#LBB-code

# Length-based Bayesian Biomass estimator (LBB)

# Fits LBB model to length frequency data to estimate Linf, Lc, M/K, F/K

# Derives reference points F/M, Z/K, Lopt, Lc\_opt, B/B0, B/Bmsy, Y/R

# Main code developed by Rainer Froese in May-June 2017, modified in April-May 2018

# Gianpaolo Coro and Henning Winker did the JAGS coding

# Gianpaolo added the code for "Best year" in April 2019

# Gives option in the ID file to correct for the piling-up effect, with Pile=0 no correction, Pile=1 full correction, Pile=999 degree of correction determined by fit

# Automatic package installation

list.of.packages <- c("R2jags", "Hmisc","lattice","survival","Formula","ggplot2","crayon")

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]

if(length(new.packages)) install.packages(new.packages)

rm(list=ls(all=TRUE)) # clear previous variables etc

options(digits=3) # displays all numbers with three significant digits as default

graphics.off() # close graphics windows from previous sessions

library(R2jags)

library(Hmisc)

library(crayon) # to display bold and italics in console

# Select stock to be analysed

Stock <- "ALB"

# Select file with stock ID info

ID.File <- "Example\_ID.csv"

# Settings

n.sim <- 10 # ifelse(Stock %in% c("CodRedFSim"),1,10) # number of years to be created in simulations

smooth.ts <- T # use three years moving average for B/B0 time series

##############################################################

# Functions

##############################################################

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

# Exploited B/B0 ratio from B&H equations, for variable F

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

# assuming that reported lengths are the lower bounds of length classes

# get lowest exploited (>= 0.01 F) length class and class width

BH <- function(AllLength,Linf,MK,FK,GausSel,selpar1,selpar2) {

 if(GausSel==F) {

 r.Lc <- selpar1

 r.alpha <- selpar2

 Lx <- AllLength[AllLength >= Linf\*(r.Lc-4.59/r.alpha)][1]

 } else if(GausSel==T) {

 r.GLmean <- selpar1

 r.SD <- selpar2

 Lx <- AllLength[AllLength >= Linf\*(r.GLmean-3\*r.SD)][1]

 }

 class.width <- median(diff(sort(unique(AllLength))))

 FM <- FK/MK

 # Linf=120;Lx=22.5;r.Lc=0.2917;r.alpha=60;MK=1.5385;FK=0.7692;FM=0.5;ZK=2.3077

 # uncomment above row for comparison of Y'R= 0.0332, B/B0=0.467 with CodLightSim

 r <- vector() # auxilliary reduction factor

 G <- vector() # product of reduction factors

 SL.bh <- vector() # selection at length

 YR1.2 <- vector() # relative yield per recruit per length class

 CPUER1.2 <- vector() # relative CPUE per recruit per length class

 B1.2 <- vector() # relative unexploited biomass per recruit by length class

 L.bh <- seq(from=Lx, to=Linf, by=class.width) # lengths to be considered

 r.L.bh <- L.bh / Linf # standardized lengths

 # calculate selection, Y'/R and CPUE'/R for every length class

 for(o in 1 : length(r.L.bh)) {

 if(GausSel==F) {

 if(o<length(r.L.bh)) { SL.bh[o] <- mean(c(1/(1+exp(-r.alpha\*(r.L.bh[o]-r.Lc))), # mean selection in length class

 1/(1+exp(-r.alpha\*(r.L.bh[o+1]-r.Lc)))))

 } else SL.bh[o] <- 1/(1+exp(-r.alpha\*(r.L.bh[o]-r.Lc)))

 } else if(GausSel==T) { # gill net selection

 if(o<length(r.L.bh)) { SL.bh[o] <- mean(c(exp(-((r.L.bh[o]-r.GLmean)^2/(2\*r.SD^2))), # mean selection in length class

 exp(-((r.L.bh[o+1]-r.GLmean)^2/(2\*r.SD^2)))))

 } else SL.bh[o] <- exp(-((r.L.bh[o]-r.GLmean)^2/(2\*r.SD^2)))

 } # end of calculation of selectivity loop

 if(o<length(r.L.bh)) {

 r[o] <- (1-r.L.bh[o+1])^(FK\*SL.bh[o])/(1-r.L.bh[o])^(FK\*SL.bh[o])

 G[o] <- prod(r[1:o]) }

 if(o==1) {

 YR1.2[o] <-(FM\*SL.bh[o]/(1+FM\*SL.bh[o])\*(1-r.L.bh[o])^MK\*(1-3\*(1-r.L.bh[o])/(1+1/

