initial starting values
if(Catch.CV==FALSE){
  inits <- function(){list(K=
rlnorm(1,log(4*75/0.46),0.5),r=rlnorm(1,log(0.46),0.5),m=rlnorm(1,log(2),0.5),
psi=rlnorm(1,log(0.8),0.5), q = runif(1,0.0000000001,10), isigma2.est=rgamma(1,0.2,0.1),
itau2=rgamma(1,0.2,0.1))}
} else {
#inits <- function(){list(K=log(4*55/0.46),r=0.46,m=2,rCatch = TC, psi=0.75, q = 5,
isigma2.est=100, itau2=100)}
  inits <- function(){list(K=
rlnorm(1,log(4*75/0.46),0.5),r=rlnorm(1,log(0.46),0.5),m=rlnorm(1,log(2),0.5),rCatch = TC,
psi=rlnorm(1,log(0.8),0.5), q = runif(1,0.0000000001,10), isigma2.est=rgamma(1,0.2,0.1),
itau2=rgamma(1,0.2,0.1))}
  }

# starting value option
if(init.values==TRUE){
inits <- function(){list(K= K.init,r=r.init,q = q.init, isigma2.est=runif(1,20,100),
itau2=runif(nvar,80,200))}
}
```

```
# JABBA input data
if(Catch.CV==FALSE){
surplus.dat = list(N=n.years, TC = TC, I=CPUE,SE2=se2,mu.m=m,r.pr=r.pr,psi.pr=psi.pr,K.pr =
K.pr,
                   nq=nq,nI = nI,nvar=nvar,sigma.fixed=ifelse(sigma.proc==TRUE,0,sigma.proc),
                   sets.var=sets.var, sets.q=sets.q,pen.bk =
rep(0,n.years),Plim=Plim,slope.HS=slope.HS,
                   nTAC=nTAC,pyrs=pyrs,TAC=TAC,igamma =
igamma,stI=stI,TACint=TACint,TACint_17=TACint_17,P_bound=P_bound,proc.pen=0,K.pen
= 0,
                   obs.pen =
rep(0,nvar),q_bounds=q_bounds,sigmaobs_bound=sigmaobs_bound,sigmaproc_bound=sigmapr
oc_bound,K_bounds=K_bounds)
} else {
surplus.dat = list(N=n.years, TC = TC, catch.cv = catch.cv,
I=CPUE,SE2=se2,mu.m=m,r.pr=r.pr,psi.pr=psi.pr,K.pr = K.pr,
                   nq=nq,nI = nI,nvar=nvar,sigma.fixed=ifelse(sigma.proc==TRUE,0,sigma.proc),
                   sets.var=sets.var, sets.q=sets.q,pen.bk =
rep(0,n.years),Plim=Plim,slope.HS=slope.HS,
                   nTAC=nTAC,pyrs=pyrs,TAC=TAC,igamma =
igamma,stI=stI,TACint=TACint,TACint_17=TACint_17,P_bound=P_bound,proc.pen=0,K.pen
= 0,
                   obs.pen =
rep(0,nvar),q_bounds=q_bounds,sigmaobs_bound=sigmaobs_bound,sigmaproc_bound=sigmapr
oc_bound,K_bounds=K_bounds)
}




# If shape parameter is estimated (Model =4)
if(Model==4){
  surplus.dat$m.CV = shape.CV }


# JAGS model file
JABBA = "JABBA.jags"


# PARAMETERS TO MONITOR -
if(Catch.CV==FALSE){
params <- c("K","r", "q", "psi","sigma2", "tau2","m","Hmsy","SBmsy", "MSY",
"BtoBmsy","HtoHmsy","Overfishing_ind","CPUE","Proc.Dev","P","SB","H","prP","prBtoBms
y","prHtoHmsy", "prOverfishing_ind", "prH","TOE")
} else {
```

```
params <- c("K","r", "q", "psi","sigma2", "tau2","m","Hmsy","SBmsy", "MSY",
"BtoBmsy","HtoHmsy","Overfishing_ind",
"CPUE","Proc.Dev","P","SB","H","prP","prBtoBmsy","prHtoHmsy","prOverfishing_ind","prH"
,"TOE")
}



cat(paste0("\n","><> RUN ",Mod.names," model for ",assessment," ",Scenario," in JAGS
<><","\n","\n"))

# JAGS MODEL Standard
sink("JABBA.jags")
cat("

   model {

   # Prior specifications
   eps <- 0.00000000000000000000000000000001 # small constant

   #Catchability coefficients
   for(i in 1:nq)
   {
   q[i] ~ dunif(q_bounds[1],q_bounds[2])
   }


   ")

if(Model==4){
  cat("
     # Shape m prior
     m ~ dlnorm(log(mu.m),pow(m.CV,-2))
     ",append=TRUE)
  }else{ cat("
     m <- mu.m
    ",append=TRUE)}

if(psi.dist =="beta"){
  cat("
     # Beta Prior for Biomass depletion at the start (deterministic)
     psi ~ dbeta(psi.pr[1],psi.pr[2])
     ",append=TRUE)
} else {
  cat("
     # Lognormal for Biomass depletion at the start (deterministic)
```

```
    psi ~ dlnorm(log(psi.pr[1]),pow(psi.pr[2],-2)) #I(0.1,1.1)
    ",append=TRUE)
}

if(sigma.proc==TRUE){
  cat("
    # Process variance
    isigma2 <- isigma2.est
    sigma2 <- pow(isigma2,-1)
    sigma <- sqrt(sigma2)
    fakesigma.fixed <- sigma.fixed # Prevent unused variable error msg
    ",append=TRUE)
}else{ cat("
    isigma2 <- pow(sigma.fixed+eps,-2)
        sigma2 <- pow(isigma2,-1)
        sigma <- sqrt(sigma2)

        ",append=TRUE)}

if(sigma.est==TRUE){
  cat("
    # Observation variance
    for(i in 1:nvar)
    {
    # Observation error
    itau2[i]~ dgamma(0.2,0.1)   #was 0.001,0.001
    tau2[i] <- 1/itau2[i]
    }

    for(i in 1:nI)
    {
    for(t in 1:N)
    {
    var.obs[t,i] <- SE2[t,i]+tau2[sets.var[i]]
    ivar.obs[t,i] <- 1/var.obs[t,i]
    # note total observation error (TOE)
    TOE[t,i] <- sqrt(var.obs[t,i]) # Total observation variance

    }}
    ",append=TRUE)
}else{ cat("
    # Observation variance
        for(i in 1:nvar)
        {
        # Observation error
        itau2[i]~ dgamma(4,0.01)
```

```
        tau2[i] <- 1/itau2[i]
        }

        for(i in 1:nI)
        {
        for(t in 1:N)
        {
        var.obs[t,i] <- SE2[t,i] # drop tau2
        fake.tau[t,i] <- tau2[sets.var[i]]

        ivar.obs[t,i] <- 1/var.obs[t,i]
        # note total observation error (TOE)
        TOE[t,i] <- sqrt(var.obs[t,i])

        }}

        ",append=TRUE)}

# Process section divided into 2 chunks, replicated for Catch.CV = FALSE/TRUE
if(Catch.CV==FALSE){
cat("
   # Process variance prior
   isigma2.est ~ dgamma(igamma[1],igamma[2])

   # Carrying Capacity SB0
   K ~ dlnorm(log(K.pr[1]),pow(K.pr[2], -2))


   # informative priors for Hmsy as a function of r
   r ~ dlnorm(log(r.pr[1]),pow(r.pr[2],-2))


   #Process equation
   Pmean[1] <- log(psi)
   iPV[1] <- ifelse(1<(stI),10000,isigma2) # inverse process variance
   P[1] ~ dlnorm(Pmean[1],iPV[1]) # set to small noise instead of isigma2
   penB[1]  <- ifelse(P[1]<P_bound[1],log(K*P[1])-
log(K*P_bound[1]),ifelse(P[1]>P_bound[2],log(K*P[1])-log(K*P_bound[2]),0)) # penalty if
Pmean is outside viable biomass

   # Process equation
   for (t in 2:N)
   {
   Pmean[t] <- ifelse(P[t-1] > Plim,
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1)) - TC[t-1]/K,0.005)),
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1))*P[t-1]*slope.HS - TC[t-1]/K,0.005)))
```

```
   iPV[t] <- ifelse(t<(stI),10000,isigma2) # inverse process variance
   P[t] ~ dlnorm(Pmean[t],iPV[t])
   penB[t]  <- ifelse(P[t]<(P_bound[1]),log(K*P[t])-
log(K*(P_bound[1])),ifelse(P[t]>P_bound[2],log(K*P[t])-log(K*(P_bound[2])),0)) # penalty if
Pmean is outside viable biomass
   }

   # Process error deviation
   for(t in 1:N){
   Proc.Dev[t] <- P[t]-exp(Pmean[t])}

   # Enforce soft penalties on bounds for P
   for(t in 1:N){
   pen.bk[t] ~ dnorm(penB[t],1000) # enforce penalty with CV = 0.1
   }
   ", append=TRUE)
} else{ cat("
   # Process variance prior
   isigma2.est ~ dgamma(igamma[1],igamma[2])

   # Carrying Capacity SB0
   K ~ dlnorm(log(K.pr[1]),pow(K.pr[2], -2))

   # informative priors for Hmsy as a function of r
   r ~ dlnorm(log(r.pr[1]),pow(r.pr[2],-2))


#Process equation
    Pmean[1] <- log(psi)
    iPV[1] <- ifelse(1<(stI),10000,isigma2) # inverse process variance
    P[1] ~ dlnorm(Pmean[1],iPV[1]) # set to small noise instead of isigma2
    penB[1]  <- ifelse(P[1]<P_bound[1],log(K*P[1])-
log(K*P_bound[1]),ifelse(P[1]>P_bound[2],log(K*P[1])-log(K*P_bound[2]),0)) # penalty if
Pmean is outside viable biomass
    Catch_precision[1]<-1/log(1+catch.cv[1,]*catch.cv[1,])
    rCatch[1] ~ dlnorm(log(TC[1]),Catch_precision[1])

# Process equation
   for (t in 2:N)
   {
   Pmean[t] <- ifelse(P[t-1] > Plim,
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1)) - rCatch[t-1]/K,0.005)),
   log(max(P[t-1] +  r/(m-1)*P[t-1]*(1-pow(P[t-1],m-1))*P[t-1]*slope.HS - rCatch[t-
1]/K,0.005)))

    iPV[t] <- ifelse(t<(stI),10000,isigma2) # inverse process variance
```

```
     P[t] ~ dlnorm(Pmean[t],iPV[t])
     Catch_precision[t]<-1/log(1+catch.cv[t,]*catch.cv[t,])
     rCatch[t] ~ dlnorm(log(TC[t]),Catch_precision[t])
     penB[t]  <- ifelse(P[t]<(P_bound[1]),log(K*P[t])-
log(K*(P_bound[1])),ifelse(P[t]>P_bound[2],log(K*P[t])-log(K*(P_bound[2])),0)) # penalty if
Pmean is outside viable biomass
     }

   # Process error deviation
    for(t in 1:N){
    Proc.Dev[t] <- P[t]-exp(Pmean[t])}

    # Enforce soft penalties on bounds for P
   for(t in 1:N){
   pen.bk[t] ~ dnorm(penB[t],1000) # enforce penalty with CV = 0.1
    }
          ", append=TRUE)

}


# Run rest of code
    cat("
    Hmsy <- r*pow(m-1,-1)*(1-1/m)

    for (t in 1:N)
    {
    SB[t] <- K*P[t]
    }"
       , append=TRUE)

    #Include calculation of H for random catch
    if(Catch.CV==FALSE){
      cat("
    for (t in 1:N)
    {
    H[t] <- TC[t]/SB[t]
    }
     ", append=TRUE)} else{
      cat("
    for (t in 1:N)
    {
    H[t] <- rCatch[t]/SB[t]
    }
     ", append=TRUE)}
```

```
   # Observation equation in related to EB
   cat("
   for(i in 1:nI)
   {
   for (t in 1:N)
   {
   Imean[t,i] <- log(q[sets.q[i]]*P[t]*K);
   I[t,i] ~ dlnorm(Imean[t,i],(ivar.obs[t,i]));
   CPUE[t,i] <- q[sets.q[i]]*P[t]*K
   }}


   #Management quantities
   SBmsy_K <- (m)^(-1/(m-1))
   SBmsy <- SBmsy_K*K

   MSY <- SBmsy*Hmsy
   for (t in 1:N)
   {
   # use x y to put them towards the end of the alphabetically sorted  mcmc object
   #SP[t] <- pow(r.pella,-(m-1))*SB[t]*(1-pow(P[t],m-1))
   BtoBmsy[t] <- SB[t]/SBmsy
   HtoHmsy[t] <- H[t]/(Hmsy)

   #modified for territorial HCR:
   Overfishing_ind[t]<-ifelse(BtoBmsy[t]>0.7, H[t]/(Hmsy),  H[t]/((Hmsy*SB[t])/(0.7*SBmsy)))
   }


   # Enforce soft penalty on K if < K_bounds >
   K.pen ~ dnorm(penK,1000) # enforce penalty
   penK  <- ifelse(K<(K_bounds[1]),log(K)-log(K_bounds[1]),ifelse(K>K_bounds[2],log(K)-
log(K_bounds[2]),0)) # penalty if Pmean is outside viable biomass


   # Enforce soft penalty on process deviance if sigma.proc > 0.2
   proc.pen ~ dnorm(penProc,1000) # enforce penalty
   penProc  <- ifelse(sigma>sigmaproc_bound,log(sigma)-log(sigmaproc_bound),0)


   # Enforce soft penalty on observation error if sigma.obs > sigma_bound
   for(i in 1:nvar){
   obs.pen[i] ~ dnorm(penObs[i],1000) # enforce penalty
   penObs[i]  <- ifelse(pow(tau2[i],0.5)>sigmaobs_bound,log(pow(tau2[i],0.5))-
log(sigmaobs_bound),0)
   }
```

```
   ", append=TRUE)

# PROJECTION
if(Projection==TRUE){
  cat("
     for(i in 1:nTAC){
     # Project first year into the future
     prPmean[1,i] <- ifelse(P[N] > Plim,
     log(max(P[N] +  Hmsy/(1-1/m)*P[N]*(1-pow(P[N],m-1)) - TACint_17/K,0.005)),    #JS 2-
28-17  changed TAC_int to TACint_17
     log(max(P[N] +  Hmsy/(1-1/m)*P[N]*(1-pow(P[N],m-1))*4*P[N] - TACint_17/K,0.005)))
#JS 2-28-17 changed TAC_int to TACint_17
     prP[1,i] ~ dlnorm(prPmean[1,i],isigma2)
     # Project all following years
     for(t in 2:pyrs){
     prPmean[t,i] <- ifelse(prP[t-1,i] > Plim,
     log(max(prP[t-1,i] +  Hmsy/(1-1/m)*prP[t-1,i]*(1-pow(prP[t-1,i],m-1)) - TAC[t-
1,i]/K,0.001)),
     log(max(prP[t-1,i] +  Hmsy/(1-1/m)*prP[t-1,i]*(1-pow(prP[t-1,i],m-1))*slope.HS*prP[t-1,i] -
TAC[t-1,i]/K,0.005)))
     # process error (as monte-carlo simular)
     prP[t,i] ~ dlnorm(prPmean[t,i],isigma2)}
     for(t in 1:pyrs){
     prB[t,i] <- prP[t,i]*K
     prH[t,i] <- TAC[t,i]/prB[t,i]
     prHtoHmsy[t,i] <- prH[t,i]/Hmsy
     prBtoBmsy[t,i] <- prB[t,i]/SBmsy
     prOverfishing_ind[t,i]<-ifelse(prBtoBmsy[t,i]>0.7, prH[t,i]/(Hmsy),
prH[t,i]/((Hmsy*prB[t,i])/(0.7*SBmsy))) #JS 2-13-19
     }}
     ",append=TRUE)} else {
       cat("
          #Prevent error for unused input
          fakeTAC <-  TAC
          fakepyrs <- pyrs
          fakenTAC <- nTAC
          fakeTACint <- TACint
          prHtoHmsy <- 1
          prP <- 1
          prBtoBmsy <- 1
          prOverfishing_ind<-1
          prH<-1
          ", append=TRUE)}

cat("
```

```
} # END OF MODEL
   ",append=TRUE,fill = TRUE)
sink()


ptm <- proc.time()

mod <- jags(surplus.dat, inits,params,paste(JABBA), n.chains = nc, n.thin = nt, n.iter = ni,
n.burnin = nb)  # adapt is burn-in

proc.time() - ptm
save.time = proc.time() - ptm

cat(paste0("\n",paste0("><> Scenario ",Scenario,"_",Mod.names," completed in
",as.integer(save.time[3]/60)," min and ",round((save.time[3]/60-
as.integer(save.time[3]/60))*100)," sec <><","\n")))

cat(paste0("\n","><> Produce results output of ",Mod.names," model for ",assessment,"
",Scenario," <><","\n"))

# if run with library(rjags)
posteriors = mod$BUGSoutput$sims.list



#------------------------------------------------------------
# <><><><><><><><><><>< Outputs ><><><><><><><><><><><><><>
#------------------------------------------------------------
output.dir = paste0(File,"/",assessment,"/",Scenario,"_",Mod.names,"/Output")
dir.create(output.dir, showWarnings = FALSE)

# run some mcmc convergence tests
par.dat= data.frame(posteriors[params[c(1:7)]])

geweke1 = geweke.diag(data.frame(par.dat[1:(nsaved/nc),]))
geweke2 = geweke.diag(data.frame(par.dat[(nsaved/nc+1):nsaved,]))
pvalues1 <- 2*pnorm(-abs(geweke1$z))
pvalues2 <- 2*pnorm(-abs(geweke2$z))

heidle1 = heidel.diag(data.frame(par.dat[1:(nsaved/nc),]))
heidle2 = heidel.diag(data.frame(par.dat[(nsaved/nc+1):nsaved,]))

autocorr1=lapply(par.dat[1:(nsaved/nc),],FUN=acf,plot=F)
autocorr2=lapply(par.dat[(nsaved/nc+1):nsaved,],FUN=acf,plot=F)
```

```
max.acK=pmax(abs(autocorr1$K$acf),abs(autocorr1$K$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acr=pmax(abs(autocorr1$r$acf),abs(autocorr2$r$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acq=pmax(abs(autocorr1$q$acf),abs(autocorr2$q$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acpsi=pmax(abs(autocorr1$psi$acf),abs(autocorr2$psi$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acsigma2=pmax(abs(autocorr1$sigma2$acf),abs(autocorr2$sigma2$acf))[c(2,6)] #max lag1
and lag5 autocorrelation across chains
max.actau2=pmax(abs(autocorr1$tau2$acf),abs(autocorr2$tau2$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.acm=pmax(abs(autocorr1$m$acf),abs(autocorr2$m$acf))[c(2,6)] #max lag1 and lag5
autocorrelation across chains
max.ac=rbind(max.acK,max.acr,max.acq,max.acpsi,max.acsigma2,max.actau2,max.acm)

# postrior means + 95% BCIs
#Model  parameter
apply(par.dat,2,quantile,c(0.025,0.5,0.975))

man.dat = data.frame(posteriors[params[8:10]])
#Management quantities
apply(man.dat,2,quantile,c(0.025,0.5,0.975))

# Depletion
Depletion = posteriors$P[,c(1,n.years)]
colnames(Depletion) = c(paste0("P",years[1]),paste0("P",years[n.years]))

# Current stock status (Kobe posterior)
H_Hmsy.cur = posteriors$HtoHmsy[,c(n.years)]
B_Bmsy.cur = posteriors$BtoBmsy[,c(n.years)]


# Prepare posterior quantities
man.dat = data.frame(man.dat,Depletion,B_Bmsy.cur,H_Hmsy.cur)

results = round(t(cbind(apply(par.dat,2,quantile,c(0.025,0.5,0.975)))),6)

results = data.frame(Median =
results[,2],LCI=results[,1],UCI=results[,3],Geweke.p1=round(pvalues1,3),Geweke.p2=round(pv
alues2,3),Heidel.p1 = round(heidle1[,3],3),Heidel.p2 =
round(heidle2[,3],3),Heidel.it1=heidle1[,2],Heidel.it2=heidle2[,2],Heidel.hw.pass1=heidle1[,4]=
=1,Heidel.hw.pass2=heidle2[,4]==1,lag1=max.ac[,1],lag5=max.ac[,2]) #added all heidleberger
and welch diagnostics 10-25-2018

ref.points = round(t(cbind(apply(man.dat,2,quantile,c(0.025,0.5,0.975)))),3)
```

```
ref.points = data.frame(Median = ref.points[,2],LCI=ref.points[,1],UCI=ref.points[,3])

# get number of parameters
npar = length(par.dat)
# number of years
N=n.years

# Save posteriors (Produces large object!)
if(save.all==TRUE) save(posteriors,file=paste0(output.dir,"/",Scenario,"_posteriors"))

#------------------------------------------------------------------------
# Save parameters, results table and current status posterior in csv files
#------------------------------------------------------------------------

# Save model estimates and convergence p-values
write.csv(data.frame(results),paste0(output.dir,"/Estimates_",assessment,"_",Scenario,".csv"))

# Make standard results table with parameter estimates and reference points
Table = rbind(data.frame(results)[,1:3],data.frame(ref.points))
Table[4,] = round(sqrt((Table[4,])),3)
rownames(Table)[4] = "sigma.proc"
write.csv(Table,paste0(output.dir,"/Results_",assessment,"_",Scenario,".csv"))
#Save posterior of recent assessment year (KOBE posterior)
write.csv(data.frame(BtoBmsy=B_Bmsy.cur,FtoFmsy=H_Hmsy.cur),paste0(output.dir,"/Status_
posterior",assessment,".csv"))

## source all plotting scripts
source(paste0(JABBA.file,'/plot_JABBA_report_combined.R'))


if(save.trajectories==TRUE){
  cat(paste0("\n","><> Saving Posteriors of FRP trajectories <><","\n"))

  # FRP trajectories
  trajectories = array(NA,c(nsaved,n.years,3))
  trajectories[,,1] = posteriors$P
  trajectories[,,2] = posteriors$BtoBmsy
  trajectories[,,3] = posteriors$Overfishing_ind

  kb=kobeJabba(trajectories,years[1])
  save(kb,file=paste0(output.dir,"/",Scenario,"_trajectories"))

}
```

```
cat(paste0("\n","><> Scenario ",Mod.names,"_",Scenario," for ",assessment," - DONE!
<><","\n"))
```

1.4. Plotting files to generate output from JABBA model. Common to all territories

```
library(dplyr)
library(tseries)