 (MK+FK\*SL.bh[o]))+3\*(1-r.L.bh[o])^2/(1+2/(MK+FK\*SL.bh[o]))-

 (1-r.L.bh[o])^3/(1+3/(MK+FK\*SL.bh[o])))) -

 (FM\*SL.bh[o]/(1+FM\*SL.bh[o])\*(1-r.L.bh[o+1])^MK\*(1-3\*(1-r.L.bh[o+1])/(1+1/

 (MK+FK\*SL.bh[o]))+3\*(1-r.L.bh[o+1])^2/(1+2/(MK+FK\*SL.bh[o]))-

 (1-r.L.bh[o+1])^3/(1+3/(MK+FK\*SL.bh[o]))))\*G[o]

 } else if(o==length(r.L.bh)) {

 YR1.2[o] <- (FM\*SL.bh[o]/(1+FM\*SL.bh[o])\*(1-r.L.bh[o])^MK\*(1-3\*(1-r.L.bh[o])/(1+1/

 (MK+FK\*SL.bh[o]))+3\*(1-r.L.bh[o])^2/(1+2/(MK+FK\*SL.bh[o]))-

 (1-r.L.bh[o])^3/(1+3/(MK+FK\*SL.bh[o])))) \* G[o-1]

 } else {

 YR1.2[o] <- (FM\*SL.bh[o]/(1+FM\*SL.bh[o])\*(1-r.L.bh[o])^MK\*(1-3\*(1-r.L.bh[o])/(1+1/

 (MK+FK\*SL.bh[o]))+3\*(1-r.L.bh[o])^2/(1+2/(MK+FK\*SL.bh[o]))-

 (1-r.L.bh[o])^3/(1+3/(MK+FK\*SL.bh[o])))) \* G[o-1] -

 (FM\*SL.bh[o]/(1+FM\*SL.bh[o])\*(1-r.L.bh[o+1])^MK\*(1-3\*(1-r.L.bh[o+1])/(1+1/

 (MK+FK\*SL.bh[o]))+3\*(1-r.L.bh[o+1])^2/(1+2/(MK+FK\*SL.bh[o]))-

 (1-r.L.bh[o+1])^3/(1+3/(MK+FK\*SL.bh[o]))))\*G[o]

 } # end of loop to calculate yield per length class

 CPUER1.2[o] <- YR1.2[o] / FM # CPUE/R = Y/R divided by F/M

 if(o<length(r.L.bh)) {

 B1.2[o] <- ((1-r.L.bh[o])^MK\*(1-3\*(1-r.L.bh[o])/(1+1/MK)+3\*(1-r.L.bh[o])^2/

 (1+2/MK)-(1-r.L.bh[o])^3/(1+3/MK)) -

 (1-r.L.bh[o+1])^MK\*(1-3\*(1-r.L.bh[o+1])/(1+1/MK)+3\*(1-r.L.bh[o+1])^2/

 (1+2/MK)-(1-r.L.bh[o+1])^3/(1+3/MK)))\*SL.bh[o]

 } else {

 B1.2[o] <- ((1-r.L.bh[o])^MK\*(1-3\*(1-r.L.bh[o])/(1+1/MK)+3\*(1-r.L.bh[o])^2/

 (1+2/MK)-(1-r.L.bh[o])^3/(1+3/MK)))\*SL.bh[o]

 }

 } # end of B&H loop through length classes

 BB0 <- sum(CPUER1.2)/sum(B1.2)

 YR <- sum(YR1.2)

 if(BB0 < 0.25) YR <- YR \* BB0 / 0.25 # reduce YR if recruitment and thus productivity is reduced

 return(list(BB0,YR))

} # end of BH function

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

# Function to aggregate data by year

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

AG <- function(dat) { # where dat contains dat$Year, dat$Length in cm, dat$CatchNo

# aggregate normalized annual LFs by weighing with square root of sample size

# get sum of frequencies per year

sum.Ny <- aggregate(Freq~Year,dat,sum)$Freq

# get the sqrt of the sum of frequencies for every year

sqrt.Ny <- sqrt(sum.Ny)

# get highest frequency in each year

max.Ny <- aggregate(Freq~Year,dat,max)$Freq

# get Number of Length bins in each year

binsN <- aggregate(Freq~Year,dat,length)$Freq

# create vectors for sqrt.Ni and sum.Ni to weigh LF data

sqrt.Ni = rep(sqrt.Ny,binsN)

sum.Ni = rep(sum.Ny,binsN)

#Do weighing

# Divide all years by sum.Ni and multiply by sqrt.Ni

LF.w = dat$Freq/sum.Ni\*sqrt.Ni

# Aggregate

LF = aggregate(LF.w, by=list(dat$Length),FUN=sum)