## some preset variables
natM = 0.3 ## used in kobe

## PLOT TOTAL LANDINGS ----
cat(paste0("\n","-Plotting Total Landings","\n"))
Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)

png(file = paste0(output.dir,"/Landings_",assessment,"_",Scenario,".png"), width = 5, height =
3.5,
   res = 720, units = "in")
par(Par)

cord.x <- c(years,rev(years))
y<-rep(0,length(years))
plot(years,(TC),type="l",ylim=c(0,max(TC)),lty=1,lwd=1.3,xlab="Year",ylab=paste0("Catch
('000 t",catch.metric,")"),main="")
polygon(cord.x,c(TC,rev(y)),col="gray",border=1,lty=1)
dev.off()


## PLOT POSTERIORS ----
cat(paste0("\n","-Plotting Posteriors","\n"))
sel.par = c(1,2,7,4,3,5,6)

out=data.frame(posteriors[params[sel.par]])
if(nSel>1) out=out[,-c(3:(3+nSel-2))]

node_id = names(out)

#informative priors
Prs = as.matrix(cbind(K.pr,r.pr,c(0,0),psi.pr))

#Posteriors
Par = list(mfrow=c(round(length(node_id)/3+0.33,0),3),mai=c(0.6,0.1,0,.1),omi =
c(0.3,0.5,0.1,0) + 0.1,mgp=c(1,0.1,0), tck = -0.02,cex=0.8, cex.lab=1) #mai was(0.4,0.1,0,.1)
png(file = paste0(output.dir,"/Fig9_Posteriors_",assessment,"_",Scenario,".png"),width  = 8,
height = 2.5*round(length(node_id)/3,0),
   res = 720, units = "in")
par(Par)
```

```
node_id = names(out)



#par(mfrow=c(4,2),oma=c(0,1,1,0), mar=c(4,4,1,1))

for(i in 1:length(node_id))
{

  post.par = as.numeric(unlist(out[paste(node_id[i])]))

  if(i==1){

    rpr = rlnorm(10000,log(K.pr[1]),K.pr[2])
    pdf = stats::density(post.par,adjust=2)
    prior = dlnorm(sort(rpr),log(K.pr[1]),K.pr[2])

plot(pdf,type="l",ylim=range(prior,pdf$y),xlim=range(c(pdf$x,quantile(rpr,c(0.0001,0.95)))),yax
t="n",xlab=" ",ylab="",xaxs="i",yaxs="i",main="")
    title(xlab = "K", cex.lab = 1.1,line = 1.75)
    polygon(c(sort(rpr),rev(sort(rpr))),c(prior,rep(0,length(sort(rpr)))),col=gray(0.4,1))
    polygon(c(pdf$x,rev(pdf$x)),c(pdf$y,rep(0,length(pdf$y))),col=gray(0.7,0.7))
    legend('topright',c("Prior","Posterior"),pch=22,pt.cex=1.5,pt.bg =
c(grey(0.4,1),grey(0.8,0.6)),bty="n")
  }


  if(i==2){

    rpr = rlnorm(10000,log(Prs[1,i]),Prs[2,i])
    pdf = stats::density(post.par,adjust=2)
    prior = dlnorm(sort(rpr),log(Prs[1,i]),Prs[2,i])

plot(pdf$x,pdf$y,type="l",ylim=range(prior,pdf$y),xlim=range(c(post.par,quantile(rpr,c(0.0001,
0.95)))),yaxt="n",xlab="",ylab="",xaxs="i",yaxs="i")
    title(xlab = paste(node_id[i]), cex.lab = 1.1,line = 1.75)
    polygon(c(sort(rpr),rev(sort(rpr))),c(prior,rep(0,length(sort(rpr)))),col=gray(0.4,1))
    polygon(c(pdf$x,rev(pdf$x)),c(pdf$y,rep(0,length(pdf$y))),col=gray(0.7,0.7))
  }

  if(i==3){
    if(Model<4){
      plot(1,1,type="n",xlim=range(0.5,2.5),yaxt="n",xlab=" ",ylab="",xaxs="i",yaxs="i")
      abline(v=m,lwd=2)
      title(xlab = paste(node_id[i]), cex.lab = 1.1,line = 1.75)
    }
```

```
    if(Model==4){
      mpr = rlnorm(10000,log(m),shape.CV)
      pdf = stats::density(post.par,adjust=2)
      prior = dlnorm(sort(mpr),log(m),shape.CV)

plot(pdf$x,pdf$y,type="l",ylim=range(prior,pdf$y),xlim=range(c(post.par,quantile(rpr,c(0.0001,
0.95)))),yaxt="n",xlab="",ylab="",xaxs="i",yaxs="i")
      title(xlab = paste(node_id[i]), cex.lab = 1.1,line = 1.75)
      polygon(c(sort(mpr),rev(sort(mpr))),c(prior,rep(0,length(sort(rpr)))),col=gray(0.4,1))
      polygon(c(pdf$x,rev(pdf$x)),c(pdf$y,rep(0,length(pdf$y))),col=gray(0.7,0.7))
      abline(v=1, col="white")  #No solution at m=1

    }
  }


  if(i==4){
    if(psi.dist=="beta"){
      parm = fitdist(post.par[post.par<1 & post.par>0.01], "beta")$estimate
      rpr = rbeta(10000,(psi.pr[1]),psi.pr[2])
      pdf = stats::density(post.par,adjust=2)
      prior = dbeta(sort(rpr),psi.pr[1],psi.pr[2])
    } else {
      rpr = rlnorm(10000,log(psi.prior[1]),psi.prior[2])
      pdf = stats::density(post.par,adjust=2)
      prior = dlnorm(sort(rpr),log(psi.prior[1]),psi.prior[2])}

plot(pdf,type="l",ylim=range(quantile(c(prior,pdf$y,c(0,0.95)))),xlim=range(c(0.5,post.par,pdf$
x,quantile(rpr,c(0.001,0.999)))),yaxt="n",xlab=" ",ylab="",xaxs="i",yaxs="i",main="")
    title(xlab=expression(psi), cex.lab = 1.1,line = 1.75)
    polygon(c(sort(rpr),rev(sort(rpr))),c(prior,rep(0,length(sort(rpr)))),col=gray(0.4,1))
    polygon(c(pdf$x,rev(pdf$x)),c(pdf$y,rep(0,length(pdf$y))),col=gray(0.7,0.7))
    #legend('topright',c("Prior","Posterior"),pch=22,pt.cex=1.5,pt.bg =
c(grey(0.4,1),grey(0.8,0.6)),bty="n")
  }

  if(i>4){

    if(sigma.proc!=TRUE & i==length(node_id)) {
      if(node_id[i]=="sigma2"){new_xlab<-(expression(paste(sigma^2)))}else{
        new_xlab<-paste(node_id[i])
      }
      if(node_id[i]=="tau2"){new_xlab<-(expression(paste(tau^2)))}else{
        new_xlab<-paste(node_id[i])
      }
```

```
    #
plot(1,1,type="n",xlim=range(0,0.15^2),yaxt="n",xlab=paste(node_id[i]),ylab="",xaxs="i",yaxs
="i")
    plot(1,1,type="n",xlim=range(0,0.15^2),yaxt="n",xlab=" ", ylab="",xaxs="i",yaxs="i")
    title(xlab=new_xlab, cex.lab = 1.1,line = 1.75)
    abline(v=sigma.proc^2,lwd=2)} else {

      if(node_id[i]=="sigma2"){
        new_xlab<-expression(paste(sigma[eta]^2))
      }else{
        new_xlab<-paste(node_id[i])}
      #added because the above code wont work

      if(node_id[i]=="tau2"){new_xlab<-expression(paste(sigma[tau][estimated]^2))}

      pdf = stats::density(post.par,adjust=2)
      #
plot(pdf,type="l",xlim=range(0,post.par),yaxt="n",xlab=paste(node_id[i]),ylab="",xaxs="i",yaxs
="i",main="")
      plot(pdf,type="l",xlim=range(0,post.par),yaxt="n",xlab="
",ylab="",xaxs="i",yaxs="i",main="")
      title(xlab=new_xlab, cex.lab = 1.1,line = 1.75)
      if(i==length(node_id)& igamma[1]>0.9){
        rpr = 1/rgamma(10000,igamma[1],igamma[2])
        prior = stats::density(rpr,adjust=2)
        polygon(c(prior$x,rev(prior$x)),c(prior$y,rep(0,length(prior$y))),col=gray(0.4,1))
      }

      polygon(c(pdf$x,rev(pdf$x)),c(pdf$y,rep(0,length(pdf$y))),col=gray(0.7,0.7))
      #legend('topright',c("Posterior"),pch=22,pt.cex=1.5,pt.bg = c(grey(0.8,0.6)),bty="n")
    } }

}
mtext(paste("Density"), side=2, outer=TRUE, at=0.5,line=1,cex=0.9)
dev.off()


## PLOT MCMC CHAINS ----
cat(paste0("\n","-Plotting MCMC Chains","\n"))
Par = list(mfrow=c(round(length(node_id)/3+0.33,0),3),mai=c(0.4,0.1,0,.1),omi =
c(0.3,0.5,0.1,0) + 0.1,mgp=c(1,0.1,0), tck = -0.02,cex=0.8)
png(file = paste0(output.dir,"/MCMC_",assessment,"_",Scenario,".png"), width = 8, height =
2.5*round(length(node_id)/3,0),
   res = 720, units = "in")
par(Par)
for(i in 1:length(node_id)){
```

```
  post.par = as.numeric(unlist(out[paste(node_id[i])]))
  plot(out[1:(nsaved/nc),i],xlab=paste(node_id[i]),ylab="",type="l",col=2) #js added 4 lines
below
  lines(out[(nsaved/nc+1):nsaved,i],xlab=paste(node_id[i]),ylab="",type="l",col=4)
  lines(rep(mean(out[1:(nsaved/nc),i]),length(out[,i])),col=6,lwd=2,lty=1)
  lines(rep(mean(out[(nsaved/nc+1):nsaved,i]),length(out[,i])),col=5,lwd=2,lty=2)
}
dev.off()

## MK PLOT CPUE FITS ----
cat(paste0("\n","-Plotting CPUE Fits","\n"))

# extract predicted CPUE + CIs  #JS removed normalization 2-7-19
N = n.years
series <- 1:n.indices
check.yrs = apply(CPUE,1,sum,na.rm=TRUE)
cpue.yrs = years[check.yrs>0]

Par = list(
  mfrow = c(1,1),
  mai = c(0.35, 0.15, 0, .15),
  omi = c(0.3, 0.25, 0.2, 0) + 0.1,
  mgp = c(2, 0.5, 0),
  tck = -0.02,
  cex = 0.8
)
png(
  file = paste0(output.dir, "/Fig7_Fits_", assessment, "_", Scenario, ".png"),
  width = 6,
  height = 4,
  res = 720,
  units = "in"
)
par(Par)
for(i in 1:n.indices){

  # set observed vs predicted CPUE
  #par(mfrow=c(1,1))
  Yr = years
  Yr = min(Yr):max(Yr)
  yr = Yr-min(years)+1

  fit = apply(posteriors$CPUE[,,i],2,quantile,c(0.025,0.5,0.975))
  mufit = mean(fit[2,])
  fit = fit #/mufit
```

```
  cpue.i = CPUE[is.na(CPUE[,i])==F,i]
  yr.i = Yr[is.na(CPUE[,i])==F]
  se.i = sqrt(se2[is.na(CPUE[,i])==F,(i)])

 # ylim = c(min(fit*0.9,exp(log(cpue.i)-1.96*se.i)/mufit),
max(fit*1.05,exp(log(cpue.i)+1.96*se.i)/mufit))
  ylim = c(min(fit*0.9,exp(log(cpue.i)-1.96*se.i)), max(fit*1.05,exp(log(cpue.i)+1.96*se.i)))
  cord.x <- c(Yr,rev(Yr))
  cord.y <- c(fit[1,yr],rev(fit[3,yr]))

  # Plot Observed vs predicted CPUE
  #
plot(years,CPUE[,i],ylab="",xlab="",ylim=ylim,xlim=range(years),type='n',xaxt="n",yaxt="n")
  plot(years,CPUE[,i],ylab="",xlab="",ylim=ylim,xlim=range(yr.i),type='n',xaxt="n",yaxt="n")
  axis(1,labels=TRUE,cex=0.8)
  axis(2,labels=TRUE,cex=0.8)
  polygon(cord.x,cord.y,col=grey(0.5,0.5),border=0,lty=2)


  lines(Yr,fit[2,yr],lwd=2,col=1)
  if (SE.I == TRUE |
      max(se2) > 0.01) {
    plotCI(
      yr.i,
  #    cpue.i / mufit,
     cpue.i,
  #    ui = exp(log(cpue.i) + 1.96 * se.i) / mufit,
  #    li = exp(log(cpue.i) - 1.96 * se.i) / mufit,
      ui = exp(log(cpue.i) + 1.96 * se.i),
      li = exp(log(cpue.i) - 1.96 * se.i),

      add = T,
      gap = 0,
      pch = 21,
      xaxt = "n",
      yaxt = "n"
    )
  } else{
    points(
      yr.i,
     # cpue.i / mufit,
      cpue.i,
      pch = 21,
      xaxt = "n",
      yaxt = "n",
      bg = "white"
```

```
   )
  }

#  legend('bottomleft',paste(indices[i]),bty="n",y.intersp = -0.2,cex=0.9)  #only 1 series 2-7-19
  legend('top',
    # c('topright','topleft')[ifelse(i %% 2 == 0,2,1)],
    legend = c(
      "Observed CPUE with 95% C.I.",
      "Model Estimated CPUE",
      "Model Estimated CPUE 95% C.I."
    ),
    fill = c('white', NA, "grey"),
    lty = c(1, 1, NA),
    lwd = c(0.8, 2.5, 0) ,
    col = c("black", "black", "black"),
    pch = c(21, NA, NA),
    bg = c('white', NA, NA),
    border = c(NA, NA, "grey"),
    bty = "n",
    cex = 0.75
  )
}
mtext(paste("Year"), side=1, outer=TRUE, at=0.5,line=1,cex=1)
mtext(paste("CPUE (lb/line hr)"), side=2, outer=TRUE, at=0.5,line=1,cex=1)
dev.off()
## end CPUE Fits

##JS plot pairs and correlations for posteriorz#########################################
cat(paste0("\n","-Plotting pairs and correlations ","\n"))

#grab prameters of interest
newdat<-as.data.frame(cbind(posteriors$K, posteriors$r, posteriors$psi, posteriors$m,
posteriors$q, posteriors$tau2, posteriors$sigma2))

correlations<-cor(newdat)
colnames(correlations)<-c("K","R", "psi", "m", "q", "tau2", "sigma2")
rownames(correlations)<-c("K","R", "psi", "m", "q", "tau2", "sigma2")

panel.cor <- function(x, y, digits = 3, prefix = "")
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- round(cor(x, y),3)
  #txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- format(ifelse(abs(r)>0.001,r,"<0.001"),digits=digits)
  txt <- paste0(prefix, txt)
```

```
  text(0.5, 0.5, txt, cex=1.5)
}

#create pairs plot with correlations on lower panel
png(file = paste0(output.dir,"/Corr_",assessment,"_",Scenario,".png"), width = 6, height = 4,
    res = 800, units = "in")
par(Par)


pairs(newdat, labels=c("K","R", expression(psi), "m",
"q",expression(paste(sigma[tau][estimated]^2))
                  ,expression(paste(sigma[eta]^2))),lower.panel=panel.cor)
dev.off()
##############################################



## MK PLOT JABBA RESIDUALS, log ----
cat(paste0("\n","-Plotting CPUE Residuals","\n"))
## Generate & Save Log Residuals
Resids = NULL
for (i in 1:n.indices) {
  Resids = rbind(Resids, log(CPUE[, i]) - log(apply(posteriors$CPUE[, , i], 2, quantile, c(0.5))))
}
Res.CPUE = data.frame(Resids)
row.names(Res.CPUE) = indices
colnames(Res.CPUE) = paste(Yr)
write.csv(Res.CPUE,paste0(output.dir,"/ResCPUE_",assessment,"_",Scenario,".csv"))

Par = list(mfrow=c(n.indices,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
png(file = paste0(output.dir,"/LogResiduals_",assessment,"_",Scenario,".png"), width = 6, height
= 4,
    res = 1080, units = "in")
par(Par)
for(i in 1:n.indices){

  tempResid <- data.frame(na.omit(cbind(Yr,t(Resids)[,i])))

  plot(
    Yr,
    Yr,
    type = "n",
    ylim = c(min(-1, -1.2 * max(
      abs(Resids), na.rm = T
    )), max(1, 1.2 * max(
      abs(Resids), na.rm = T
```

```
    ))),
    # xlim = c(1958,2005),
    xlim = c(min(tempResid$Yr), max(tempResid$Yr)),
    ylab = "Log Residuals",
    xlab = "Year"
  )

  abline(h = 0, lty = 2)
  positions = runif(series, -0.2, 0.2)

  for (t in 1:length(tempResid$Yr)) {
    lines(rep((tempResid$Yr + positions[i])[t], 2), c(0, tempResid[t,'V2']),
          col = as.character(wink.colors[wink.colors$idx == indices[i], 'cols']))
  }
  points(
    tempResid$Yr + positions[i],
    tempResid$V2,
    # Yr + positions[i],
    # Resids[i, ],
    col = NA,
    pch = 21,
    bg = as.character(wink.colors[wink.colors$idx == indices[i], 'cols']))
}

dev.off()

## MK PLOT JABBA RESIDUALS, standardized ----
## Generate & Save Std Residuals
StResid = NULL
for(i in 1:n.indices){
  StResid =rbind(StResid,log(CPUE[,i]/apply(posteriors$CPUE[,,i],2,quantile,c(0.5)))/

apply(posteriors$TOE[,,i],2,quantile,c(0.5))+0.5*apply(posteriors$TOE[,,i],2,quantile,c(0.5)))
}

DIC =round(mod$BUGSoutput$DIC,1)
Nobs =length(as.numeric(Resids)[is.na(as.numeric(Resids))==FALSE])
DF = Nobs-npar

DIC =round(mod$BUGSoutput$DIC,1)
SDNR = round(sqrt(sum(StResid^2,na.rm =TRUE)/(Nobs-1)),2)
Crit.value = (qchisq(.95, df=(Nobs-1))/(Nobs-1))^0.5
StRes.CPUE = data.frame(StResid)
row.names(Res.CPUE) = indices
colnames(Res.CPUE) = paste(Yr)
write.csv(Res.CPUE,paste0(output.dir,"/StResCPUE_",assessment,"_",Scenario,".csv"))
```

```
## do plot, standardized

Par = list(mfrow=c(n.indices,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
png(file = paste0(output.dir,"/Fig8_StdResiduals_",assessment,"_",Scenario,".png"), width = 6,
height = 4,
   res = 1080, units = "in")
par(Par)
for (i in 1:n.indices) {
  tempResid <- data.frame(na.omit(cbind(Yr, t(StResid)[, i])))
  plot(
    Yr,
    Yr,
    type = "n",
    ylim = c(min(-1,-1.2 * max(
      abs(StResid), na.rm = T
    )), max(1, 1.2 * max(
      abs(StResid), na.rm = T
    ))),
    xlim = c(min(tempResid$Yr), max(tempResid$Yr)),
    ylab = "Standardized Residuals",
    xlab = "Year"
  )
  abline(h = 0, lty = 2)
  positions = runif(n.indices,-0.2, 0.2)

  for (t in 1:length(tempResid$Yr)) {
    lines(rep((tempResid$Yr + positions[i])[t], 2), c(0, tempResid[t, 'V2']),
          col = as.character(wink.colors[wink.colors$idx == indices[i], 'cols']))
  }
  points(
    tempResid$Yr + positions[i],
    tempResid$V2,
    col = NA,
    pch = 21,
    bg = as.character(wink.colors[wink.colors$idx == indices[i], 'cols'])
  )
}

dev.off()




## Produce Goodness of Fit stats
# get degree of freedom
```

```
DIC =round(mod$BUGSoutput$DIC,1)
Nobs =length(as.numeric(Resids)[is.na(as.numeric(Resids))==FALSE])
DF = Nobs-npar
RMSE = round(100*sqrt(sum(Resids^2,na.rm =TRUE)/DF),1)

#JS additions for resid
diagnostics#############################################################################
##


#calculate residuals
#median cpue
#CPUE.pred<-apply(posteriors$CPUE,2,median)
#resids<-CPUE-CPUE.pred #-log resids already done on 287

#SHAPIRO-WILK Normality test, pval>0.05 indicates normality
shap<-shapiro.test(Resids)

#Runs test
#convert resids to binary factor
resid.bin<-ifelse(Resids>0,1,0)
rt<-runs.test(as.factor(resid.bin)) #pval<0.05 indicates nonrandomness

#linear regression of residuals through time
fit1 <- lm(t(Resids) ~ years)
lm_sum<-(summary(fit1))


GOF = data.frame(Stastistic = c("N","p","DF","SDNR","RMSE","DIC", "Shapiro-p", "Runs-p",
"Temp_resid-p"),Value =
c(Nobs,npar,DF,SDNR,RMSE,DIC,shap[[2]],rt[[3]],lm_sum[[4]][2,4]))


write.csv(GOF,paste0(output.dir,"/GOF_",assessment,"_",Scenario,".csv"))

## PLOT Proc Devs ----
cat(paste0("\n","-Plotting Process Deviation","\n"))
proc.dev = apply(posteriors$Proc.Dev,2,quantile,c(0.025,0.5,0.975))

Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)

png(file = paste0(output.dir,"/ProcDev_",assessment,"_",Scenario,".png"), width = 6, height = 4,
   res = 800, units = "in")
par(Par)

ylim = range(proc.dev)*1.1
```

```
cord.x <- c(years,rev(years))
cord.y <- c(proc.dev[1,],rev(proc.dev[3,]))
# Process Error
plot(years,proc.dev[2,],ylab="Process deviation P[t]",xlab="Year",ylim=ylim,type="n")
polygon(cord.x,cord.y,col='grey',border=0,lty=2)
lines(years,proc.dev[2,],lwd=2)
lines(years,rep(0,length(years)),lty=5)

dev.off()

## MK Plot Stacked Tau ----
cat(paste0("\n","-Plotting Stacked Obs Error","\n"))
styr.index = apply(se, 2, function(x) min(which(x>0)) )[2:(1+n.indices)] #added by MD Fitchett
for monitoring obs error by series, starting indexed
endyr.index = apply(se, 2, function(x) max(which(x>0)) )[2:(1+n.indices)] #added by MD
Fitchett for monitoring obs error by series, ending indexed

png(file = paste0(
  output.dir,
  "/Fig10_TOEstack.png"),
  width = 6,
  height = 4,
  res = 720,
  units = "in"
)
Par = list(
  mfrow = c(n.indices, 1),
  mar = c(3.5, 3.5, 0.1, 0.1),
  mgp = c(2., 0.5, 0),
  tck = -0.02,
  cex.axis = 1.1,
  cex = 1.1
)
par(Par)
for(i in 1:n.indices){
  plot(1,
       type="n", ylab= "Observation Error Variance", xlab="Year",
       xlim = c(years[styr.index[i]],years[endyr.index[i]]),
       ylim=c(0,max(c(max(apply(posteriors$TOE,2,mean)*1.27))^2,0.08)))
  if(sigma.est == T){
    polygon(
      x=c(years[styr.index[i]], years[styr.index[i]:endyr.index[i]],years[endyr.index[i]]),
      y=c(0, (apply(posteriors$TOE,2,mean)[styr.index[i]:endyr.index[i]])^2,0),
      col='darkgrey'
    )
```

```
   legend("topleft",c("Estimable Observation Error Variance","Observation Error Variance From
CPUE CV"), #"Fixed Minimal Observation Error Variance"), #JS removed fixed obs 2-7-19
          col=c('darkgrey', 'lightgrey'),bg='white', pt.lwd=5,
          cex=1.1,pt.cex=c(0,0), fill=c('darkgrey', 'lightgrey'),density=c(100,100),
          angle=c(0,0),border = c(NA,NA))
  }

  ## add se2 (input cpue se ^2 + fixed obs e ^2)
  polygon(
    x=c(years[styr.index[i]], years[styr.index[i]:endyr.index[i]],years[endyr.index[i]]),
    y=c(0, se2[styr.index[i]:endyr.index[i],i],0),
    col='light grey'
  )

  ## add fixed obs error [white]
  polygon(
    x=c(years[styr.index[i]],years[styr.index[i]] , years[endyr.index[i]],years[endyr.index[i]]),
    y=c(0,fixed.obsE^2,fixed.obsE^2,0),
    col='white'
  )

  ## add fixed obs error [blue shading]  #JS removed 2-7-19
 # polygon(
 #   x=c(years[styr.index[i]],years[styr.index[i]] , years[endyr.index[i]],years[endyr.index[i]]),
 #   y=c(0,fixed.obsE^2,fixed.obsE^2,0),
  #  col='navyblue',angle=135, lwd=5, density=12,fillOddEven = F
  #)
  if(sigma.est == F){
    legend("topleft",c("Observation Error Variance From CPUE CV", "Fixed Minimal
Observation Error Variance"),
          col=c('lightgrey', 'navyblue'),bg='white', pt.lwd=5,
          cex=0.9,pt.cex=c(0,5), fill=c('lightgrey', 'navyblue'),density=c(100,18),
          angle=c(0,135),border = c(NA,"navyblue"))
  }
}
dev.off()



## JABBA Management Plots ----
cat(paste0("\n","-Producing Management Plots","\n"))
## PLOT BIOMASS
B_t = posteriors$SB
mu.B = apply(B_t,2,quantile,c(0.025,0.5,0.975))
```

```
Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.5, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
png(file = paste0(output.dir,"/Biomass_",assessment,"_",Scenario,".png"), width = 5, height =
3.5,
   res = 720, units = "in")
par(Par)
ylim = c(0, max(mu.B ))
cord.x <- c(years,rev(years))
cord.y <- c(mu.B [1,],rev(mu.B [3,]))

# B_t
plot(years,mu.B[2,],ylab="Biomass (1000 lb)",xlab="Year",ylim=ylim,type="n")
polygon(cord.x,cord.y,col='grey',border=0,lty=2)
lines(years,mu.B[2,],lwd=2,col='blue')
lines(years,rep(median(posteriors$SBmsy),length(years)),lty=5) #js changed from mean 10-25-
18
text((max(years)-
min(years))/30+years[1],median(posteriors$SBmsy)*0.9,expression(paste(B[MSY])))
dev.off()


## PLOT H/HMSY and B/BMSY
HtoHmsy = posteriors$HtoHmsy
HtoHCR = posteriors$Overfishing_ind #JS added HCR
BtoBmsy = posteriors$BtoBmsy

mu.f = apply(HtoHmsy,2,quantile,c(0.025,0.5,0.975))
mu.b = apply(BtoBmsy,2,quantile,c(0.025,0.5,0.975))
mu.h = apply(HtoHCR,2,quantile,c(0.025,0.5,0.975))

f = HtoHmsy[,N]
b = BtoBmsy[,N]

Par = list(mfrow=c(2,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
png(file = paste0(output.dir,"/Fig11_TrendMSY_",assessment,"_",Scenario,".png"), width = 5,
height = 7,
   res = 720, units = "in")
par(Par)
## H/HCR #JS changed
ylim = c(0, max(mu.h))
cord.x <- c(years,rev(years))
cord.y <- c(mu.h[1,],rev(mu.h[3,]))


#plot(years,mu.f[2,],ylab=ifelse(harvest.label=="Hmsy",expression(paste(H/H[MSY])),expressio
n(paste(H/H[MSY]))),xlab="Year",ylim=ylim,type="n") #JS changed for HCR
```

```
plot(years,mu.h[2,],ylab=expression(H/H[CR]),xlab="Year",ylim=ylim,type="n")  #JS changed
for HCR
polygon(cord.x,cord.y,col='grey',border=0,lty=2)
lines(years,mu.h[2,],lwd=2,col=4)
lines(years,rep(1,length(years)),lty=5)
legend("topright", legend=c("95% C.I."), fill=c("grey"),
       col=c("black"),
       border=c("grey"), bty="n", cex=0.8)

## B/Bmsy
ylim = c(0, max(mu.b,1.1))
cord.x <- c(years,rev(years))
cord.y <- c(mu.b[1,],rev(mu.b[3,]))


plot(years,mu.b[2,],ylab=expression(paste(B/B[MSY])),xlab="Year",ylim=ylim,type="n")
polygon(cord.x,cord.y,col='grey',border=0,lty=2)
lines(years,mu.b[2,],lwd=2,col=4)
lines(years,rep((1-natM),length(years)),lty=5)
legend("topright", legend=c("95% C.I."), fill=c("grey"),
       col=c("black"),
       border=c("grey"), bty="n", cex=0.8)

dev.off()

## 4-panel plot for B, H, H/HCR, B/BMSY

#All 4 plots############################
Par = list(
  mfrow = c(2, 2),
  mar = c(3.5, 3.5, 0.5, 0.1),
  mgp = c(2., 0.5, 0),
  tck = -0.02,
  cex = 0.9
)
png(
  file = paste0(output.dir,
    "/four-panel.png"
  ),
  width = 9,
  height = 6.5,
  res = 800,
  units = "in"
)

par(Par)
```

```
#biomass
ylim = c(0, max(mu.B ))
cord.x <- c(years,rev(years))
cord.y <- c(mu.B [1,],rev(mu.B [3,]))

# B_t
plot(years,mu.B[2,],ylab="Biomass (1000 lb)",xlab="Year",ylim=ylim,type="n")
polygon(cord.x,cord.y,col='grey',border=0,lty=2)
lines(years,mu.B[2,],lwd=2,col='blue')
lines(years,rep(median(posteriors$SBmsy),length(years)),lty=5) #js changed from mean 10-25-
18
text((max(years)-
min(years))/30+years[1],median(posteriors$SBmsy)*0.9,expression(paste(B[MSY])))
lines(years,rep(median((posteriors$SBmsy)*0.7),length(years)),lty=1) #js changed from mean
10-25-18
text((max(years)-
min(years))/30+years[1],median(posteriors$SBmsy)*0.6,expression(paste(0.7*B[MSY])))


#Harvest rate
H_t = posteriors$H
mu.H = apply(H_t,2,quantile,c(0.025,0.5,0.975))
ylim = c(0, max(mu.H ))
cord.x <- c(years,rev(years))
cord.y <- c(mu.H [1,],rev(mu.H [3,]))


plot(years,mu.H[2,],ylab=paste0("Harvest Rate"),xlab="Year",ylim=ylim,type="n")
polygon(cord.x,cord.y,col='grey',border=0,lty=2)
lines(years,mu.H[2,],lwd=2,col='blue')
lines(years,rep(median(posteriors$Hmsy),length(years)),lty=5) #js changed from mean 10-25-18
text((max(years)-
min(years))/30+years[1],median(posteriors$Hmsy)*0.9,expression(paste(H[MSY])))


#B.BMSY
ylim = c(0, max(mu.b,1.1))
cord.x <- c(years,rev(years))
cord.y <- c(mu.b[1,],rev(mu.b[3,]))


plot(years,mu.b[2,],ylab=expression(paste(B/B[MSY])),xlab="Year",ylim=ylim,type="n")
polygon(cord.x,cord.y,col='grey',border=0,lty=2)
lines(years,mu.b[2,],lwd=2,col=4)
lines(years,rep((1-natM),length(years)),lty=1)
```

```
text((max(years)-min(years))/30+years[1],0.6,expression(paste(0.7*B[MSY])))
lines(years,rep(1,length(years)),lty=5) #js changed from mean 10-25-18

#H.HCR
ylim = c(0, max(mu.h))
cord.x <- c(years,rev(years))
cord.y <- c(mu.h[1,],rev(mu.h[3,]))


#plot(years,mu.f[2,],ylab=ifelse(harvest.label=="Hmsy",expression(paste(H/H[MSY])),expressio
n(paste(H/H[MSY]))),xlab="Year",ylim=ylim,type="n") #JS changed for HCR
plot(years,mu.h[2,],ylab=expression(H/H[CR]),xlab="Year",ylim=ylim,type="n")  #JS changed
for HCR
polygon(cord.x,cord.y,col='grey',border=0,lty=2)
lines(years,mu.h[2,],lwd=2,col=4)
lines(years,rep(1,length(years)),lty=1)



dev.off()


## JABBA SP PHASE PLOTS ----
if(SP.plot!="phase"){
  Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
  png(file = paste0(output.dir,"/SP_",assessment,"_",Scenario,".png"), width = 6, height = 6,
      res = 800, units = "in")
  par(Par)
  Bit = seq(1,mean(posteriors$K),mean(posteriors$K)/500)
  SP = mean(posteriors$r)/(m-1)*Bit*(1-(Bit/median(posteriors$K))^(m-1))
  B = apply(posteriors$SB,2,mean)
  MSY = quantile(posteriors$MSY,c(0.025,0.5,0.975))
  plot(Bit,SP,type =
"n",ylim=c(0,max(c(max(Catch,na.rm=T),max(MSY*1.1)))),ylab=paste0("Surplus Production
",catch.metric),xlab="Biomass (t)")
  polygon(c(-10000,10^7,10^7,-10000),c(rep(MSY[1],2),rep(MSY[3],2)),border =
0,col=grey(0.5,0.4))
  lines(Bit,SP,col=2,lwd=2)
  lines(B,Catch,lty=1)
  points(B,Catch,cex=0.5,pch=4)
  sel.yr = c(1,round(quantile(1:N,0.7),0),N)
  points(B[sel.yr],Catch[sel.yr],col= 1,pch=c(22,21,24),bg="white",cex=1.5)
  abline(h=max(SP),col=4)
  sel.years = c(min(years),years[sel.yr[2]],max(years))
  lines(rep(median(posteriors$SBmsy),2),c(-1000,max(SP)),lty=2,col=2)
  legend('topright',
```

```
        c(expression(B[MSY]),"MSY","Catch",paste(sel.years)),
        lty=c(2,1,1,1,1,1),pch=c(-1,-1,4,22,21,24),pt.bg=c(0,0,0,rep("white",3)),
        col=c(2,4,rep(1,4)),lwd=1,cex=0.9,pt.cex=c(-1,-1,0.5,rep(1.3,3)),bty="n")

  dev.off()
}

if(SP.plot=="phase"){
  #----------------------------------------
  # Produce JABBA SP-phase plot
  #----------------------------------------
  Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
  png(file = paste0(output.dir,"/SPphase_",assessment,"_",Scenario,".png"), width = 6, height =
6,
      res = 800, units = "in")
  par(Par)
  Bit = seq(1,median(posteriors$K),median(posteriors$K)/500)
  Cmsy = Bit*median(posteriors$Hmsy)
  SP = median(posteriors$r)/(m-1)*Bit*(1-(Bit/median(posteriors$K))^(m-1))
  B = apply(posteriors$SB,2,mean)
  MSY = quantile(posteriors$MSY,c(0.025,0.5,0.975))
  Bmsy.sp = median(posteriors$SBmsy)
  K.sp = median(posteriors$K)
  green.x = c(max(Bit,B),max(Bit,B),Bmsy.sp,Bmsy.sp,max(Bit))
  green.y = c(Bmsy.sp,0,0,max(SP),max(Cmsy))
  red.x = c(0,0,Bmsy.sp,Bmsy.sp,0)
  red.y = c(K.sp,0,max(SP),K.sp,K.sp)
  plot(Bit,SP,type =
"n",ylim=c(0,max(c(max(Catch,na.rm=T)*1.05,max(MSY*1.1)))),xlim=c(0,max(Bit,B)),ylab=p
aste0("Surplus Production ",catch.metric),xlab="Biomass (t)",xaxs="i",yaxs="i")
  rect(0,0,K.sp*1.1,K.sp*1.1,col="green",border=0)
  rect(0,0,K.sp,K.sp,col="yellow",border=0)
  if(KOBE.type!="ICCAT") rect(0,max(SP),K.sp,K.sp,col="orange",border=0)
  polygon(green.x,green.y,border = 0,col="green")
  polygon(red.x,red.y,border = 0,col="red")

  ry.sp = Bit[Bit<=Bmsy.sp]
  for(i in 1:length(ry.sp)){

    lines(rep(Bit[i],2),c(Cmsy[i],SP[i]),col=ifelse(i %% 2== 0,"yellow","red"),lty=3)
    #i = i+1
  }

  gy.sp = Bit[Bit>Bmsy.sp]
  for(i in (length(ry.sp)+1):length(Bit)){
```

```
   #lines(rep(Bit[i],2),c(max(SP),Cmsy[i]),col=ifelse(i %% 2==
0,ifelse(KOBE.type=="ICCAT","yellow","orange"),"green"),lty=3)
   #i = i+1
  }


  polygon(c(-10000,10^7,10^7,-10000),c(rep(MSY[1],2),rep(MSY[3],2)),border =
FALSE,col=rgb(0,0,1,0.4))
  lines(Bit,SP,col=4,lwd=2)
  lines(B,Catch,lty=1,lwd=1)
  points(B,Catch,cex=0.8,pch=16)
  lines(Bit,Cmsy,col=1,lwd=1,lty=2)
  N=n.years
  sel.yr = c(1,round(quantile(1:N,0.7),0),N)
  points(B[sel.yr],Catch[sel.yr],col= 1,pch=c(22,21,24),bg="white",cex=1.7)
  abline(h=max(SP),col=4,lty=5)
  sel.years =years[sel.yr]
  lines(rep(median(posteriors$SBmsy),2),c(-1000,max(SP)),lty=2,col=4)

  legend('topright',
        c(expression(B[MSY]),"MSY","SP","Catch",paste(sel.years)),
        lty=c(2,5,1,1,1,1,1),pch=c(-1,-1,-1,16,22,21,24),pt.bg=c(0,0,0,0,rep("white",3)),
        col=c(4,4,4,rep(1,4)),lwd=c(1,1,2,1,1,1),cex=0.8,pt.cex=c(-1,-1,-1,0.5,rep(1.3,3)),bty="n")

  dev.off()
}

## PLOT BIPLOT ----

if(Biplot==TRUE){

  #-------------------------------------------------------
  # Produce 'post-modern' biplot (see Quinn and Collie 2005)
  #-------------------------------------------------------

  # read ftarget,bthreshold
  ftarget<-0.8
  bthreshold<-0.2

  # fit kernel function
  kernelF <-
ci2d(f,b,nbins=201,factor=2,ci.levels=c(0.50,0.80,0.75,0.90,0.95),show="none",col=1,ylab=
ifelse(harvest.label=="Fmsy",expression(paste(F/F[MSY])),expression(paste(H/H[MSY]))),xlab
=expression(paste(B/B[MSY])))
```

```
  Par = list(mfrow=c(1,1),mai=c(0.2,0.15,0,.15),omi = c(0.3,0.25,0.2,0) + 0.1, mgp =c(3,1,0), tck
= -0.02,cex=0.8)
  png(file = paste0(output.dir,"/Biplot_",assessment,"_",Scenario,".png"), width = 6, height = 6,
      res = 800, units = "in")
  par(Par)

  #Create plot
  plot(1000,1000,type="b", ylim=c(0,2.5),
xlim=c(0,max(apply(HtoHmsy,2,quantile,c(0.5)),quantile(f,0.85),2.)),lty=3,xaxs="i",yaxs="i")

  # and fill areas using the polygon function
  fint = seq(0.001,100,0.01)
  #Zone X
  xb=bthreshold+(1.0-bthreshold)/ftarget*fint
  xf =  ifelse(xb>1,0.8,fint)
  polygon(c(0,0,xf),c(max(xb),bthreshold,xb),col="green")
  zb = bthreshold+(1.0-bthreshold)*fint
  zf  = ifelse(zb>1,1,fint)
  polygon(c(zf,rep(max(fint),2),rep(0,2)),c(zb,max(zb),0,0,bthreshold),col="red")

  polygon(c(xf,rev(zf)),c(xb,rev(zb)),col="yellow")

  c1 <- c(-1,100)
  c2 <- c(1,1)

  # extract interval information from ci2d object
  # and fill areas using the polygon function
  polygon(kernelF$contours$"0.95",lty=2,border=NA,col="cornsilk4")
  polygon(kernelF$contours$"0.8",border=NA,lty=2,col="grey")
  polygon(kernelF$contours$"0.5",border=NA,lty=2,col="cornsilk2")
  points(mu.f[2,],mu.b[2,],pch=16,cex=1)

  lines(c1,c2,lty=3,lwd=0.7)
  lines(c2,c1,lty=3,lwd=0.7)
  lines(mu.f[2,],mu.b[2,], lty=1,lwd=1.)
  sel.yr = c(1,round(quantile(1:N,0.7),0),N)
  points(mu.f[2,sel.yr],mu.b[2,sel.yr],col=
         1,pch=c(22,21,24),bg="white",cex=1.9)


  sel.years = years[sel.yr]
  ## Add legend
  legend('topright',
        c(paste(sel.years),"50% C.I.","80% C.I.","95% C.I."),
        lty=c(1,1,1,-1,-1,-
1),pch=c(22,21,24,22,22,22),pt.bg=c(rep("white",3),"cornsilk2","grey","cornsilk4"),
```

```
       col=1,lwd=1.1,cex=0.9,pt.cex=c(rep(1.3,4),1.7,1.7,1.7),bty="n")



  Zone  = NULL
  Status = NULL
  X  = 0.15
  Y = 0
  Z = -0.15

  for(i  in 1:length(f))
  {
   if(b[i]>1.0){
     if(f[i]<ftarget){
       Zone[i]<-X
     } else if (f[i]>1.0){
       Zone[i]<-Z
     } else {
       Zone[i]<-Y
     }
   } else {
     if(b[i]>bthreshold+(1.0-bthreshold)/ftarget*f[i]){
       Zone[i]<-X
     } else if(b[i]<bthreshold+(1.0-bthreshold)*f[i]){
     } else {
       Zone[i]<-Y
     }
   }}

  perGreen = round(length(Zone[Zone==0.15])/length(Zone)*100,1)
  perYellow = round(length(Zone[Zone==0])/length(Zone)*100,1)
  perRed = round(length(Zone[Zone==-0.15])/length(Zone)*100,1)

  mtext(expression(paste(B/B[MSY])), side=2, outer=TRUE, at=0.5,line=1,cex=0.9)

mtext(ifelse(harvest.label=="Fmsy",expression(paste(F/F[MSY])),expression(paste(H/H[MSY]))
), side=1, outer=TRUE, at=0.5,line=1,cex=0.9)

  text(0.65,2.4,paste0(perGreen,"%"))
  text(0.9,2.4,paste0(perYellow,"%"))
  text(1.2,2.4,paste0(perRed,"%"))

  dev.off()

}
```

```
## PLOT PROJECTIONS ----

cat(paste0("\n","-Plotting TAC Projections","\n"))


m=median(posteriors$m) #added by MDFitchett, needed to be corrected
BmsyK=(m)^(-1/(m-1))

## changed projection plots to B/BMSY instead of B/K

if(Projection == TRUE){


  Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
  png(file = paste0(output.dir,"/Fig21_Projections_",assessment,"_",Scenario,".png"), width = 7,
height = 5,
      res = 800, units = "in")
  par(Par)

  proj.yrs =  years[n.years]:(years[n.years]+pyrs)
  # Dims 1: saved MCMC,2: Years, 3:alternatic TACs, 4: P,  B/Bmsy, H/Hhcr,Hrate
  projections = array(NA,c(nsaved,length(proj.yrs),nTAC,4))
  for(i in 1:nTAC){
    projections[,,i,1] = cbind(posteriors$P[,(n.years):n.years],posteriors$prP[,,i])
  }

  for(i in 1:nTAC){
    projections[,,i,2] = cbind(posteriors$BtoBmsy[,(n.years):n.years],posteriors$prBtoBmsy[,,i])
  }
  for(i in 1:nTAC){
    projections[,,i,3] =
cbind(posteriors$Overfishing_ind[,(n.years):n.years],posteriors$prOverfishing_ind[,,i])
  }

  for(i in 1:nTAC){
    projections[,,i,4] = cbind(posteriors$H[,(n.years):n.years],posteriors$prH[,,i])
  }

  kjp = kobeJabbaProj(projections,proj.yrs[1])



  # Change here for ICCAT Bmsy plot
  Traj = cbind(posteriors$P[,(n.years):n.years],posteriors$prP[,,nTAC])/BmsyK
```

```
#plot(proj.yrs,apply(Traj,2,mean),ylim=c(0,1),xlim=c(min(proj.yrs),max(proj.yrs)+length(proj.yr
s)*0.2),type="n",ylab="Biomass depletion (B/K)",xlab="Projection Years")

plot(proj.yrs,apply(Traj,2,mean),ylim=c(0,1.05*max(apply(Traj,2,mean))),xlim=c(min(proj.yrs),
max(proj.yrs)),type="n",ylab= expression(paste(B/B[MSY])),xlab="Projection Years")

  cols = rev(seq(0.4,0.9,0.5/nTAC))
  plot.order = (1:nTAC)
  for(j in 1:(nTAC)){
    i =  plot.order[j]
    Traj = cbind(posteriors$P[,(n.years):n.years],posteriors$prP[,,i])/BmsyK

#polygon(c(proj.yrs,rev(proj.yrs)),c(apply(Traj,2,quantile,0.05),rev(apply(Traj,2,quantile,0.95))),
col=grey(cols[i],1),border=NA)
  }
  for(j in 1:(nTAC)){
    i =  plot.order[j]
    Traj = cbind(posteriors$P[,(n.years):n.years],posteriors$prP[,,i])/BmsyK
    lines(proj.yrs,apply(Traj,2,median),col=rev(rainbow(nTAC))[j],lwd=2)
  }
  lines(proj.yrs[1:(yr.now-yr.last+2)],apply(Traj,2,median)[1:(yr.now-yr.last+2)],col=1,lwd=2)
  m=median(posteriors$m) #added by MDFitchett, needed to be corrected
  BmsyK=(m)^(-1/(m-1))
  abline(h=(1-natM),lty=2,lwd=2)

  dev.off()
  save(kjp, file = paste0(output.dir, "/", Scenario, "_projections"))
}

## Attempt to Recreate 'statusResults' table
pBoth  <- pYellow <- pOrange <- pGreen <- p_overfishing <- NULL
for (i in 1:length(years)) {
  ## be sure to count upper quadrant and triangle using OR
  pBoth[i] <-
    sum(
      posteriors$BtoBmsy[,i ] < (1 - natM) & posteriors$Overfishing_ind[,i ] > 1
    ) / nsaved
  pOrange[i] <- sum(
      posteriors$Overfishing_ind[,i ] > 1 &
      posteriors$BtoBmsy[,i ] > (1 - natM)
  ) / nsaved
  p_overfishing[i]<-sum(posteriors$Overfishing_ind[,i ] > 1)/nsaved
  pYellow[i] <- sum(posteriors$BtoBmsy[,i ] < (1 - natM) & posteriors$Overfishing_ind[,i ] <
1)/nsaved
```

```
  pGreen[i] <- sum(posteriors$BtoBmsy[,i ] > (1 - natM) & posteriors$Overfishing_ind[,i ] < 1
)/nsaved
}

table7 <- data.frame("Year" = years,
        "Biomass" = apply(posteriors$SB,2,median),
        "B/Bmsy" =  apply(posteriors$BtoBmsy,2,median),
        "ProbabilityStockOverfished" = apply(posteriors$BtoBmsy,2,FUN = function(x) sum(x <
(1-natM))/nsaved),
        "H" = apply(posteriors$HtoHmsy,2,median)*apply(posteriors$Hmsy,2,median),
        "H/Hmsy" = apply(posteriors$HtoHmsy,2,median),
        "H/HCR"= apply(posteriors$Overfishing_ind,2,median),
        # "ProbabilityOverfishing" = apply(posteriors$HtoHmsy,2,FUN = function(x) sum(x >
1)/10000),
        "ProbabilityOverfishing" = p_overfishing,
       # "ProbabilityOverfishing"= pOrange, #not what we want because mutually exclusive of
pBoth
        "ProbabilityOverfishedOverfishing" = pBoth)

write.csv(table7,file = paste0(output.dir,"/Table7_StatusResults_Update.csv"), row.names = F)

##This table has H monitored and outputted properly
table7b <- data.frame("Year" = years,
                "Biomass" = apply(posteriors$SB,2,median),
                "B/Bmsy" =  apply(posteriors$BtoBmsy,2,median),
                "ProbabilityStockOverfished" = apply(posteriors$BtoBmsy,2,FUN = function(x)
sum(x < (1-natM))/nsaved),
                "H" = apply(posteriors$H,2,median),
                "H/Hmsy" = apply(posteriors$HtoHmsy,2,median),
                "H/HCR"= apply(posteriors$Overfishing_ind,2,median),
                # "ProbabilityOverfishing" = apply(posteriors$HtoHmsy,2,FUN = function(x)
sum(x > 1)/10000),
                "ProbabilityOverfishing" = p_overfishing,
                # "ProbabilityOverfishing"= pOrange, #not what we want because mutually
exclusive of pBoth
                "ProbabilityOverfishedOverfishing" = pBoth)

write.csv(table7b,file = paste0(output.dir,"/Table7b_StatusResults_Update.csv"), row.names = F)

## plot Kobe - modified ----
cat(paste0("\n","-Producing Kobe Plot","\n"))

## mf version
if(KOBE.plot==TRUE){
  # prepare
```

```
  # fit kernel function
  kernelF <-
ci2d(b,f,nbins=151,factor=1.5,ci.levels=c(0.50,0.80,0.75,0.90,0.95),show="none",col=1,xlab=
ifelse(harvest.label=="Fmsy",expression(paste(F/F[MSY])),expression(paste(H/H[MSY]))),ylab
=expression(paste(B/B[MSY])))


  Par = list(mfrow=c(1,1),mar = c(3.5, 3.5, 0.1, 0.1), mgp =c(2.,0.5,0), tck = -0.02,cex=0.8)
  png(file = paste0(output.dir,"/Fig12_Kobe_",assessment,"_",Scenario,".png"), width = 6, height
= 6,
      res = 800, units = "in")
  par(Par)

  #Create plot
  plot(
    1000,
    1000,
    type = "b",
   # xlim = c(0,max(kernelF$contours$"0.95")*1.1),
    xlim = c(0,2),#JS chnged to avoid large range
 #   ylim = c(0, max(
 #     apply(HtoHmsy, 2, quantile, c(0.5)), quantile(f, 0.85), 2.
 #   )),
    ylim = c(0,3),
    lty = 3,
    ylab = ifelse(
      harvest.label == "Fmsy",
      expression(paste(F / F[MSY])),
      expression(paste(H / H[MSY]))
    ),
    xlab = expression(paste(B / B[MSY])),
    xaxs = "i",
    yaxs = "i"
  )
  c1 <- c(-1,100)
  c2 <- c(1,1)

  # extract interval information from ci2d object
  # and fill areas using the polygon function
  zb2 = c(0,1-natM)
  zf2  = c(1,100)
  zb1 = c(1-natM,100)
  zf1  = c(0,1)

  ## fixed HCR
  polygon(c(zb2,rev(zb2)),c(0,0,1,1),col="yellow",border = 'yellow')
```

```
  polygon(c(zb1,rev(zb1)),c(0,0,1,1),col="limegreen",border = 'limegreen')

  polygon(c(1-natM,100,100,1-
natM),c(1,1,100,100),col=ifelse(KOBE.type=="ICCAT","yellow","orange"),border = 'orange')
#  polygon(c(1-natM,1,1-natM),c(1-
natM,1,1),col=ifelse(KOBE.type=="ICCAT","yellow","orange"),border = 'orange')

  polygon(c(0,0,1-natM,1-natM),c(0,1,1, 1),col="red3",border = 'red3')
  polygon(c(0,1-natM,1-natM,0),c(1,1,100,100),col="red3",border = 'red3')

  ## add contours
  polygon(kernelF$contours$"0.95",lty=2,border=NA,col="cornsilk4")
  polygon(kernelF$contours$"0.8",border=NA,lty=2,col="grey")
  polygon(kernelF$contours$"0.5",border=NA,lty=2,col="cornsilk2")
  points(mu.b[2,],mu.f[2,],pch=16,cex=1)


  # lines(c1,c2,lty=3,lwd=0.7)
  # lines(c(1-natM, 1-natM),c1,lty=3,lwd=0.7)
  lines(mu.b[2,],mu.f[2,], lty=1,lwd=1.)
  #sel.yr = c(1,round(quantile(1:N,0.7),0),N) #JS removed 2006
  sel.yr = c(1,N)
  points(mu.b[2,sel.yr],mu.f[2,sel.yr],col=
         1,pch=c(21,24),bg="white",cex=1.9)


  ## get probabilities from table 7 computed above
  Pr.red <- last(pBoth) * 100
  Pr.yellow <- last(pYellow)* 100
  Pr.orange <- last(pOrange)* 100
  Pr.green <- last(pGreen)* 100

  abline(v=0.7, lty=2)
  segments(0,0,0.7,1.0, lty=2)
  segments(0.7,1.0,2.0,1.0, lty=2)

  sel.years = c(years[sel.yr])
  ## Add legend
  if(KOBE.type=="ICCAT"){
    legend('topright',
           c(paste(sel.years),"50% C.I.","80% C.I.","95%
C.I.",paste0(round(c(Pr.red,Pr.yellow,Pr.green),1),"%")),
           lty=c(1,1,rep(-
1,7)),pch=c(21,24,rep(22,7)),pt.bg=c(rep("white",2),"cornsilk2","grey","cornsilk4","red","yellow
","green"),
```

```
        col=1,lwd=1.1,cex=0.9,pt.cex=c(rep(1.3,2),rep(1.7,3),rep(2.1,3)),bty="n")
  }else{
    legend('topright',
        c(paste(sel.years),"50% C.I. 2017","80% C.I. 2017","95% C.I.
2017",paste0(round(c(Pr.red,Pr.yellow,Pr.orange,Pr.green),1),"% 2017")),
        lty=c(1,1,rep(-
1,8)),pch=c(21,24,rep(22,8)),pt.bg=c(rep("white",2),"cornsilk2","grey","cornsilk4","red","yellow
","orange","green"),
        col=1,lwd=1.1,cex=0.9,pt.cex=c(rep(1.3,2),rep(1.7,3),rep(2.2,4)),bty="n")

  }
  dev.off()
}
```

Supplement 2. Data scripts for calculating catch time series and CPUE index within R for use in the assessment models

2.1. Guam

2.1.1. Calculating group proportions

```
#####################
#Script to assign proportions by year to group species codes

#Ten groups for Guam (BBS-SBS):
#Carangidae, Juvenile Caranx (same as Car.), Lethrinidae, Lutjanidae,
#Serranidae, Assorted.Bottomfish, Shallow.Bottomfish (same as A.B),
#Deep.Bottom.Fish (samae as A.B),Shallow.Snappers (same as Lutj.),
#Deep.Snappers (same as Lutj.)

#Seven groups for Guam (ComPurc):
#Jacks, Bottom Fish, Deep bottom (same as BF), Grouper,
#Tagafi-RedSnapper (same as snapper), Mafutu(emperor), and Snapper
#####################

rm(list=ls())
loc="Location where the data are"

##############Guam############################
#Read in catch data

#BBS
gdatabbs=read.csv(paste0(loc,"Guam\\G_BBS_SPC_R.csv"),header=T,na.strings="NULL")
gdatabbs$year=as.numeric(substr(gdatabbs$SPC_PK,2,5))
gdatabbs2017=read.csv(paste0(loc,"Guam\\G_BBS_SPC2017.csv"),header=T)
gdatabbs2017$year=2017
gdatabbs=rbind(gdatabbs,gdatabbs2017[,-1]) #combine all years into one database
names(gdatabbs)[5]="KG_CAUGHT" #EVEN THOUGH GUAM BBS FIELDS SAY LBS,
VALUES ARE IN KGS!
#SBS
gdatasbs=read.csv(paste0(loc,"Guam\\G_SBS_SPC_R_31May18.csv"),header=T)
gdatasbs$year=substr(gdatasbs$SPC_PK,1,4)
#Commercial Purchase
gdatacp=read.csv(paste0(loc,"Guam\\guam_gcl_commercial_14May18.csv"),header=T)
gdatacp$year=as.numeric(substr(gdatacp$INVOICE_DATE,1,4))
gdatacp=gdatacp[!is.na(gdatacp$year),] #remove the records without a date
gdatacp$KGS_SOLD=gdatacp$TOT_LBS/2.20462 #LBS is properly labeled so convert to KG
gdatacp=gdatacp[!gdatacp$RESALE_F,] #remove RESALE_F=T records. These fish have
already been sold.

#Read in species key data
```

```
ggroups=read.csv(paste0(loc,"Species lists\\Guam_BBS-SBS_GroupKey_final.csv"),header=T)
gspecies=read.csv(paste0(loc,"Species
lists\\Guam_BFspecieslist_2018.csv"),header=T)[,c("species","spec_name")]
ggroupscp=read.csv(paste0(loc,"Species
lists\\Guam_ComPurc_GroupKey_final.csv"),header=T)
gspeciescp=read.csv(paste0(loc,"Species
lists\\Guam_ComPurc_BFspecieslist_2018.csv"),header=T)[,c("SPECIES_PK","SPECIES_NA
ME")]

#Determine amount of catch from managed species within groups that
#could include managed species, as stated at the beginning of this
#code script.
g.perc.bbs=c(1982:2017)
g.perc.sbs=c(1985:2017)
g.perc.cp=data.frame("year"=1979:2017) #some groups dont appear in each year
for(i in 7:16){
  g.X.codes=ggroups[ggroups[,i]==1,"SPECIES_PK"]
  #BBS

g.Xbbs.all=aggregate(KG_CAUGHT~year,data=subset(gdatabbs,SPECIES_FK%in%g.X.codes)
,FUN=sum)

g.Xbbs.bmus=aggregate(KG_CAUGHT~year,data=subset(gdatabbs,SPECIES_FK%in%g.X.cod
es & SPECIES_FK%in%gspecies$species),FUN=sum)
  g.Xbbs.perc=g.Xbbs.bmus$KG_CAUGHT/g.Xbbs.all$KG_CAUGHT #For 1982-2017
  g.perc.bbs=cbind(g.perc.bbs,g.Xbbs.perc)
  dimnames(g.perc.bbs)[[2]][i-5]=substr(names(ggroups)[i],8,nchar(names(ggroups)[i]))
  plot(x=g.perc.bbs[,1],y=g.perc.bbs[,i-5],type="l",main=paste(dimnames(g.perc.bbs)[[2]][i-
5],"BBS"),ylim=c(0,1))
  #SBS

g.Xsbs.all=aggregate(EXP_KGS~year,data=subset(gdatasbs,SPECIES_FK%in%g.X.codes),FU
N=sum)
  g.Xsbs.bmus=aggregate(EXP_KGS~year,data=subset(gdatasbs,SPECIES_FK%in%g.X.codes
& SPECIES_FK%in%gspecies$species),FUN=sum)
  temp=merge(g.Xsbs.all,g.Xsbs.bmus,by="year",all.x=T,suffixes=c(".all",".bmus"))
  g.Xsbs.perc=temp$EXP_KGS.bmus/temp$EXP_KGS.all #For 1985-2017
  g.Xsbs.perc[is.na(g.Xsbs.perc)]=0 #set NAs to zero (since no BMUS in that year)
  g.perc.sbs=cbind(g.perc.sbs,g.Xsbs.perc)
  dimnames(g.perc.sbs)[[2]][i-5]=substr(names(ggroups)[i],8,nchar(names(ggroups)[i]))
  plot(x=g.perc.sbs[,1],y=g.perc.sbs[,i-5],type="l",main=paste(dimnames(g.perc.sbs)[[2]][i-
5],"SBS"),ylim=c(0,1))
}
dimnames(g.perc.bbs)[[2]][1]="year"
dimnames(g.perc.sbs)[[2]][1]="year"
```

```
#There is no species within Emperor or Grouper categories. We cannot assign
#catch to these groups. Thus assign as 0. ComPurch is generally low enough
#so that this assumption doesn't affect results.
for(i in 5:11){
  #Commercial Purchase
  g.X.codes.cp=ggroupscp[ggroupscp[,i]==1,"SPECIES_PK"]
  if(i==8 | i==10){ #For the group 'grouper' and 'mafute(emperor)' there are no species codes
within them
    g.Xcp.all=data.frame("year"=1979:2017,"KGS_SOLD"=0)
    g.Xcp.bmus=data.frame("year"=1979:2017,"KGS_SOLD"=0)
  }else{

g.Xcp.all=aggregate(KGS_SOLD~year,data=subset(gdatacp,SPECIES_FK%in%g.X.codes.cp),F
UN=sum)

g.Xcp.bmus=aggregate(KGS_SOLD~year,data=subset(gdatacp,SPECIES_FK%in%g.X.codes.cp
& SPECIES_FK%in%gspeciescp$SPECIES_PK),FUN=sum)
  }
  temp=merge(g.Xcp.all,g.Xcp.bmus,by="year",all.x=T,suffixes=c(".all",".bmus"))
  temp[is.na(temp$KGS_SOLD.bmus),"KGS_SOLD.bmus"]=0 #set NAs to zero (since no
BMUS in that year)
  temp2=merge(g.perc.cp,temp,by="year",all.x=T) #Way to ensure all years have an entry
  g.Xcp.perc=temp2$KGS_SOLD.bmus/temp2$KGS_SOLD.all #For 1979-2017
  g.Xcp.perc[is.na(g.Xcp.perc)]=sum(temp$KGS_SOLD.bmus)/sum(temp$KGS_SOLD.all) #set
NAs to be average across years (since no species from group reported separately in that year, so
use overall average)
  if(i==8 | i==10) {g.Xcp.perc=as.numeric(is.finite(g.Xcp.perc))} #If overall average is NAs,
then set all to zero (since no species from group in any year)
  g.perc.cp=cbind(g.perc.cp,g.Xcp.perc)
  dimnames(g.perc.cp)[[2]][i-3]=substr(names(ggroupscp)[i],6,nchar(names(ggroupscp)[i]))
  plot(x=g.perc.cp[,1],y=g.perc.cp[,i-3],type="l",main=paste(dimnames(g.perc.cp)[[2]][i-
3],"ComPurc"),ylim=c(0,1))
}

###############Output final percentages#############################
output_loc="Location where want to output"
write.csv(round(g.perc.bbs,3),paste0(output_loc,"SpeciesGroups\\Gbbs_GroupProportions.csv"),
row.names=F)
write.csv(round(g.perc.sbs,3),paste0(output_loc,"SpeciesGroups\\Gsbs_GroupProportions.csv"),r
ow.names=F)
write.csv(round(g.perc.cp,3),paste0(output_loc,"SpeciesGroups\\GComPurc_GroupProportions.c
sv"),row.names=F)
```

2.1.2. Catch data filtering and calculation

```
###
```

```
#Calculated total catch used in the model from all three catch datasets.
#Applies previously calculated group proportions for each dataset.
###

rm(list=ls())
library(dplyr)
library(tidyr)
loc="Folder where datasets reside"

##############Guam############################
#BMUS species/groups
gsp <- read.csv(paste0(loc,"Guam_BFspecieslist_2018.csv"),header=T)
gsp.cp <- read.csv(paste0(loc,"Guam_ComPurc_BFspecieslist_2018.csv"),header=T)

#Proportions of species groups that are BMUS
gspg.prop.bbs=read.csv(paste0(loc,"Gbbs_GroupProportions.csv"),header=T)
gspg.prop.sbs=read.csv(paste0(loc,"Gsbs_GroupProportions.csv"),header=T)
gspg.prop.cp=read.csv(paste0(loc,"GComPurc_GroupProportions.csv"),header=T)

###
#BBS data
###
#Combined 2017 data with rest of data. Ensure NULLs are set as NAs so that values
#are read in as numbers.
gbbs=rbind(read.csv(paste0(loc,"G_BBS_SPC_R.csv"),header=T,na.strings="NULL"),
        read.csv(paste0(loc,"G_BBS_SPC2017.csv"),header=T)[,-1])
names(gbbs)[5]="KGS_CAUGHT" #Values are in KGS despite field name of LBS
gbbs$year=as.numeric(substr(gbbs$SPC_PK,2,5))

#Sum catches by year and species
gbbs.yrsums = gbbs %>%
  filter(SPECIES_FK %in% gsp$species) %>%
  select(SPECIES_FK,KGS_CAUGHT,year) %>%
  group_by(year,SPECIES_FK) %>%
  summarise(total = sum(KGS_CAUGHT,na.rm=T))
#table the catches by year (rows) and species codes (columns)
gbbs.table = spread(gbbs.yrsums,"SPECIES_FK","total")

#Match the columns in cbbs.table with associated species groups so as
#to apply proportion. Then....
group.cols=match(gsp$species,as.numeric(names(gbbs.table)[-1]))+1
cbind(gsp[,c("species","spec_name")],group.cols)
#....multiply group catch by the corresponding proportions of that group that are BMUS
gbbs.table[,group.cols[15:24][!is.na(group.cols[15:24])]]=gbbs.table[,group.cols[15:24][!is.na(gr
oup.cols[15:24])]]*(gspg.prop.bbs[,-1][,which(!is.na(group.cols[15:24]))])
```

```
##
#Add variance now for BBS only (1982-2017).
##
gvar.bbs=read.csv(paste0(loc,"G_BBS_SPCbootstrap_31May18.csv"),header=T)
gvar.bbs=gvar.bbs[,c("year","species_fk","common_name","catch","var_catch","proportion_zer
o_catch")]
#Merge with yrsum catches.
gbbs.yrsums.wvar=merge(gbbs.yrsums,gvar.bbs,by.x=c("year","SPECIES_FK"),by.y=c("year","
species_fk"))

#Repeat process for applying group proportions to variance as for catch above
gbbs.table.wvar =
spread(gbbs.yrsums.wvar[,c("year","SPECIES_FK","var_catch")],"SPECIES_FK","var_catch")
group.cols.wvar=match(gsp$species,as.numeric(names(gbbs.table.wvar)[-1]))+1
cbind(gsp[,c("species","spec_name")],group.cols)
gbbs.table.wvar[,group.cols.wvar[15:24][!is.na(group.cols.wvar[15:24])]]=gbbs.table.wvar[,grou
p.cols.wvar[15:24][!is.na(group.cols.wvar[15:24])]]*((gspg.prop.bbs[,-
1][,which(!is.na(group.cols.wvar[15:24]))])*(gspg.prop.bbs[,-
1][,which(!is.na(group.cols.wvar[15:24]))]))


###
#SBS data
###
gsbs=read.csv(paste0(loc,"G_SBS_SPC_R_31May18.csv"),header=T)
gsbs$year=as.numeric(substr(gsbs$SPC_PK,1,4))

#Sum catches by year and species
gsbs.yrsums = gsbs %>%
  filter(SPECIES_FK %in% gsp$species) %>%
  select(SPECIES_FK,EXP_KGS,year) %>%
  group_by(year,SPECIES_FK) %>%
  summarise(total = sum(EXP_KGS,na.rm=T))
#table the catches by year (rows) and species codes (columns)
gsbs.table = spread(gsbs.yrsums,"SPECIES_FK","total")

#Match columns in tabled catches with groups and then...
group.cols=match(gsp$species,as.numeric(names(gsbs.table)[-1]))+1
cbind(gsp[,c("species","spec_name")],group.cols)
#....multiply group catch by the corresponding proportions of that group that are BMUS
gsbs.table[,group.cols[15:24][!is.na(group.cols[15:24])]]=gsbs.table[,group.cols[15:23][!is.na(gr
oup.cols[15:24])]]*(gspg.prop.sbs[,-1][,which(!is.na(group.cols[15:24]))])


###
#Commercial Purchase Data
```

```
###
gcp=read.csv(paste0(loc,"guam_gcl_commercial_14May18.csv"),header=T)
gcp$year=as.numeric(substr(gcp$INVOICE_DATE,1,4))
gcp=gcp[!is.na(gcp$year),]
gcp$KGS_SOLD=gcp$LBS_SOLD/2.20462 #LBS is properly labelled so convert to KG
gcp1=gcp[!gcp$RESALE_F,] #remove RESALE_F=T records. These fish have already been
sold.

#Sum catches by year and species
gcp.yrsums = gcp1 %>%
  filter(SPECIES_FK %in% gsp.cp$SPECIES_PK) %>%
  select(SPECIES_FK,KGS_SOLD,year) %>%
  group_by(year,SPECIES_FK) %>%
  summarise(total = sum(KGS_SOLD,na.rm=T))
#table the catches by year (rows) and species codes (columns)
gcp.table = spread(gcp.yrsums,"SPECIES_FK","total")

#Match columns in tabled catches with groups and then...
group.cols=match(gsp.cp$SPECIES_PK,as.numeric(names(gcp.table)[-1]))+1
cbind(gsp.cp[,c("SPECIES_PK","SPECIES_NAME")],group.cols)
#....multiply group catch by the corresponding proportions of that group that are BMUS
gcp.table[,group.cols[8:14][!is.na(group.cols[8:14])]]=gcp.table[,group.cols[8:14][!is.na(group.c
ols[8:14])]]*(gspg.prop.cp[,-1][,which(!is.na(group.cols[8:14]))])


###
#Final catch by year
###
gbbs.TOTC=data.frame("year"=gbbs.table[,1],"gbbs.kg"=rowSums(gbbs.table[,-
1],na.rm=T),"gbbs.kg.var"=rowSums(gbbs.table.wvar[,-1],na.rm=T))
gsbs.TOTC=data.frame("year"=gsbs.table[,1],"gsbs.kg"=rowSums(gsbs.table[,-1],na.rm=T))
gcp.TOTC=data.frame("year"=gcp.table[,1],"gcp.kg"=rowSums(gcp.table[,-1],na.rm=T))
g.TOTC=merge(merge(gbbs.TOTC,gsbs.TOTC,by="year",all=T),gcp.TOTC,by="year",all=T)
write.csv(g.TOTC,paste0(loc,"TotalCatch_Guam_init.csv"),row.names=F)



###############Algorithms for combining missing data in
catches############################
#Read in all files
catch.folder="Folder where catch data by dataset are located"
gCatch=read.csv(paste0(catch.folder,"TotalCatch_Guam_init.csv"),header=T)
cCatch=read.csv(paste0(catch.folder,"TotalCatch_CNMI_init.csv"),header=T)

#Combine files into single variable
allCatch=round(merge(gCatch,cCatch,by="year",all=T),1)
```

```
#Algorithm for filling in missing values
#0. Set dataset to start in 1982 (remove the first three years from G ComPurc)
allCatch=allCatch[allCatch$year%in%c(1982:2017),]
#1. For filling missing SBS values, use overall average SBS catch.
areNAg=is.na(allCatch$gsbs.kg)
allCatch$gsbs.kg[areNAg]=mean(allCatch$gsbs.kg,na.rm=T)
#2. Sum together SBS and BBS data.
allCatch$gTOT.kg=round(allCatch$gbbs.kg+allCatch$gsbs.kg,0)
#3. Take max of SBS+BBS and original ComPurc data
gmax=which(allCatch$gTOT.kg<allCatch$gcp.kg) #ComPurc always lower
#4. Add in CVs for BBS data for final file output
allCatch$gCV=round(sqrt(allCatch$gbbs.kg.var)/allCatch$gbbs.kg,2)

##
#Output final datasets for use in the assessment
write.csv(allCatch[!is.na(allCatch$gTOT.kg),c("year","gTOT.kg","gCV")],paste0(loc,"TotalCatc
h_Guam_FINAL.csv"),row.names=F)
```

2.1.3. CPUE data filtering

```
###
#Steps to obtain nominal CPUE dataset
###

rm(list=ls())
library(dplyr)
library(tidyr)
loc="Folder where datasets reside"

############## Guam ############################

######
#READ IN DATA SETUP FILES
######

#BMUS species/groups
gsp <- read.csv(paste0(loc,"Guam_BFspecieslist_2018.csv"),header=T)
#Interview data
gint.bbs=read.csv(paste0(loc,"guam_bbs_interview_14May18.csv"),header=T)
gint.bbs$SAMPLE_DATE=as.Date(gint.bbs$SAMPLE_DATE)
gint.bbs$year=as.numeric(format(gint.bbs$SAMPLE_DATE,"%Y"))
gint.bbs$month=as.numeric(format(gint.bbs$SAMPLE_DATE,"%m"))
#Proportions of species groups that are BMUS
g.proptable.bbs=read.csv(paste0(loc,"Gbbs_GroupProportions.csv"),header=T)
```

```
######
#CPUE CALCULATIONS SETUP
######

##
#Add group proportions
##
#Reduce group proportion table to row-column key of proportions, with
#species key instead of species name as column header
colnames(g.proptable.bbs)=c("year",as.character(gsp$species[15:24]))
g.propkey.bbs=gather(g.proptable.bbs,key="SPECIES_FK",value="prop",-1)

#Add in group proportionality as the value from the group proportion
#key for each group and year. For non-group species, set proportion to 1
gint1.bbs=merge(gint.bbs,g.propkey.bbs,by=c("year","SPECIES_FK"),all.x=T)
gint1.bbs[is.na(gint1.bbs$prop),"prop"]=1

#Multiply proportion by EST_KGS, which is the species specific total
gint1.bbs$est_kgs=gint1.bbs$EST_KGS*gint1.bbs$prop

##
#Calculate proportion of BMUS by interview. There is only one method per interview.
##
#Assign proportion of BMUS by trip. Since weights for groups are already
#allocated to BMUS/nonBMUS, include species group keys too.
#Must first reduce data into unique combinations of interview, species, and pounds.
temp.bbs=unique(gint1.bbs[,c("INTERVIEW_PK","SPECIES_FK","EST_KGS","est_kgs")])
bbs.all=aggregate(EST_KGS~INTERVIEW_PK,data=temp.bbs,FUN=sum)
bbs.bmus=aggregate(est_kgs~INTERVIEW_PK,data=temp.bbs[temp.bbs$SPECIES_FK%in%g
sp$species,],FUN=sum)
bbs.perctrip=merge(bbs.all,bbs.bmus,by="INTERVIEW_PK",all.x=T)
names(bbs.perctrip)[2:3]=c("EST_KGS.all","est_kgs.bmus")
bbs.perctrip[is.na(bbs.perctrip$est_kgs.bmus),"est_kgs.bmus"]=0
bbs.perctrip$percBMUS=bbs.perctrip$est_kgs.bmus/bbs.perctrip$EST_KGS.all
gint2.bbs=merge(gint1.bbs,bbs.perctrip,by="INTERVIEW_PK",all.x=T)


##
#Add effort and cpue
##
gint2.bbs$effort=gint2.bbs$NUM_GEAR*gint2.bbs$HOURS_FISHED
gint2.bbs$cpue=gint2.bbs$est_kgs.bmus/gint2.bbs$effort

##
```

```
#Filter dataset. Bottom method, vessels that caught BMUS, data with valid fields
##
#Reduce dataset to pertinent fields
gint3.bbs=gint2.bbs[,c("INTERVIEW_PK","year","SPECIES_FK","SAMPLE_DATE","TYPE_
OF_DAY",
                "INTERVIEW_TIME","AREA_FK","METHOD_FK","METHOD_NAME",

"WEATHER_NAME","WIND_DIRECTION","WIND_SPEED","DEPTH","HOURS_FISHED"
,
                "NUM_FISHER","NUM_GEAR","VESSEL_NAME","PORT_NAME",
                "TOT_EST_KGS","TOT_SPECIES","LUNAR_DAY","EST_KGS",
                "prop","est_kgs","EST_KGS.all","est_kgs.bmus","percBMUS",
                "effort","cpue","month","CHARTER_F")]
#Remove SPECIES_FK, est_kgs, prop, and EST_KGS and reduce to unique trips
gint4.bbs=unique(gint3.bbs[,-c(3,22,23,24)])

#Limit interviews to be those on Method "Bottom", METHOD_FK=2
gint4.bbs=gint4.bbs[gint4.bbs$METHOD_NAME=="BOTTOM",]

#Remove the 314 vessels (and their 468 interviews) that never caught any BMUS
tempvg=aggregate(est_kgs.bmus~VESSEL_NAME,data=gint4.bbs,FUN=sum,na.rm=T)
zero.vessg=tempvg[tempvg$est_kgs.bmus==0,"VESSEL_NAME"]
gint5.bbs=gint4.bbs[!gint4.bbs$VESSEL_NAME%in%zero.vessg,]

#Use fields: type of day, area, wind speed, depth category, vessel name, and charter indicator.
#Remove any interviews without records of these.
gint6.bbs=gint5.bbs[,c("year","month","cpue","effort","TYPE_OF_DAY","AREA_FK","WIND
_SPEED","DEPTH","VESSEL_NAME","CHARTER_F")]
gint6.bbs=gint6.bbs[is.finite(gint6.bbs$cpue),]
gint6.bbs=gint6.bbs[!is.na(gint6.bbs$VESSEL_NAME),]
gint6.bbs[is.na(gint6.bbs$DEPTH),"DEPTH"]="U"
gint6.bbs=droplevels(gint6.bbs)

##
#Output dataset
##
write.csv(gint6.bbs,paste0(loc,"FinalizedGUAM_intCPUE_forStandardization_080318.csv"),ro
w.names=F)
```

2.1.4. CPUE standardization

```
###
#Steps to perform CPUE standardization
###
```

```
rm(list=ls())
library(lme4)
library(MASS)
library(statmod)
loc="Folder where nominal dataset resides"

#<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>#
############### Guam ############################
#<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>#

datag=read.csv(paste0(loc,"FinalizedGUAM_intCPUE_forStandardization_080318.csv"),header
=T)

#Turn year, month, area, wind speed into factors
datag$year=as.factor(datag$year)
datag$month=as.factor(datag$month)
datag$AREA_FK=as.factor(datag$AREA_FK)
datag$WIND_SPEED=as.factor(5*ceiling(datag$WIND_SPEED/5))
#Add 0-1 indicator variable
datag$z=0
datag[datag$cpue>0,"z"]=1
#Remove interviews from area 99, which is northern marianas, and the
#interviews without a valid area code
datag=datag[!(datag$AREA_FK==99 | datag$AREA_FK==0 | datag$AREA_FK==9),]
datag$AREA_FK=droplevels(datag$AREA_FK)
#Remove charters (CHARTER_F equals T)
datag=datag[!(datag$CHARTER_F),]
#Set up positive only database
datag.pos=datag[datag$cpue>0,]


####
#BERNOULLI
####

#Full model
gb.model0=glm(z~year+month+AREA_FK+TYPE_OF_DAY+WIND_SPEED+DEPTH,family
=binomial,data=datag)
gb.model0a=glm(z~year+month+AREA_FK+TYPE_OF_DAY+WIND_SPEED,family=binomi
al,data=datag)
gb.model0b=glm(z~year+month+AREA_FK+TYPE_OF_DAY+DEPTH,family=binomial,data=
datag)
gb.model0c=glm(z~year+month+AREA_FK+WIND_SPEED+DEPTH,family=binomial,data=d
atag)
gb.model0d=glm(z~year+month+TYPE_OF_DAY+WIND_SPEED+DEPTH,family=binomial,d
ata=datag)
```

```
gb.model0e=glm(z~year+AREA_FK+TYPE_OF_DAY+WIND_SPEED+DEPTH,family=bino
mial,data=datag)
gbAIC0=rbind("Full"=extractAIC(gb.model0),"-D"=extractAIC(gb.model0a),"-
WS"=extractAIC(gb.model0b),
        "-TD"=extractAIC(gb.model0c),"-A"=extractAIC(gb.model0d),"-
M"=extractAIC(gb.model0e))
gbAIC0=cbind(gbAIC0,(gbAIC0[,2]-gbAIC0[1,2])) #if values is at least +2, then remove

which.min(gbAIC0[,3]) #Remove wind speed (0b)

#Step 1 - removed wind speed
gb.model1a=glm(z~year+month+AREA_FK+TYPE_OF_DAY,family=binomial,data=datag)
gb.model1b=glm(z~year+month+AREA_FK+DEPTH,family=binomial,data=datag)
gb.model1c=glm(z~year+month+TYPE_OF_DAY+DEPTH,family=binomial,data=datag)
gb.model1d=glm(z~year+AREA_FK+TYPE_OF_DAY+DEPTH,family=binomial,data=datag)
gbAIC1=rbind("Full-WS"=extractAIC(gb.model0b),"-D-WS"=extractAIC(gb.model1a),
        "-TD-WS"=extractAIC(gb.model1b),"-A-WS"=extractAIC(gb.model1c),
        "-M-WS"=extractAIC(gb.model1d))
gbAIC1=cbind(gbAIC1,(gbAIC1[,2]-gbAIC1[1,2])) #if values is at least +2, then remove

which.min(gbAIC1[,3]) #Remove month (1d)

#Step 2 - removed wind speed and month
gb.model2a=glm(z~year+AREA_FK+TYPE_OF_DAY,family=binomial,data=datag)
gb.model2b=glm(z~year+AREA_FK+DEPTH,family=binomial,data=datag)
gb.model2c=glm(z~year+TYPE_OF_DAY+DEPTH,family=binomial,data=datag)
gbAIC2=rbind("Full-WS-M"=extractAIC(gb.model1d),"-D-WS-M"=extractAIC(gb.model2a),
        "-TD-WS-M"=extractAIC(gb.model2b),"-A-WS-M"=extractAIC(gb.model2c))
gbAIC2=cbind(gbAIC2,(gbAIC2[,2]-gbAIC2[1,2])) #if values is at least +2, then remove

which.min(gbAIC2[,3]) #Remove type of day (2b)

#Step 3 - removed wind speed and month and type of day
gb.model3a=glm(z~year+AREA_FK,family=binomial,data=datag)
gb.model3b=glm(z~year+DEPTH,family=binomial,data=datag)
gbAIC3=rbind("Full-WS-M-TD"=extractAIC(gb.model2b),"-D-WS-M-
TD"=extractAIC(gb.model3a),
        "-A-WS-M-TD"=extractAIC(gb.model3b))
gbAIC3=cbind(gbAIC3,(gbAIC3[,2]-gbAIC3[1,2])) #if values is at least +2, then remove
gbAIC3

gb.modelBEST = gb.model2b #BEST MODEL CONTAINS ONLY AREA AND DEPTH


####
#LOGNORMAL
```

```
####
#Use year, month, type of day, wind speed, depth, area, vessel name

#Full model, for random effects
gp.model0=lmer(log(cpue)~year+(1|VESSEL_NAME)+month+AREA_FK+TYPE_OF_DAY+
WIND_SPEED+DEPTH,REML=F,data=datag.pos)
gp.model0a=glm(log(cpue)~year+month+AREA_FK+TYPE_OF_DAY+WIND_SPEED+DEPT
H,data=datag.pos)

AIC(gp.model0) - AIC(gp.model0a) #Full model has lower AIC. Keep Vessel

#Full model, for fixed effects
gp.model1a=lmer(log(cpue)~year+(1|VESSEL_NAME)+month+AREA_FK+TYPE_OF_DAY+
WIND_SPEED,REML=F,data=datag.pos)
gp.model1b=lmer(log(cpue)~year+(1|VESSEL_NAME)+month+AREA_FK+TYPE_OF_DAY+
DEPTH,REML=F,data=datag.pos)
gp.model1c=lmer(log(cpue)~year+(1|VESSEL_NAME)+month+AREA_FK+WIND_SPEED+D
EPTH,REML=F,data=datag.pos)
gp.model1d=lmer(log(cpue)~year+(1|VESSEL_NAME)+month+TYPE_OF_DAY+WIND_SPE
ED+DEPTH,REML=F,data=datag.pos)
gp.model1e=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+TYPE_OF_DAY+WIND_
SPEED+DEPTH,REML=F,data=datag.pos)
gpAIC1=rbind("Full"=extractAIC(gp.model0),"-D"=extractAIC(gp.model1a),"-
WS"=extractAIC(gp.model1b),
        "-TD"=extractAIC(gp.model1c),"-A"=extractAIC(gp.model1d),"-
M"=extractAIC(gp.model1e))
gpAIC1=cbind(gpAIC1,(gpAIC1[,2]-gpAIC1[1,2])) #if values is at least +2, then remove

which.min(gpAIC1[,3]) #Remove month (1e)

#Step 1 - removed month
gp.model2a=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+TYPE_OF_DAY+WIND_
SPEED,REML=F,data=datag.pos)
gp.model2b=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+TYPE_OF_DAY+DEPTH
,REML=F,data=datag.pos)
gp.model2c=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+WIND_SPEED+DEPTH,
REML=F,data=datag.pos)
gp.model2d=lmer(log(cpue)~year+(1|VESSEL_NAME)+TYPE_OF_DAY+WIND_SPEED+DE
PTH,REML=F,data=datag.pos)
gpAIC2=rbind("Full-M"=extractAIC(gp.model1e),"-M-D"=extractAIC(gp.model2a),"-M-
WS"=extractAIC(gp.model2b),
        "-M-TD"=extractAIC(gp.model2c),"-M-A"=extractAIC(gp.model2d))
gpAIC2=cbind(gpAIC2,(gpAIC2[,2]-gpAIC2[1,2])) #if values is at least +2, then remove

which.min(gpAIC2[,3]) #Remove wind speed (2b)
```

```
#Step 2 - removed month and wind speed
gp.model3a=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+TYPE_OF_DAY,REML=
F,data=datag.pos)
gp.model3b=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+DEPTH,REML=F,data=d
atag.pos)
gp.model3c=lmer(log(cpue)~year+(1|VESSEL_NAME)+TYPE_OF_DAY+DEPTH,REML=F,d
ata=datag.pos)
gpAIC3=rbind("Full-M-WS"=extractAIC(gp.model2b),"-WS-M-D"=extractAIC(gp.model3a),"-
WS-M-TD"=extractAIC(gp.model3b),
        "-TD-M-A"=extractAIC(gp.model3c))
gpAIC3=cbind(gpAIC3,(gpAIC3[,2]-gpAIC3[1,2])) #if values is at least +2, then remove

which.min(gpAIC3[,3]) #Remove type of day (3b)

#Step 3 - removed month and type of day and wind speed
gp.model4a=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK,REML=F,data=datag.pos)
gp.model4b=lmer(log(cpue)~year+(1|VESSEL_NAME)+DEPTH,REML=F,data=datag.pos)
gpAIC4=rbind("Full-M-WS-TD"=extractAIC(gp.model3b),"-TD-WS-M-
D"=extractAIC(gp.model4a),
        "-TD-WS-M-A"=extractAIC(gp.model4b))
gpAIC4=cbind(gpAIC4,(gpAIC4[,2]-gpAIC4[1,2])) #if values is at least +2, then remove
gpAIC4

#BEST MODEL CONTAINS AREA AND DEPTH AND VESSEL
gp.modelBEST=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+DEPTH,REML=T,dat
a=datag.pos)


######## Save Rdata output ##########
save(datac,datag,gb.modelBEST,gp.modelBEST,cb.modelBEST,cp.modelBEST,file=paste0(loc,
"best_mods_GC_nocharter_080618.RData"))
```

2.1.5. CPUE index calculation

```
###
#Steps to calculate standardized index
###

rm(list=ls())
loc="Folder where standardized model scripts reside"
load(file=paste0(loc,"best_mods_GC_nocharter_080618.RData"))

datag.pos=datag[datag$cpue>0,]

#<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>#
```

```
############## Guam ############################
#<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>#

## Extract predicted values for Bernoulli process ("b")
gb.pred = data.frame('BIN.CATCH' = predict(gb.modelBEST, type = 'response'),'year' =
datag[,'year'])
## get means, sd and variance for each year
gb.pred.a = aggregate(BIN.CATCH ~ year, gb.pred,  FUN = function(x) c(mean = mean(x), sd =
sd(x), var = var(x)))

## Extract predicted values for Positive process ("p")
gp.pred = data.frame('LOGCPUE' = predict(gp.modelBEST), 'year' = datag.pos[,'year'])
## get means and sd for each year
gp.pred.a = aggregate(LOGCPUE ~ year, gp.pred, function(x) c(mean = mean(x), sd = sd(x), var
= var(x)))
## Backtransform positive process using dispersion
gp.trans =
list("year"=c(1982:2017),"mean"=exp(gp.pred.a$LOGCPUE[,"mean"]+((summary(gp.modelBE
ST)$sigma^2)/2)),

"var"=exp(2*gp.pred.a$LOGCPUE[,"mean"]+gp.pred.a$LOGCPUE[,"var"])*(exp(gp.pred.a$L
OGCPUE[,"var"])-1))

## Index generation
## Multiply each estimate together and calculate the variance
varI=function(pmean,pvar,bmean,bvar){
  index_totvar = bvar*pvar + bvar*(pmean^2) + pvar*(bmean^2)  }
gI = data.frame('year' = gb.pred.a[,'year'], 'INDEX.EST' = gb.pred.a$BIN.CATCH[,'mean'] *
gp.trans$mean,'VARIANCE.FORM' =
varI(gp.trans$mean,gp.trans$var,gb.pred.a$BIN.CATCH[,'mean'],gb.pred.a$BIN.CATCH[,'var']
))
gI$SD = sqrt(gI$VARIANCE.FORM)
gI$CV = gI$SD/gI$INDEX.EST
gI$N=as.numeric(table(datag$year))
gI$SE=gI$SD/sqrt(gI$N)
gI$CV_mean=gI$SE/gI$INDEX.EST

write.csv(gI,"Finalized_stdindex_Guam_nocharter_031419.csv"), row.names = F)
```

2.2. CNMI

2.2.1. Calculating group proportions

```
######################
#Script to assign proportions by year to group species codes
```

```
#Six groups for CNMI (BBS-SBS):
#Jacks (misc), EE Juvenile.Jacks (same as Jacks), Bottom.Fish,
#Grouper (misc), Emperor, Snapper

#Five groups for CNMI (ComPurc):
#Jacks (misc), Bottom.Fish, Grouper (misc), Emperor, Snapper
######################

rm(list=ls())
loc="Location where the data are"

###############CNMI#############################
#Read in catch data

#BBS
cdatabbs=read.csv(paste0(loc,"CNMI\\C_BBS_SPC_R.csv"),header=T,na.strings="NULL")
cdatabbs$year=as.numeric(substr(cdatabbs$SPC_PK,2,5))
cdatabbs2017=read.csv(paste0(loc,"CNMI\\C_BBS_SPC2017.csv"),header=T)
cdatabbs2017$year=2017
cdatabbs=rbind(cdatabbs,cdatabbs2017[,-1]) #combine all years into one database
names(cdatabbs)[5]="KG_CAUGHT" #EVEN THOUGH CNMI BBS FIELDS SAY LBS,
VALUES ARE IN KGS!
#SBS
cdatasbs=read.csv(paste0(loc,"CNMI\\C_SBS_SPC_R.csv"),header=T,na.strings="NULL")
cdatasbs$year=substr(cdatasbs$SPC_PK,1,4)
cdatasbs2017=read.csv(paste0(loc,"CNMI\\C_SBS_SPC2017.csv"),header=T)
cdatasbs2017$year=2017
cdatasbs=rbind(cdatasbs,cdatasbs2017[,-1]) #combine all years into one database
#Commercial Purchase
cdatacp=read.csv(paste0(loc,"CNMI\\cnmi_ccl_commercial_16Apr18.csv"),header=T)
cdatacp$year=as.numeric(substr(cdatacp$INVOICE_DATE,1,4))
cdatacp$KGS_SOLD=cdatacp$TOT_LBS/2.20462 #LBS is properly labelled so convert to KG
cdatacp=cdatacp[!cdatacp$RESALE_F,] #remove RESALE_F=T records. These fish have
already been sold.

#Read in species key data
cgroups=read.csv(paste0(loc,"Species lists\\CNMI_BBS-SBS_GroupKey_final.csv"),header=T)
cspecies=read.csv(paste0(loc,"Species
lists\\CNMI_BFspecieslist_2018.csv"),header=T)[,c("species","spec_name")]
cgroupscp=read.csv(paste0(loc,"Species
lists\\CNMI_ComPurc_GroupKey_final.csv"),header=T)
cspeciescp=read.csv(paste0(loc,"Species
lists\\CNMI_ComPurc_BFspecieslist_2018.csv"),header=T)[,c("SPECIES_PK","SPECIES_NA
ME")]
```

```
#Determine amount of catch from managed species within groups that
#could include managed species, as stated at the beginning of this
#code script.
c.perc.bbs=c(2000:2017)
c.perc.sbs=data.frame("year"=2005:2017) #different approach since for sbs, catches of some
groups dont appear in each year
c.perc.cp=data.frame("year"=1983:2017) #some groups dont appear in each year
for(i in 5:10){
  c.X.codes=cgroups[cgroups[,i]==1,"SPECIES_PK"]
  #BBS

c.Xbbs.all=aggregate(KG_CAUGHT~year,data=subset(cdatabbs,SPECIES_FK%in%c.X.codes),
FUN=sum)

c.Xbbs.bmus=aggregate(KG_CAUGHT~year,data=subset(cdatabbs,SPECIES_FK%in%c.X.cod
es & SPECIES_FK%in%cspecies$species),FUN=sum)
  temp=merge(c.Xbbs.all,c.Xbbs.bmus,by="year",all.x=T,suffixes=c(".all",".bmus"))
  c.Xbbs.perc=temp$KG_CAUGHT.bmus/temp$KG_CAUGHT.all #For 2000-2017
  c.Xbbs.perc[is.na(c.Xbbs.perc)]=0 #set NAs to zero (since no BMUS in that year)
  c.perc.bbs=cbind(c.perc.bbs,c.Xbbs.perc)
  dimnames(c.perc.bbs)[[2]][i-3]=substr(names(cgroups)[i],6,nchar(names(cgroups)[i]))
  plot(x=c.perc.bbs[,1],y=c.perc.bbs[,i-3],type="l",main=paste(dimnames(c.perc.bbs)[[2]][i-
3],"BBS"),ylim=c(0,1))
  #SBS

c.Xsbs.all=aggregate(EXP_KGS~year,data=subset(cdatasbs,SPECIES_FK%in%c.X.codes),FUN
=sum)
  c.Xsbs.bmus=aggregate(EXP_KGS~year,data=subset(cdatasbs,SPECIES_FK%in%c.X.codes
& SPECIES_FK%in%cspecies$species),FUN=sum)
  temp=merge(c.Xsbs.all,c.Xsbs.bmus,by="year",all.x=T,suffixes=c(".all",".bmus"))
  temp[is.na(temp$EXP_KGS.bmus),"EXP_KGS.bmus"]=0 #set NAs to zero (since no BMUS
species in that year)
  temp2=merge(c.perc.sbs,temp,by="year",all.x=T) #Way to ensure all years have an entry
  c.Xsbs.perc=temp2$EXP_KGS.bmus/temp2$EXP_KGS.all #For 2005-2017
  c.Xsbs.perc[is.na(c.Xsbs.perc)]=sum(temp$EXP_KGS.bmus)/sum(temp$EXP_KGS.all) #set
NAs to be average across years (since no species from group reported separately in that year, so
use overall average)
  c.perc.sbs=cbind(c.perc.sbs,c.Xsbs.perc)
  dimnames(c.perc.sbs)[[2]][i-3]=substr(names(cgroups)[i],6,nchar(names(cgroups)[i]))
  plot(x=c.perc.sbs[,1],y=c.perc.sbs[,i-3],type="l",main=paste(dimnames(c.perc.sbs)[[2]][i-
3],"SBS"),ylim=c(0,1))
}
dimnames(c.perc.bbs)[[2]][1]="year"

#There is no catch of lyretail grouper, which is the only BMUS grouper. Thus
#catch in the group "grouper" based on our approach must be assigned to
```

```
#non-BMUS groupers, meaning the proportion is 0.
for(i in 6:10){
  #Commercial Purchase
  c.X.codes.cp=cgroupscp[cgroupscp[,i]==1,"SPECIES_PK"]

c.Xcp.all=aggregate(KGS_SOLD~year,data=subset(cdatacp,SPECIES_FK%in%c.X.codes.cp),F
UN=sum)
  if(i==8){ #For the group 'grouper'
    c.Xcp.bmus=data.frame("year"=1983:2017,"KGS_SOLD"=0)
  }else{

c.Xcp.bmus=aggregate(KGS_SOLD~year,data=subset(cdatacp,SPECIES_FK%in%c.X.codes.cp
& SPECIES_FK%in%cspeciescp$SPECIES_PK),FUN=sum)
  }
  temp=merge(c.Xcp.all,c.Xcp.bmus,by="year",all.x=T,suffixes=c(".all",".bmus"))
  temp[is.na(temp$KGS_SOLD.bmus),"KGS_SOLD.bmus"]=0 #set NAs to zero (since no
BMUS species in that year)
  temp2=merge(c.perc.cp,temp,by="year",all.x=T) #Way to ensure all years have an entry
  c.Xcp.perc=temp2$KGS_SOLD.bmus/temp2$KGS_SOLD.all #For 1983-2017
  c.Xcp.perc[is.na(c.Xcp.perc)]=sum(temp$KGS_SOLD.bmus)/sum(temp$KGS_SOLD.all) #set
NAs to be average across years (since no species from group reported separately in that year, so
use overall average)
  c.perc.cp=cbind(c.perc.cp,c.Xcp.perc)
  dimnames(c.perc.cp)[[2]][i-4]=substr(names(cgroupscp)[i],6,nchar(names(cgroupscp)[i]))
  plot(x=c.perc.cp[,1],y=c.perc.cp[,i-4],type="l",main=paste(dimnames(c.perc.cp)[[2]][i-
4],"ComPurc"),ylim=c(0,1))
}

###############Output final percentages############################
output_loc="Location where want to output"
write.csv(round(c.perc.bbs,3),paste0(output_loc,"SpeciesGroups\\Cbbs_GroupProportions.csv"),
row.names=F)
write.csv(round(c.perc.sbs,3),paste0(output_loc,"SpeciesGroups\\Csbs_GroupProportions.csv"),r
ow.names=F)
write.csv(round(c.perc.cp,3),paste0(output_loc,"SpeciesGroups\\CComPurc_GroupProportions.c
sv"),row.names=F)
```

2.2.2. Catch data filtering and calculation

```
###
#Calculated total catch used in the model from all three catch datasets.
#Applies previously calculated group proportions for each dataset.
###

rm(list=ls())
```

```
library(dplyr)
library(tidyr)
loc="Folder where datasets reside"

###############CNMI#############################
#BMUS species/groups
csp <- read.csv(paste0(loc,"CNMI_BFspecieslist_2018.csv"),header=T)
csp.cp <- read.csv(paste0(loc,"CNMI_ComPurc_BFspecieslist_2018.csv"),header=T)

#Pproportions of species groups that are BMUS
cspg.prop.bbs=read.csv(paste0(loc,"Cbbs_GroupProportions.csv"),header=T)
cspg.prop.sbs=read.csv(paste0(loc,"Csbs_GroupProportions.csv"),header=T)
cspg.prop.cp=read.csv(paste0(loc,"CComPurc_GroupProportions.csv"),header=T)

###
#BBS data
###
#Combined 2017 data with rest of data. Ensure NULLs are set as NAs so that values
#are read in as numbers.
cbbs=rbind(read.csv(paste0(loc,"C_BBS_SPC_R.csv"),header=T,na.strings="NULL"),
        read.csv(paste0(loc,"C_BBS_SPC2017.csv"),header=T)[,-1])
names(cbbs)[5]="KGS_CAUGHT" #Values are in KGS despite field name of LBS
cbbs$year=as.numeric(substr(cbbs$SPC_PK,2,5))

#Sum catches by year and species
cbbs.yrsums = cbbs %>%
  filter(SPECIES_FK %in% csp$species) %>%
  select(SPECIES_FK,KGS_CAUGHT,year) %>%
  group_by(year,SPECIES_FK) %>%
  summarise(total = sum(KGS_CAUGHT,na.rm=T))
#table the catches by year (rows) and species codes (columns)
cbbs.table = spread(cbbs.yrsums,"SPECIES_FK","total")

#Match the columns in cbbs.table with associated species groups so as
#to apply proportion. Then....
group.cols=match(csp$species,as.numeric(names(cbbs.table)[-1]))+1
cbind(csp[,c("species","spec_name")],group.cols)
#....multiply group catch by the corresponding proportions of that group that are BMUS
cbbs.table[,group.cols[14:19]]=cbbs.table[,group.cols[14:19]]*cspg.prop.bbs[,-1]

##
#Add variance now for BBS only (2000-2017).
##
cvar.bbs=read.csv(paste0(loc,"C_BBS_SPCbootstrap_31May18.csv"),header=T)
cvar.bbs=cvar.bbs[,c("year","species_fk","common_name","catch","var_catch","proportion_zero
_catch")]
```

```
#Merge with yrsum catches.
cbbs.yrsums.wvar=merge(cbbs.yrsums,cvar.bbs,by.x=c("year","SPECIES_FK"),by.y=c("year","
species_fk"))

#Repeat process for applying group proportions to variance as for catch above
cbbs.table.wvar =
spread(cbbs.yrsums.wvar[,c("year","SPECIES_FK","var_catch")],"SPECIES_FK","var_catch")
group.cols.wvar=match(csp$species,as.numeric(names(cbbs.table.wvar)[-1]))+1
cbind(csp[,c("species","spec_name")],group.cols)
cbbs.table.wvar[,group.cols.wvar[14:19][!is.na(group.cols.wvar[14:19])]]=cbbs.table.wvar[,grou
p.cols.wvar[14:19][!is.na(group.cols.wvar[14:19])]]*((cspg.prop.bbs[,-
1][,which(!is.na(group.cols.wvar[14:19]))])*(cspg.prop.bbs[,-
1][,which(!is.na(group.cols.wvar[14:19]))]))


###
#SBS data
###
#Combined 2017 data with rest of data. Ensure NULLs are set as NAs so that values
#are read in as numbers.
csbs=rbind(read.csv(paste0(loc,"C_SBS_SPC_R.csv"),header=T,na.strings="NULL"),
        read.csv(paste0(loc,"C_SBS_SPC2017.csv"),header=T)[,-1])
csbs$year=as.numeric(substr(csbs$SPC_PK,1,4))

#Sum catches by year and species
csbs.yrsums = csbs %>%
  filter(SPECIES_FK %in% csp$species) %>%
  select(SPECIES_FK,EXP_KGS,year) %>%
  group_by(year,SPECIES_FK) %>%
  summarise(total = sum(EXP_KGS,na.rm=T))
#table the catches by year (rows) and species codes (columns)
csbs.table = spread(csbs.yrsums,"SPECIES_FK","total")

#Match columns in tabled catches with groups and then...
group.cols=match(csp$species,as.numeric(names(csbs.table)[-1]))+1
cbind(csp[,c("species","spec_name")],group.cols)
#....multiply group catch by the corresponding proportions of that group that are BMUS
csbs.table[,group.cols[14:19][!is.na(group.cols[14:19])]]=csbs.table[,group.cols[14:19][!is.na(gr
oup.cols[14:19])]]*(cspg.prop.sbs[,-1][,which(!is.na(group.cols[14:19]))])


###
#Commercial Purchase Data
###
ccp=read.csv(paste0(loc,"cnmi_ccl_commercial_16Apr18.csv"),header=T)
ccp$year=as.numeric(substr(ccp$INVOICE_DATE,1,4))
```

```
ccp$KGS_SOLD=ccp$LBS_SOLD/2.20462 #LBS is properly labelled so convert to KG
ccp1=ccp[!ccp$RESALE_F,] #remove RESALE_F=T records. These fish have already been
sold.

#Sum catches by year and species
ccp.yrsums = ccp1 %>%
  filter(SPECIES_FK %in% csp.cp$SPECIES_PK) %>%
  select(SPECIES_FK,KGS_SOLD,year) %>%
  group_by(year,SPECIES_FK) %>%
  summarise(total = sum(KGS_SOLD,na.rm=T))
#table the catches by year (rows) and species codes (columns)
ccp.table = spread(ccp.yrsums,"SPECIES_FK","total")

#Match columns in tabled catches with groups and then...
group.cols=match(csp.cp$SPECIES_PK,as.numeric(names(ccp.table)[-1]))+1
cbind(csp.cp[,c("SPECIES_PK","SPECIES_NAME")],group.cols)
#....multiply group catch by the corresponding proportions of that group that are BMUS
ccp.table[,group.cols[c(1,4,7,15,16)][!is.na(group.cols[c(1,4,7,15,16)])]]=ccp.table[,group.cols[c(
1,4,7,15,16)][!is.na(group.cols[c(1,4,7,15,16)])]]*(cspg.prop.cp[,-
1][,which(!is.na(group.cols[c(1,4,7,15,16)]))])



###
#Final catch by year
###
cbbs.TOTC=data.frame("year"=cbbs.table[,1],"cbbs.kg"=rowSums(cbbs.table[,-
1],na.rm=T),"cbbs.kg.var"=rowSums(cbbs.table.wvar[,-1],na.rm=T))
csbs.TOTC=data.frame("year"=csbs.table[,1],"csbs.kg"=rowSums(csbs.table[,-1],na.rm=T))
ccp.TOTC=data.frame("year"=ccp.table[,1],"ccp.kg"=rowSums(ccp.table[,-1],na.rm=T))
c.TOTC=merge(merge(cbbs.TOTC,csbs.TOTC,by="year",all=T),ccp.TOTC,by="year",all=T)
write.csv(c.TOTC,paste0(loc,"TotalCatch_CNMI_init.csv"),row.names=F)



###############Algorithms for combining missing data in
catches############################
#Read in all files
catch.folder="Folder where catch data by dataset are located"
gCatch=read.csv(paste0(catch.folder,"TotalCatch_Guam_init.csv"),header=T)
cCatch=read.csv(paste0(catch.folder,"TotalCatch_CNMI_init.csv"),header=T)

#Combine files into single variable
allCatch=round(merge(gCatch,cCatch,by="year",all=T),1)

#Algorithm for filling in missing values
#0. Set dataset to start in 1982 (remove the first three years from G ComPurc)
```

```
allCatch=allCatch[allCatch$year%in%c(1982:2017),]
#1. For filling missing SBS values, use overall average SBS catch.
areNAc=is.na(allCatch$csbs.kg)!=is.na(allCatch$cbbs.kg)
allCatch$csbs.kg[areNAc]=mean(allCatch$csbs.kg,na.rm=T)
#2. Sum together SBS and BBS data.
allCatch$cTOT.kg=round(allCatch$cbbs.kg+allCatch$csbs.kg,0)
#3. Take max of SBS+BBS and original ComPurc data
cmax=which(allCatch$cTOT.kg<allCatch$ccp.kg)
allCatch$cTOT.kg[cmax]=round(allCatch$ccp.kg[cmax],0) #replace value with that from
ComPurc
#4. Add in CVs for BBS data for final file output
allCatch$cCV=round(sqrt(allCatch$cbbs.kg.var)/allCatch$cbbs.kg,2)

##
#Output final datasets for use in the assessment
write.csv(allCatch[!is.na(allCatch$cTOT.kg),c("year","cTOT.kg","cCV")],paste0(loc,"TotalCatc
h_CNMI_FINAL.csv"),row.names=F)
```

2.2.3. CPUE data filtering

```
###
#Steps to obtain nominal CPUE dataset
###

rm(list=ls())
library(dplyr)
library(tidyr)
loc="Folder where datasets reside"

############## CNMI ############################

######
#READ IN DATA SETUP FILES
######

#BMUS species/groups
csp <- read.csv(paste0(loc,"CNMI_BFspecieslist_2018.csv"),header=T)
#Interview data
cint.bbs=read.csv(paste0(loc,"cnmi_bbs_interview_10Apr18.csv"),header=T)
cint.bbs$SAMPLE_DATE=as.Date(cint.bbs$SAMPLE_DATE)
cint.bbs$year=as.numeric(format(cint.bbs$SAMPLE_DATE,"%Y"))
cint.bbs$month=as.numeric(format(cint.bbs$SAMPLE_DATE,"%m"))
#Proportions of species groups that are BMUS
c.proptable.bbs=read.csv(paste0(loc,"Cbbs_GroupProportions.csv"),header=T)
```

```
######
#CPUE CALCULATIONS SETUP
######

##
#Add group proportions
##
#Reduce group proportion table to row-column key of proportions, with
#species key instead of species name as column header
colnames(c.proptable.bbs)=c("year",as.character(csp$species[14:19]))
c.propkey.bbs=gather(c.proptable.bbs,key="SPECIES_FK",value="prop",-1)

#Add in group proportionality as the value from the group proportion
#key for each group and year. For non-group species, set proportion to 1
cint1.bbs=merge(cint.bbs,c.propkey.bbs,by=c("year","SPECIES_FK"),all.x=T)
cint1.bbs[is.na(cint1.bbs$prop),"prop"]=1

#Multiply proportion by EST_KGS, which is the species specific total
cint1.bbs$est_kgs=cint1.bbs$EST_KGS*cint1.bbs$prop

##
#Calculate proportion of BMUS by interview
##
#Assign proportion of BMUS by trip. Since weights for groups are already
#allocated to BMUS/nonBMUS, include group keys too.
#Must first reduce data into unique combinations of interview, species, and pounds.
temp.bbs=unique(cint1.bbs[,c("INTERVIEW_PK","SPECIES_FK","EST_KGS","est_kgs")])
bbs.all=aggregate(EST_KGS~INTERVIEW_PK,data=temp.bbs,FUN=sum)
bbs.bmus=aggregate(est_kgs~INTERVIEW_PK,data=temp.bbs[temp.bbs$SPECIES_FK%in%cs
p$species,],FUN=sum)
bbs.perctrip=merge(bbs.all,bbs.bmus,by="INTERVIEW_PK",all.x=T)
names(bbs.perctrip)[2:3]=c("EST_KGS.all","est_kgs.bmus")
bbs.perctrip[is.na(bbs.perctrip$est_kgs.bmus),"est_kgs.bmus"]=0
bbs.perctrip$percBMUS=bbs.perctrip$est_kgs.bmus/bbs.perctrip$EST_KGS.all
cint2.bbs=merge(cint1.bbs,bbs.perctrip,by="INTERVIEW_PK",all.x=T)


##
#Add effort and cpue
##
cint2.bbs$effort=cint2.bbs$NUM_GEAR*cint2.bbs$HOURS_FISHED
cint2.bbs$cpue=cint2.bbs$est_kgs.bmus/cint2.bbs$effort

##
#Filter dataset. Bottom method, vessels that caught BMUS, data with valid fields
```

```
##
#Reduce dataset to pertinent fields
cint3.bbs=cint2.bbs[,c("INTERVIEW_PK","year","SPECIES_FK","SAMPLE_DATE","TYPE_
OF_DAY",

"INTERVIEW_TIME","AREA_FK","METHOD_FK","METHOD_NAME","DEPTH",

"WEATHER_NAME","WIND_DIRECTION","WIND_SPEED","HOURS_FISHED",
                "NUM_FISHER","NUM_GEAR","VESSEL_NAME","PORT_NAME",
                "TOT_EST_KGS","TOT_SPECIES","LUNAR_DAY","EST_KGS",
                "prop","est_kgs","EST_KGS.all","est_kgs.bmus","percBMUS",
                "effort","cpue","month","CHARTER_F")]

#Remove SPECIES_FK, est_kgs, prop, and EST_KGS and reduce to unique trips
cint4.bbs=unique(cint3.bbs[,-c(3,22,23,24)])

#Limit interviews to be those on Method "Bottom", METHOD_FK=2
cint4.bbs=cint4.bbs[cint4.bbs$METHOD_NAME=="BOTTOM",]

#Remove the 51 vessels (and their 58 interviews) that never caught any BMUS
tempvc=aggregate(est_kgs.bmus~VESSEL_NAME,data=cint4.bbs,FUN=sum,na.rm=T)
zero.vessc=tempvc[tempvc$est_kgs.bmus==0,"VESSEL_NAME"]
cint5.bbs=cint4.bbs[!cint4.bbs$VESSEL_NAME%in%zero.vessc,]

#Use fields: type of day, area, depth category, vessel name, and charter.
#Remove any interviews without records of these.
cint6.bbs=cint5.bbs[,c("year","month","cpue","effort","TYPE_OF_DAY","AREA_FK","DEPT
H","VESSEL_NAME","CHARTER_F")]
cint6.bbs=cint6.bbs[is.finite(cint6.bbs$cpue),]
cint6.bbs=cint6.bbs[!is.na(cint6.bbs$VESSEL_NAME),]
cint6.bbs[is.na(cint6.bbs$DEPTH) | cint6.bbs$DEPTH==2,"DEPTH"]="U"
cint6.bbs=droplevels(cint6.bbs)

##
#Output dataset
##
write.csv(cint6.bbs,paste0(loc,"FinalizedCNMI_intCPUE_forStandardization_080318.csv"),row.
names=F)
```

2.2.4. CPUE standardization

```
###
#Steps to perform CPUE standardization
###
```

```
rm(list=ls())
library(lme4)
library(MASS)
library(statmod)
loc="Folder where nominal dataset resides"

#<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>#
############### CNMI ############################
#<<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>#

datac=read.csv(paste0(loc,"FinalizedCNMI_intCPUE_forStandardization_080318.csv"),header=
T)

#Turn year and area and month into factors
datac$year=as.factor(datac$year)
datac$month=as.factor(datac$month)
datac$AREA_FK=as.factor(datac$AREA_FK)
#Add 0-1 indicator variable
datac$z=0
datac[datac$cpue>0,"z"]=1
#Remove the 11 interviews without an area code
datac=datac[!(datac$AREA_FK==0),]
datac$AREA_FK=droplevels(datac$AREA_FK)
#Remove charters (CHARTER_F equals T)
datac=datac[!(datac$CHARTER_F),]
#Set up positive only database
datac.pos=datac[datac$cpue>0,]


####
#BERNOULLI
####

#Full model
cb.model0=glm(z~year+month+TYPE_OF_DAY+DEPTH+AREA_FK,family=binomial,data=d
atac)
cb.model0a=glm(z~year+month+TYPE_OF_DAY+AREA_FK,family=binomial,data=datac)
cb.model0b=glm(z~year+month+DEPTH+AREA_FK,family=binomial,data=datac)
cb.model0c=glm(z~year+TYPE_OF_DAY+DEPTH+AREA_FK,family=binomial,data=datac)
cb.model0d=glm(z~year+month+TYPE_OF_DAY+DEPTH,family=binomial,data=datac)
cbAIC0=rbind("Full"=extractAIC(cb.model0),"-D"=extractAIC(cb.model0a),
        "-TD"=extractAIC(cb.model0b),"-M"=extractAIC(cb.model0c),
        "-A"=extractAIC(cb.model0d))
cbAIC0=cbind(cbAIC0,(cbAIC0[,2]-cbAIC0[1,2])) #if values is at least +2, then remove

which.min(cbAIC0[,3]) #Remove area (0d)
```

```
#Step 1 - removed area
cb.model1a=glm(z~year+month+TYPE_OF_DAY,family=binomial,data=datac)
cb.model1b=glm(z~year+month+DEPTH,family=binomial,data=datac)
cb.model1c=glm(z~year+TYPE_OF_DAY+DEPTH,family=binomial,data=datac)
cbAIC1=rbind("Full-A"=extractAIC(cb.model0d),"-D-A"=extractAIC(cb.model1a),
        "-TD-A"=extractAIC(cb.model1b),"-M-A"=extractAIC(cb.model1c))
cbAIC1=cbind(cbAIC1,(cbAIC1[,2]-cbAIC1[1,2])) #if values is at least +2, then remove

which.min(cbAIC1[,3]) #Remove month (1c)

#Step 2 - removed month and area
cb.model2a=glm(z~year+TYPE_OF_DAY,family=binomial,data=datac)
cb.model2b=glm(z~year+DEPTH,family=binomial,data=datac)
cbAIC2=rbind("FUll-M-A"=extractAIC(cb.model1c),"-D-M-A"=extractAIC(cb.model2a),
        "-TD-M-A"=extractAIC(cb.model2b))
cbAIC2=cbind(cbAIC2,(cbAIC2[,2]-cbAIC2[1,2])) #if values is at least +2, then remove
cbAIC2

cb.modelBEST = cb.model1c #BEST MODEL CONTAINS ONLY DEPTH AND TYPE OF
DAY


####
#LOGNORMAL
####
#Use year, month, type of day, depth, area, vessel name

#Full model, for random effects
cp.model0=lmer(log(cpue)~year+(1|VESSEL_NAME)+month+AREA_FK+TYPE_OF_DAY+
DEPTH,REML=F,data=datac.pos)
cp.model0a=glm(log(cpue)~year+month+AREA_FK+TYPE_OF_DAY+DEPTH,data=datac.pos
)

AIC(cp.model0) - AIC(cp.model0a) #Full model has lower AIC. Keep Vessel

#Full model, for fixed effects
cp.model1a=lmer(log(cpue)~year+(1|VESSEL_NAME)+month+AREA_FK+TYPE_OF_DAY,
REML=F,data=datac.pos)
cp.model1b=lmer(log(cpue)~year+(1|VESSEL_NAME)+month+AREA_FK+DEPTH,REML=F,
data=datac.pos)
cp.model1c=lmer(log(cpue)~year+(1|VESSEL_NAME)+month+TYPE_OF_DAY+DEPTH,RE
ML=F,data=datac.pos)
cp.model1d=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+TYPE_OF_DAY+DEPTH
,REML=F,data=datac.pos)
```

```
cpAIC1=rbind("Full"=extractAIC(cp.model0),"-D"=extractAIC(cp.model1a),"-
TD"=extractAIC(cp.model1b),
        "-A"=extractAIC(cp.model1c),"-M"=extractAIC(cp.model1d))
cpAIC1=cbind(cpAIC1,(cpAIC1[,2]-cpAIC1[1,2])) #if values is at least +2, then remove

which.min(cpAIC1[,3]) #Remove month (1d)

#Step 2 - removed month
cp.model2a=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+TYPE_OF_DAY,REML=
F,data=datac.pos)
cp.model2b=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+DEPTH,REML=F,data=da
tac.pos)
cp.model2c=lmer(log(cpue)~year+(1|VESSEL_NAME)+TYPE_OF_DAY+DEPTH,REML=F,d
ata=datac.pos)
cpAIC2=rbind("Full-M"=extractAIC(cp.model1d),"-M-D"=extractAIC(cp.model2a),"-M-
TD"=extractAIC(cp.model2b),
        "-M-A"=extractAIC(cp.model2c))
cpAIC2=cbind(cpAIC2,(cpAIC2[,2]-cpAIC2[1,2])) #if values is at least +2, then remove
cpAIC2

#BEST MODEL CONTAINS AREA AND DEPTH AND VESSEL AND TYPE_OF_DAY
cp.modelBEST=lmer(log(cpue)~year+(1|VESSEL_NAME)+AREA_FK+DEPTH+TYPE_OF_D
AY,REML=T,data=datac.pos)


######## Save Rdata output ##########
save(datac,datag,gb.modelBEST,gp.modelBEST,cb.modelBEST,cp.modelBEST,file=paste0(loc,
"best_mods_GC_nocharter_080618.RData"))
```

2.2.5. CPUE index calculation

```
###
#Steps to calculate standardized index
###

rm(list=ls())
loc="Folder where standardized model scripts reside"
load(file=paste0(loc,"best_mods_GC_nocharter_080618.RData"))

datac.pos=datac[datac$cpue>0,]

#<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>#
############## CNMI ############################
#<<<<<<<<<<<<<<<<<<>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>#
```

```
## Extract predicted values for Bernoulli process ("b")
cb.pred = data.frame('BIN.CATCH' = predict(cb.modelBEST, type = 'response'),'year' =
datac[,'year'])
## get means, sd and variance for each year
cb.pred.a = aggregate(BIN.CATCH ~ year, cb.pred,  FUN = function(x) c(mean = mean(x), sd =
sd(x), var = var(x)))

## Extract predicted values for Positive process ("p")
cp.pred = data.frame('LOGCPUE' = predict(cp.modelBEST), 'year' = datac.pos[,'year'])
## get means and sd for each year
cp.pred.a = aggregate(LOGCPUE ~ year, cp.pred, function(x) c(mean = mean(x), sd = sd(x), var
= var(x)))
## Backtransform positive process using dispersion
cp.trans =
list("year"=c(2000:2017),"mean"=exp(cp.pred.a$LOGCPUE[,"mean"]+((summary(cp.modelBE
ST)$sigma^2)/2)),

"var"=exp(2*cp.pred.a$LOGCPUE[,"mean"]+cp.pred.a$LOGCPUE[,"var"])*(exp(cp.pred.a$LO
GCPUE[,"var"])-1))

## Index generation
## Multiply each estimate together and calculate the variance
varI=function(pmean,pvar,bmean,bvar){
  index_totvar = bvar*pvar + bvar*(pmean^2) + pvar*(bmean^2)  }
cI = data.frame('year' = cb.pred.a[,'year'], 'INDEX.EST' = cb.pred.a$BIN.CATCH[,'mean'] *
cp.trans$mean, 'VARIANCE.FORM' =
varI(cp.trans$mean,cp.trans$var,cb.pred.a$BIN.CATCH[,'mean'],cb.pred.a$BIN.CATCH[,'var'])
)
cI$SD = sqrt(cI$VARIANCE.FORM)
cI$CV = cI$SD/cI$INDEX.EST
cI$N=as.numeric(table(datac$year))
cI$SE=cI$SD/sqrt(cI$N)
cI$CV_mean=cI$SE/cI$INDEX.EST

write.csv(cI,"Finalized_stdindex_CNMI_nocharter_031419.csv", row.names = F)
```

2.3. American Samoa

2.3.1. Calculating group proportions

```
######################
#Script to assign proportions by year to group species codes
#and apply that proportion of the catch for the group to
#managed species
```

```
#Six groups for AmSam (although 9 total will appear below):
#Trevally, Jacks (same as Trevally), Bottomfishes, Groupers (same as inshore groupers),
#Deep snappers (same as inshore snappers), Pristipomoides/Etelis, Emporers
####################

rm(list=ls())
loc="Location where the data are"

##############AmSam############################
#Read in catch data
####
#NOTE: AmSam values are in Lbs.
####
#BBS
adatabbs=read.csv(paste0(loc,"AmSam\\A_BBS_SPC_R.csv"),header=T,na.strings="NULL")
adatabbs$year=as.numeric(substr(adatabbs$SPC_PK,2,5))
adatabbs2017=read.csv(paste0(loc,"AmSam\\A_BBS_SPC2017.csv"),header=T)
adatabbs2017$year=2017
adatabbs=rbind(adatabbs,adatabbs2017[,-1]) #combine all years into one database

#SBS
adatasbs=read.csv(paste0(loc,"AmSam\\A_SBS_SPC_R.csv"),header=T,na.strings="NULL")

#the years are absent, but appear in this file:
adatasbs_exp=read.csv(paste0(loc,"AmSam\\A_SBS_EXP_R.csv"),header=T,na.strings="NULL
")

dat<-merge(adatasbs,adatasbs_exp) # merge the data sets to provide year for the spc values

adatasbs2017=read.csv(paste0(location,"AmSam\\A_SBS_SPC2017.csv"),header=T)
adatasbs2017$year=2017

# "smart bind"
library(plyr)
adatasbs=rbind.fill(dat,adatasbs2017[,-1]) #combine all years into one database

#fill in 2017 YEAR column
adatasbs$YEAR[3486:3608]<-2017

#Read in species key data
agroups=read.csv(paste0(loc,"Species lists\\AmSam_BBS-
SBS_GroupKey_final.csv"),header=T)
aspecies=read.csv(paste0(loc,"Species
lists\\AmSam_BFspecieslist_2018.csv"),header=T)[,c("species","spec_name")]

#Determine amount of catch from managed species within groups that
```

```
#could include managed species, as stated at the beginning of this
#code script.
a.perc.bbs=c(1986:2017)
a.perc.sbs=data.frame("YEAR"=1990:2017) #different approach since for sbs, we are missing
1997-2004
for(i in 7:15){
  a.X.codes=agroups[agroups[,i]==1,"SPECIES_PK"]

   #BBS

a.Xbbs.all=aggregate(LBS_CAUGHT~year,data=subset(adatabbs,SPECIES_FK%in%a.X.codes
),FUN=sum)

a.Xbbs.bmus=aggregate(LBS_CAUGHT~year,data=subset(adatabbs,SPECIES_FK%in%a.X.co
des & SPECIES_FK%in%aspecies$species),FUN=sum)
  #missing years with Lethrinids requires inserting 0s for years that "aggregate" omits
  if (i==13){
    a.Xbbs.allv<-c(0,a.Xbbs.all[,2]) #1986 is empty
    a.Xbbs.bmusv<-c(0,0,a.Xbbs.bmus[1:10,2],0,a.Xbbs.bmus[11,2],0,a.Xbbs.bmus[12:28,2])
  }
  if (i!=13){
  a.Xbbs.allv<-a.Xbbs.all$LBS_CAUGHT
  a.Xbbs.bmusv<-a.Xbbs.bmus$LBS_CAUGHT
  }
  a.Xbbs.perc=a.Xbbs.bmusv/a.Xbbs.allv #For 1986-2017
  a.perc.bbs=cbind(a.perc.bbs,a.Xbbs.perc)
  dimnames(a.perc.bbs)[[2]][i-5]=substr(names(agroups)[i],1,nchar(names(agroups)[i]))
  plot(x=a.perc.bbs[,1],y=a.perc.bbs[,i-5],type="l",main=paste(dimnames(a.perc.bbs)[[2]][i-
5],"BBS"),ylim=c(0,1))

  #SBS

a.Xsbs.all=aggregate(EXP_LBS~YEAR,data=subset(adatasbs,SPECIES_FK%in%a.X.codes),F
UN=sum)
  a.Xsbs.bmus=a.Xsbs.all #no records for BMUS trevallies/jacks, hence they are zero. run the
code for other species
  a.Xsbs.bmus[,2]<-0
  if(i>8){
  a.Xsbs.bmus=aggregate(EXP_LBS~YEAR,data=subset(adatasbs,SPECIES_FK%in%a.X.codes
& SPECIES_FK%in%aspecies$species),FUN=sum)
  }

  temp=merge(a.Xsbs.all,a.Xsbs.bmus,by="YEAR",all.x=T,suffixes=c(".all",".bmus"))
  temp[is.na(temp)]=0 #set NAs to zero (since no BMUS in that year)
  temp2=merge(a.perc.sbs,temp,by="YEAR",all.x=T) #Way to ensure all years have an entry
  a.Xsbs.perc=temp2$EXP_LBS.bmus/temp2$EXP_LBS.all #For 1990-2017
```

```
  a.perc.sbs=cbind(a.perc.sbs,a.Xsbs.perc)
  dimnames(a.perc.sbs)[[2]][i-5]=substr(names(agroups)[i],1,nchar(names(agroups)[i]))
  plot(x=a.perc.sbs[,1],y=a.perc.sbs[,i-5],type="b",main=paste(dimnames(a.perc.sbs)[[2]][i-
5],"SBS"),ylim=c(0,1))
}

dimnames(a.perc.bbs)[[2]][1]="year"
a.perc.bbs[1,8] <- 0 #convert the Nan to a zero

dimnames(a.perc.sbs)[[2]][1]="year"

output_loc="Location for output"
write.csv(round(a.perc.bbs,3),paste0(output_loc,"Abbs_GroupProportions.csv"),row.names=F)
write.csv(round(a.perc.sbs,3),paste0(output_loc,"Asbs_GroupProportions.csv"),row.names=F)


#Commercial purchase
data#############################################################################
##############
#1990 - 2017

acp<-read.csv(paste0(loc,"\\AmSam\\asam_acl_commercial_14May18.csv"),header=T)
acp$year=as.numeric(substr(acp$INVOICE_DATE,1,4))
acp<-acp[(acp$IMPORTED_F==FALSE),] #get rid of the imports


a.perc.cp=data.frame("year"=1990:2017) #some groups dont appear in each year

for(i in 7:15){
  #Commercial Purchase
  a.X.codes=agroups[agroups[,i]==1,"SPECIES_PK"]

a.Xcp.all=aggregate(SOLD_LBS~year,data=subset(acp,SPECIES_FK%in%a.X.codes),FUN=su
m)

  a.Xcp.bmus=aggregate(SOLD_LBS~year,data=subset(acp,SPECIES_FK%in%a.X.codes &
SPECIES_FK%in%aspecies$species),FUN=sum)

  temp=merge(a.Xcp.all,a.Xcp.bmus,by="year",all.x=T,suffixes=c(".all",".bmus"))
  temp[is.na(temp$SOLD_LBS.bmus),"SOLD_LBS.bmus"]=0 #set NAs to zero (since no BMUS
species in that year)
  temp2=merge(a.perc.cp,temp,by="year",all.x=T) #Way to ensure all years have an entry
  a.Xcp.perc=temp2$SOLD_LBS.bmus/temp2$SOLD_LBS.all
  a.Xcp.perc[is.na(a.Xcp.perc)]=sum(temp$SOLD_LBS.bmus)/sum(temp$SOLD_LBS.all) #set
NAs to be average across years (since no species from group reported separately in that year, so
use overall average)
```

```
  a.perc.cp=cbind(a.perc.cp,a.Xcp.perc)
  dimnames(a.perc.cp)[[2]][i-5]=substr(names(agroups)[i],1,nchar(names(agroups)[i]))
  plot(x=a.perc.cp[,1],y=a.perc.cp[,i-5],type="l",main=paste(dimnames(a.perc.cp)[[2]][i-
5],"ComPurc"),ylim=c(0,1))
}

write.csv(round(a.perc.cp,3),paste0(output_loc,"Acp_GroupProportions.csv"),row.names=F)
```

2.3.2. Catch data filtering and calculation

```
###
#Calculated total catch used in the model from all three catch datasets.
#Applies previously calculated group proportions for each dataset.
###

rm(list=ls())
library(dplyr)
library(tidyr)
loc="Folder where datasets reside"

##############AmSam############################
#BMUS species/groups
asp <- read.csv(paste0(loc,"Raw data\\Species lists\\AS_BFspecieslist_2018.csv"),header=T)


#Proportions of species groups that are BMUS
aspg.prop.bbs=read.csv(paste0(loc,"Data processing\\Abbs_GroupProportions.csv"),header=T)
aspg.prop.sbs=read.csv(paste0(loc,"Data processing\\Asbs_GroupProportions.csv"),header=T)
aspg.prop.cp=read.csv(paste0(loc,"Data processing\\Acp_GroupProportions.csv"),header=T)


###
#BBS data
###

#Combined 2017 data with rest of data. Ensure NULLs are set as NAs so that values
#are read in as numbers.
abbs=rbind(read.csv(paste0(loc,"Raw
data\\AmSam\\A_BBS_SPC_R.csv"),header=T,na.strings="NULL"),
        read.csv(paste0(loc,"Raw data\\AmSam\\A_BBS_SPC2017.csv"),header=T)[,-1])

sort(unique(abbs$SPECIES_FK)) #243 is not a real species.  should be Aphareus rutilans
abbs$SPECIES_FK[abbs$SPECIES_FK==243]<-247

abbs$year=as.numeric(substr(abbs$SPC_PK,2,5))
```

```
#Sum catches by year and species
abbs.yrsums = abbs %>%
  filter(SPECIES_FK %in% asp$species) %>%
  select(SPECIES_FK,LBS_CAUGHT,year) %>%
  group_by(year,SPECIES_FK) %>%
  summarise(total = sum(LBS_CAUGHT,na.rm=T))
#table the catches by year (rows) and species codes (columns)
abbs.table = spread(abbs.yrsums,"SPECIES_FK","total")
#Ensure all years are included in table. They are
abbs.table$year
#pull out catch from individual BMUS species
ind.bmus.bbs=abbs.table[,colnames(abbs.table)%in%asp$species[1:11]]

#Match the columns in abbs.table associated with the species groups. Need to remove
#the first column (year) in abbs.table but then have to add it back to get
#numbering right. Then....
group.cols=match(asp$species,as.numeric(names(abbs.table)[-1]))+1
cbind(asp[,c("species","spec_name")],group.cols)
#....multiply group catch by the corresponding proportions of that group that are BMUS
#pull out catch from species groups assumed to be BMUS
group.bmus.bbs=abbs.table[,group.cols[12:20][!is.na(group.cols[12:20])]]*(aspg.prop.bbs[,-
1][,which(!is.na(group.cols[12:20]))])
#Need to used only the groups that are in the data. Thus take only these columns from
aspg.prop.bbs
abbs.table[,group.cols[12:20][!is.na(group.cols[12:20])]]=abbs.table[,group.cols[12:20][!is.na(gr
oup.cols[12:20])]]*(aspg.prop.bbs[,-1][,which(!is.na(group.cols[12:20]))])



##
#Add variance now for BBS only
##
abbs=read.csv(paste0(loc,"Raw
data\\AmSam\\A_BBS_SPCbootstrap31May18.csv"),header=T,na.strings="NULL")


#if species is a group, multiply the variance by the square of the group proportion
#Reassign the species (i.e. column) names as numbers only
groups<-c(109,110,200,210,230,240,260,380,390)
names(aspg.prop.bbs)<-c("year",groups)

Final_var<-vector(length=nrow(abbs))
for (i in 1:nrow(abbs)){
  if(abbs[i,]$species_pk==210 | abbs[i,]$species_pk==109 | abbs[i,]$species_pk==200 |
abbs[i,]$species_pk==230 | abbs[i,]$species_pk==260 | abbs[i,]$species_pk==390|
```

```
    abbs[i,]$species_pk==110| abbs[i,]$species_pk==380| abbs[i,]$species_pk==240){
   y<-paste(abbs[i,]$species_pk)
   Final_var[i]<-abbs[i,]$var_catch*(aspg.prop.bbs[((abbs[i,]$year)-1985),y]^2)
  } else{
   Final_var[i]<-abbs[i,]$var_catch
  }
} #wait a minute or two...


new_bbs<-cbind(abbs,Final_var)

#243 is not a real species.  should be Aphareus rutilans
abbs$SPECIES_FK[new_bbs$SPECIES_FK==243]<-247

#Sum variances by year and species
asbs.yrsums = new_bbs %>%
  filter(species_pk %in% asp$species) %>%
  select(species_pk,Final_var,year) %>%
  group_by(year,species_pk) %>%
  summarise(total = sum(Final_var,na.rm=T))
#table the catches by year (rows) and species codes (columns)
abbs.table = spread(asbs.yrsums,"species_pk","total")
#Ensure all years are included in table. They are
abbs.table$year

#sum among the species within a year
abbs.TOTV=data.frame("year"=abbs.table[,1],"totvar"=rowSums(abbs.table[,-1],na.rm=T))


################################################################################
##########################################

################################################################################
#################################
#SBS data
###

#SBS
adatasbs=read.csv(paste0(loc,"Raw
data\\AmSam\\A_SBS_SPC_R.csv"),header=T,na.strings="NULL")

#the years are absent, but appear in this file:
adatasbs_exp=read.csv(paste0(loc,"Raw
data\\AmSam\\A_SBS_EXP_R.csv"),header=T,na.strings="NULL")
```

```
dat<-merge(adatasbs,adatasbs_exp) # merge the data sets to provide year for the spc values

adatasbs2017=read.csv(paste0(loc,"Raw data\\AmSam\\A_SBS_SPC2017.csv"),header=T)
adatasbs2017$YEAR=2017

adatasbs=bind_rows(dat,adatasbs2017[,-1])

#243 is not a real species.  should be Aphareus rutilans (247)
adatasbs$SPECIES_FK[adatasbs$SPECIES_FK==243]<-247


#Sum catches by year and species
asbs.yrsums = adatasbs %>%
  filter(SPECIES_FK %in% asp$species) %>%
  select(SPECIES_FK,EXP_LBS,YEAR) %>%
  group_by(YEAR,SPECIES_FK) %>%
  summarise(total = sum(EXP_LBS,na.rm=T))
#table the catches by year (rows) and species codes (columns)
asbs.table = spread(asbs.yrsums,"SPECIES_FK","total")
#Ensure all years are included in table. They are
asbs.table$YEAR
#pull out catch from individual BMUS species
ind.bmus.sbs=asbs.table[,colnames(asbs.table)%in%asp$species[1:11]]

#The columns in abbs.table associated with the species groups. Need to remove
#the first column (year) in abbs.table but then have to add it back to get
#numbering right. Then....
group.cols=match(asp$species,as.numeric(names(asbs.table)[-1]))+1
cbind(asp[,c("species","spec_name")],group.cols)
#....multiply group catch by the corresponding proportions of that group that are BMUS
#pull out catch from species groups assumed to be BMUS
group.bmus.sbs=asbs.table[,group.cols[12:20][!is.na(group.cols[12:20])]]*(aspg.prop.sbs[-
(8:16),-1][,which(!is.na(group.cols[12:20]))])
#Need to used only the groups that are in the data. Thus take only these columns from
aspg.prop.bbs
#aspg.prop.sbs has 9 more rows than asbs.table - need to remove 1997-2005 hence -(8:16)
asbs.table[,group.cols[12:20][!is.na(group.cols[12:20])]]=asbs.table[,group.cols[12:20][!is.na(gr
oup.cols[12:20])]]*(aspg.prop.sbs[-(8:16),-1][,which(!is.na(group.cols[12:20]))])



################################################################################
################################################################################

################################################################################
################################################################
```

```
#Commercial purchase data


acp<-read.csv(paste0(loc,"Raw data\\AmSam\\asam_acl_commercial_14May18.csv"),header=T)
acp$year=as.numeric(substr(acp$INVOICE_DATE,1,4))
acp<-acp[(acp$IMPORTED_F==FALSE) & acp$RESALE_F==FALSE,] #get rid of the imports

#243 is not a real species.  Should be Aphareus rutilans (247)
acp$SPECIES_FK[acp$SPECIES_FK==243]<-247

#Sum catches by year and species
acp.yrsums = acp %>%
  filter(SPECIES_FK %in% asp$species) %>%
  select(SPECIES_FK,SOLD_LBS,year) %>%
  group_by(year,SPECIES_FK) %>%
  summarise(total = sum(SOLD_LBS,na.rm=T))
#table the catches by year (rows) and species codes (columns)
acp.table = spread(acp.yrsums,"SPECIES_FK","total")
#pull out catch from individual BMUS species
ind.bmus.cp=acp.table[,colnames(acp.table)%in%asp$species[c(1:11)]]

#Match columns in tabled catches with groups and then...
group.cols=match(asp$species,as.numeric(names(acp.table)[-1]))+1
cbind(asp[,c("species","spec_name")],group.cols)
#....multiply group catch by the corresponding proportions of that group that are BMUS
#pull out catch from species groups assumed to be BMUS
group.bmus.cp=acp.table[,group.cols[12:20][!is.na(group.cols[12:20])]]*(aspg.prop.cp[,-
1][,which(!is.na(group.cols[12:20]))])
#Need to used only the groups that are in the data. Thus take only these columns from
aspg.prop.sbs - 1990 through 2017
acp.table[,group.cols[12:20][!is.na(group.cols[12:20])]]=acp.table[,group.cols[12:20][!is.na(grou
p.cols[12:20])]]*(aspg.prop.cp[,-1][,which(!is.na(group.cols[12:20]))])


###
#Final catch by year
###
abbs.TOTC=data.frame("year"=abbs.table[,1],"totlb"=rowSums(abbs.table[,-1],na.rm=T))
asbs.TOTC=data.frame("year"=asbs.table[,1],"totlb"=rowSums(asbs.table[,-1],na.rm=T))
colnames(asbs.TOTC)[colnames(asbs.TOTC)=="YEAR"] <- "year" #rename for merge
acp.TOTC=data.frame("year"=acp.table[,1], "totlb"=rowSums(acp.table[,-1],na.rm=T))
a.TOTC=merge(merge(abbs.TOTC,asbs.TOTC,by="year",all=T),acp.TOTC,by="year",all=T)
write.csv(a.TOTC,paste0(loc,"TotalCatch_AmSam_init.csv"),row.names=F)
```

```
##############Algorithms for combining missing data in
catches############################
#Read in all files
catch.folder="Folder where catch data by dataset are located"

catch.folder=loc

aCatch=read.csv(paste0(catch.folder,"TotalCatch_AmSam_init.csv"),header=T)

allCatch=round(aCatch,1)

#Algorithm for filling in missing values
#1. For filling missing SBS values, use overall average SBS catch.
#after merge column header totlb.x = BBS, totlb.y=SBS, totlb=Commercial purchase

areNA=is.na(allCatch$totlb.y)!=is.na(allCatch$totlb.x)
allCatch$totlb.y[areNA]=mean(allCatch$totlb.y,na.rm=T)

#2. Sum together SBS and BBS data.
allCatch$cTOT.lb=round(allCatch$totlb.x+allCatch$totlb.y,0)

#3. Take max of SBS+BBS and original ComPurc data
cmax=which(allCatch$cTOT.lb<allCatch$totlb)
allCatch$cTOT.lb[cmax]=round(allCatch$totlb[cmax],0) #replace value with that from ComPurc

#4. Add in CVs for BBS data for final file output
allCatch$cCV=round(sqrt(abbs.TOTV$totvar)/allCatch$totlb.x,2)

##
#Output final datasets for use in the assessment
write.csv(allCatch[!is.na(allCatch$cTOT.lb),c("year","cTOT.lb","cCV")],paste0(loc,"TotalCatch
_AmSam_FINAL.csv"),row.names=F)
```

2.3.3. CPUE data filtering

```
###
#Steps to obtain nominal CPUE dataset
###

rm(list=ls())
library(dplyr)
library(tidyr)
options(scipen=999)
loc="Folder where datasets reside"
```

```
######
#READ IN DATA SETUP FILES
######

#BMUS species/groups
asp <- read.csv(paste0(loc,"Raw data\\Species
lists\\AmSam_BFspecieslist_2018.csv"),header=T)
#Interview data
aint.bbs=read.csv(paste0(loc,"Raw
data\\AmSam\\asam_bbs_interview_01May18.csv"),header=T)
key<-read.csv("C:\\Users\\John.Syslo\\Documents\\Territorial Bottomfish\\Data
processing\\A_area.csv",header=T)
#proportions of species groups that are BMUS
a.proptable.bbs=read.csv(paste0(loc,"Raw data\\Species
lists\\Abbs_GroupProportions.csv"),header=T)


######
#CPUE CALCULATIONS SETUP
######

#Need to replace SPECIES_FK 243 with 247 - Aphareus rutilans
aint.bbs$SPECIES_FK[aint.bbs$SPECIES_FK==243]<-247

#convert AREA_FK to larger PRIMARY_AREA_FK
aint.bbs<-merge(aint.bbs,key,by.x="AREA_FK", by.y="AREA_PK", all=TRUE)

#now get rid of everything not PRIMARY_AREA_FK
names(aint.bbs)
names(key)
aint.bbs<-aint.bbs[,-c(64,65,67:72)]

#INTERVIEW_PK field does not represent unique interviews.  Create field that does
unique_interview<-
paste(aint.bbs$INTERVIEW_PK,"_",aint.bbs$VESSEL_NAME,"_",aint.bbs$TOT_EST_LBS,s
ep="")

aint.bbs<-cbind(aint.bbs,unique_interview)

#Set up year column
aint.bbs$year<-as.numeric(format(as.Date(aint.bbs$SAMPLE_DATE,
format="%m/%d/%Y"),"%Y"))

colnames(a.proptable.bbs)=c("year",as.character(asp$species[12:20]))
a.propkey.bbs=gather(a.proptable.bbs,key="year",value="prop")
```

```
#need to add year ?
yr<-rep(seq(1986,2017,1),(20-12+1))
a.propkey.bbs<-cbind(yr,a.propkey.bbs)
colnames(a.propkey.bbs)[1]="year"
colnames(a.propkey.bbs)[2]="SPECIES_FK"


aint1.bbs=merge(aint.bbs,a.propkey.bbs,by=c("year","SPECIES_FK"),all.x=T)
aint1.bbs[is.na(aint1.bbs$prop),"prop"]=1


aint1.bbs$est_lbs=aint1.bbs$EST_LBS*aint1.bbs$prop

temp.bbs=unique(aint1.bbs[,c("unique_interview","SPECIES_FK","EST_LBS","est_lbs")])
bbs.all=aggregate(EST_LBS~unique_interview,data=temp.bbs,FUN=sum) #Aggregate over
EST_LBS since it doesn't have group proportions applied
bbs.bmus=aggregate(est_lbs~unique_interview,data=temp.bbs[temp.bbs$SPECIES_FK%in%asp
$species,],FUN=sum) #Use est_kgs since it has group proportions applied
bbs.perctrip=merge(bbs.all,bbs.bmus,by="unique_interview",all.x=T)
names(bbs.perctrip)[2:3]=c("EST_LBS.all","est_lbs.bmus")
bbs.perctrip[is.na(bbs.perctrip$est_lbs.bmus),"est_lbs.bmus"]=0 #set interviews without BMUS
to be 0 LBS
bbs.perctrip$percBMUS=bbs.perctrip$est_lbs.bmus/bbs.perctrip$EST_LBS.all
aint2.bbs=merge(aint1.bbs,bbs.perctrip,by="unique_interview",all.x=T)


aint2.bbs$effort=aint2.bbs$NUM_GEAR*aint2.bbs$HOURS_FISHED
aint2.bbs$cpue=aint2.bbs$est_lbs.bmus/aint2.bbs$effort
aint2.bbs$month=as.factor(format(as.Date(aint2.bbs$SAMPLE_DATE,
format="%m/%d/%Y"),"%m"))


aint3.bbs=aint2.bbs[,c("unique_interview","year","month","SPECIES_FK","SAMPLE_DATE","
TYPE_OF_DAY","INTERVIEW_TIME","AREA_FK", "PRIMARY_AREA_FK",
"METHOD_FK","METHOD_NAME","CHARTER_F",
"HOURS_FISHED","NUM_FISHER","NUM_GEAR","VESSEL_NAME","PORT_NAME","T
OT_EST_LBS","LUNAR_DAY","EST_LBS","prop","est_lbs","EST_LBS.all","est_lbs.bmus","
percBMUS","effort","cpue")]

#Remove SPECIES_FK, est_lbs, prop, and EST_LBS and reduce to unique trips
aint4.bbs=unique(aint3.bbs[,-c(4,20,21,22)])

#How many vessel_names have never caught a BMUS?
tempv=aggregate(est_lbs.bmus~VESSEL_NAME,data=aint4.bbs[aint4.bbs$METHOD_NAME
=="BOTTOMFISHING",],FUN=sum,na.rm=T)
```

```
head(table(tempv$est_lbs.bmus))
zero.vess=tempv[tempv$est_lbs.bmus==0,"VESSEL_NAME"]

#Limit interviews to be those on Method "Bottom", METHOD_FK=2
a<-aint4.bbs[aint4.bbs$METHOD_NAME=="BOTTOMFISHING",]

#Remove vessels that never caught BMUS
aint5.bbs=aint4.bbs[(aint4.bbs$METHOD_NAME=="BOTTOMFISHING" &
!aint4.bbs$VESSEL_NAME%in%zero.vess),]

#Use fields: type of day, area, wind speed, depth category, vessel name, and charter indicator.
#Remove any interviews without records of these.
aint6.bbs=aint5.bbs[,c("year","month","cpue","effort","TYPE_OF_DAY","PRIMARY_AREA_
FK","AREA_FK","VESSEL_NAME")]
aint6.bbs=aint6.bbs[is.finite(aint6.bbs$cpue),]
aint6.bbs=aint6.bbs[!is.na(aint6.bbs$VESSEL_NAME),]
aint6.bbs=aint6.bbs[!is.na(aint6.bbs$month),]
aint6.bbs=aint6.bbs[!is.na(aint6.bbs$year),]
aint6.bbs=aint6.bbs[!is.na(aint6.bbs$PRIMARY_AREA_FK),]
aint6.bbs=aint6.bbs[!is.na(aint6.bbs$AREA_FK),]
aint6.bbs=droplevels(aint6.bbs)

##
#Output dataset
##
#write.csv(aint6.bbs,paste0(loc,"Data
processing\\CPUE\\FinalizedAmSam_intCPUE_forStandardization_111318.csv"),row.names=F)
```

2.3.4. CPUE standardization

```
###
#Steps to perform CPUE standardization for American Samoa
###

rm(list=ls())
library(lme4)
library(MASS)
library(statmod)
library(png)
loc="Folder where nominal dataset resides"

dat=read.csv(paste0(loc,"FinalizedAmSam_intCPUE_forStandardization_111318.csv"),header=
T)
```

```
#remove 1985 - removes 30 values
dat<-dat[dat$year>1985,]

#Turn year, month, area, into factors
dat$year=as.factor(dat$year)
dat$month=as.factor(dat$month)
dat$AREA_FK=as.factor(dat$AREA_FK)
dat$PRIMARY_AREA_FK=as.factor(dat$PRIMARY_AREA_FK)

#Add 0-1 indicator variable
dat$z=0
dat[dat$cpue>0,"z"]=1

#Set up positive only database
dat.pos=dat[dat$cpue>0,] #6.5% are zeros

#Check distributional assumptions using MASS package
#compare AIC scores for lognormal and gamma for positive data
#find that gamma is better than lognormal
agam=fitdistr(dat.pos$cpue,lower=0.0001,"gamma")
aln=fitdistr(dat.pos$cpue,"log-normal")
AIC(agam)
AIC(aln) #Gamma has lower AIC

####
#BERNOULLI
####

#exploratory model with vessel
#Use year, month, type of day, area,
exp.mod<-
glmer(z~year+(1|VESSEL_NAME)+month+PRIMARY_AREA_FK+TYPE_OF_DAY,family=
binomial,data=dat) #does not converge

#Full model
ab.model0=glm(z~year+month+PRIMARY_AREA_FK+TYPE_OF_DAY,family=binomial,data
=dat)
ab.model0a=glm(z~year+month+PRIMARY_AREA_FK,family=binomial,data=dat)
ab.model0b=glm(z~year+month+TYPE_OF_DAY,family=binomial,data=dat)
ab.model0c=glm(z~year+TYPE_OF_DAY+PRIMARY_AREA_FK,family=binomial,data=dat)
abAIC0=rbind("Full"=extractAIC(ab.model0),"-T"=extractAIC(ab.model0a),"-
A"=extractAIC(ab.model0b),
          "-M"=extractAIC(ab.model0c))

abAIC0=cbind(abAIC0,(abAIC0[,2]-abAIC0[1,2])) #
abAIC0
```

```
which.min(abAIC0[,3]) #remove month


#Step 2 - removed month
ab.model1a=glm(z~year+TYPE_OF_DAY+PRIMARY_AREA_FK,family=binomial,data=dat)
ab.model1b=glm(z~year+TYPE_OF_DAY,family=binomial,data=dat)
ab.model1c=glm(z~year+PRIMARY_AREA_FK,family=binomial,data=dat)

abAIC1=rbind("Full-M"=extractAIC(ab.model1a),"-A"=extractAIC(ab.model1b),
          "-T"=extractAIC(ab.model1c))

abAIC1=cbind(abAIC1,(abAIC1[,2]-abAIC1[1,2]))  #Full model has lowest AIC

which.min(abAIC1[,3]) #BEST MODEL CONTAINS year and type of day and primary area
abAIC1

summary(ab.model1a)

ab.modelBEST = ab.model1a #BEST MODEL CONTAINS year and type of day and primary
area

############################
#For table in report
####
abAIC0 #Full in first row, removed month in last
abAIC1 #Best in first

#Deviance reduction
ab.modelyear=glm(z~year,family=binomial,data=dat)
ab.modelnull=glm(z~1,family=binomial,data=dat)
(deviance(ab.modelyear)-deviance(ab.modelBEST))/deviance(ab.modelyear)
(deviance(ab.modelnull)-deviance(ab.modelBEST))/deviance(ab.modelnull)


#############################################################################
################################## gamma glm for positive process
#############################################################################

#the following glmerControl is used for convergence
#Default link (inverse) is used due to convergence issues with log link

ap.model0c=glmer(cpue~year+(1|VESSEL_NAME)+month+PRIMARY_AREA_FK+TYPE_O
F_DAY,data=dat.pos,family=Gamma,glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun
= 100000)))
summary(ap.model0c)
```

```
summary(ap.model0c)$sigma


#
ap.model1a=glmer(cpue~year+(1|VESSEL_NAME)+month+PRIMARY_AREA_FK,data=dat.p
os,family=Gamma,glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 100000)))
ap.model1b=glmer(cpue~year+(1|VESSEL_NAME)+month+TYPE_OF_DAY,data=dat.pos,fam
ily=Gamma,glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 100000)))
ap.model1c=glmer(cpue~year+(1|VESSEL_NAME)+PRIMARY_AREA_FK+TYPE_OF_DAY,
data=dat.pos,family=Gamma,glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun =
100000)))

apAIC1=rbind("Full"=extractAIC(ap.model0c),
        "-TD"=extractAIC(ap.model1a),"-A"=extractAIC(ap.model1b),"-
M"=extractAIC(ap.model1c))

apAIC1=cbind(apAIC1,(apAIC1[,2]-apAIC1[1,2]))


which.min(apAIC1[,3]) #Remove area


#step 2  - removed area

ap.model2a=glmer(cpue~year+(1|VESSEL_NAME)+TYPE_OF_DAY,data=dat.pos,family=Ga
mma,glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 100000)))
ap.model2b=glmer(cpue~year+(1|VESSEL_NAME)+month,data=dat.pos,family=Gamma,glmer
Control(optimizer = "bobyqa", optCtrl = list(maxfun = 100000)))


apAIC2=rbind("Full"=extractAIC(ap.model1b),
        "-M"=extractAIC(ap.model2a),"-TD"=extractAIC(ap.model2b))

apAIC2

apAIC2=cbind(apAIC2,(apAIC2[,2]-apAIC2[1,2]))
which.min(apAIC2[,3]) #Remove month


#step 3 - removed month

ap.model3a=glmer(cpue~year+(1|VESSEL_NAME),data=dat.pos,family=Gamma,glmerControl(
optimizer = "bobyqa", optCtrl = list(maxfun = 100000)))

apAIC3=rbind("Full"=extractAIC(ap.model2a),
        "-TD"=extractAIC(ap.model3a))
```

```
apAIC3=cbind(apAIC3,(apAIC3[,2]-apAIC3[1,2]))
which.min(apAIC3[,3]) #remove TYPE_OF_DAY
apAIC3

#BEST MODEL is assumed to be the following - we arrive at same model with lognormal!
ap.modelBEST=glmer(cpue~year+(1|VESSEL_NAME),data=dat.pos,family=Gamma,glmerCon
trol(optimizer = "bobyqa", optCtrl = list(maxfun = 100000)))
#ap.modelBEST=glmer(cpue~year+(1|VESSEL_NAME)+TYPE_OF_DAY,data=dat.pos,family
=Gamma(link="log"),glmerControl(optimizer = "bobyqa", optCtrl = list(maxfun = 100000)))
#log lnk will not converge

######## Save Rdata output ##########
save(dat,ab.modelBEST,ap.modelBEST,file=paste0(loc,"best_mods_AS_111318.RData"))
```

2.3.5. CPUE index calculation

```
#####################################
#Calculate standardized CPUE for American Samoa
#############################################

rm(list=ls())
loc="Folder where standardized model scripts reside"
load(file=paste0(loc,"best_mods_AS_111318.RData"))

dat.pos=dat[dat$cpue>0,]

## Extract predicted values for Bernoulli process ("b")- be sure to use type = 'response' which
provides the probability of having a non zero tow (Stefansson 1996)
ab.pred = data.frame('BIN.CATCH' = predict(ab.modelBEST, type = 'response'),'year' =
dat[,'year'])
## Use aggregate ('a') to get means, sd and variance for each year for the bernoulli process. Don't
transform after this.
ab.pred.a = aggregate(BIN.CATCH ~ year, ab.pred,  FUN = function(x) c(mean = mean(x), sd =
sd(x), var = var(x)))

## Extract predicted values for Positive process ("p") and bind to YEAR (for aggregating) #this
has an inverse link!!
ap.pred = data.frame('CPUE' = predict(ap.modelBEST), 'year' = dat.pos[,'year'])
## Backtransform positive process using dispersion following Brian's STM standardization and
that from Brodziak and Walsh (2013)
ap.pred$trans=1/(ap.pred$CPUE)
## Use aggregate ('a') to get means and sd for each year for the positive process (remember sd2
is var)
```

```
ap.pred.a = aggregate(trans ~ year, ap.pred, function(x) c(mean = mean(x), sd = sd(x), var =
var(x)))

## Index generation
## Multiply each estimate together and calculate the variance according to Brodziak and Walsh
(2013) but ultimately following Goodman (1960)- no bother with the covariance as it is set to 0.
varI=function(pmean,pvar,bmean,bvar){
  index_totvar = bvar*pvar + bvar*(pmean^2) + pvar*(bmean^2)  }
aI = data.frame('year' = ab.pred.a[,'year'], 'INDEX.EST' = ab.pred.a$BIN.CATCH[,'mean'] *
ap.pred.a$trans[,'mean'],'VARIANCE.FORM' =
varI(ap.pred.a$trans[,'mean'],ap.pred.a$trans[,'var'],ab.pred.a$BIN.CATCH[,'mean'],ab.pred.a$B
IN.CATCH[,'var']))
aI$SD = sqrt(aI$VARIANCE.FORM)
aI$CV = aI$SD/aI$INDEX.EST
aI$N=as.numeric(table(dat$year))
aI$SE=aI$SD/sqrt(aI$N)
aI$CV_mean=aI$SE/aI$INDEX.EST

write.csv(aI,paste0(loc,"Finalized_stdindex_AmSam_11-13-18.csv"), row.names = F)



## Plotting index compared to nominal
#Keep in mind that this nominal is for complete cases with repsect to variables of interest and so
is from 25% smaller N (1644) than previously (2200)
plot(y=aI$INDEX.EST,x=as.numeric(aI$year),type="l",col=2,ylim=c(0,6),lwd=2,xlab="Year",y
lab="CPUE (LB BMUS/line hr)",main="AmSam")
anominal=aggregate(dat$cpue,by=list(dat$year),FUN=function(x) c("mean"=mean(x)))
lines(y=anominal$x,x=anominal$Group.1,type="o")
legend("topright",c("std.","nom"),lty=1,pch=c(NA,1),col=c(2,1),bty="n",lwd=c(2,1))
```