# Add correct column names

colnames(LF) <- c("Length","Freq")

return(LF)

} #end of aggregate function

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

# Function to plot LBB-fit for a single year

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

# expects lengths relative to Linf (L/Linf)

plot.year <- function(r.L.y,r.Freq.y,r.Lopt, r.Freq.pred.y,SL1, SL2, MK, FK, Linf,main) {

 plot(x=r.L.y, y= r.Freq.pred.y,

 xlab="Length / Linf",ylab="relative Frequency",

 xlim=c(0,1),ylim = c(0,1.2\*max(r.Freq.y)),

 col="red", type="l", bty="l",main=main,las=1)

 points(x=r.L.y,y=r.Freq.y, cex=0.5)

 lines(x=c(1,1), y=c(0,1.07\*max(r.Freq.y,na.rm=T)),col="darkgreen")

 text(x=1,y=1.15\*max(r.Freq.y,na.rm=T),"Linf",col="darkgreen")

 lines(x=c(r.Lopt,r.Lopt), y=c(0,1.07\*max(r.Freq.y,na.rm=T)),col="darkgreen")

 text(x=r.Lopt,y=1.15\*max(r.Freq.y,na.rm=T),"Lopt",col="darkgreen")

 text(x=0.15,y=0.8\*max(r.Freq.y,na.rm=T),paste("Linf=",format(Linf,digits=3),sep=""))

 text(x=0.15,y=0.6\*max(r.Freq.y,na.rm=T),paste("Z/K=",format(MK+FK,digits=3),sep=""))

}

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

# Function to apply preceding 3-years moving average

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

ma <- function(x){

 x.1 <- filter(x,rep(1/3,3),sides=1)

 x.1[1] <- x[1]

 x.1[2] <- (x[1]+x[2])/2

 return(x.1)

}

#############################################################

# read files with ID and with LF data to be analyzed

#############################################################

# read ID data

dat.ID <- read.csv(ID.File, header=T, stringsAsFactors=F)

# restrict ID data to selected Stock

dat.ID <- dat.ID[dat.ID$Stock==Stock,]

# read LF data

dat.raw <- read.csv(dat.ID$File, header=T, stringsAsFactors=F)

# restrict LF data to selected stock

dat.raw <- dat.raw[dat.raw$Stock == Stock,]

# remove NA records

dat.raw <- dat.raw[which(is.na(dat.raw$CatchNo)==F),]

# restrict analysis to one or more gears

if(is.na(dat.ID$Gears.user[1])==FALSE) dat.raw <- dat.raw[dat.raw$Gear %in% dat.ID$Gears.user,]

# make sure data are numeric

dat.raw$Length <- as.numeric(dat.raw$Length)

dat.raw$CatchNo <- as.numeric(dat.raw$CatchNo)

dat.raw$Year <- as.integer(dat.raw$Year)

# if StartYear is given, restrict data to >= StartYear

if(is.na(dat.ID$StartYear)==F) dat.raw <- dat.raw[dat.raw$Year>=dat.ID$StartYear,]

# if EndYear is given, restrict data to <= EndYear

if(is.na(dat.ID$EndYear)==F) dat.raw <- dat.raw[dat.raw$Year<=dat.ID$EndYear,]

# if Years.user are given, restrict data to these years

if(is.na(dat.ID$Years.user[[1]])==F) dat.raw <- dat.raw[dat.raw$Year %in% (strsplit(dat.ID$Years.user, ","))[[1]],] # code from GP

# use largest fish as Lmax

Lmax <- max(dat.raw$Length)/10

# use median of largest fish per year as Lmax.med

Lmax.med <- median(as.numeric(by(dat.raw$Length[dat.raw$CatchNo>0],dat.raw$Year[dat.raw$CatchNo>0],max)))/10

# if Linf.user is given, restict data to < Linf.user

if(is.na(dat.ID$Linf.user)==F) dat.raw <- dat.raw[dat.raw$Length<(dat.ID$Linf.user\*ifelse(dat.ID$mm.user==TRUE,1,10)),]

# if Lcut.user is given, restrict data to >= Lcut.user

if(is.na(dat.ID$Lcut.user)==F) dat.raw <- dat.raw[dat.raw$Length>=(dat.ID$Lcut.user\*ifelse(dat.ID$mm.user==TRUE,1,10)),]

# sort data by year and length

 dat.raw <- dat.raw[order(dat.raw$Year,dat.raw$Length),]

# check for selected year to show B/B0

 if(length(dat.ID$Year.select[dat.ID$Stock==Stock]) != 0 && is.na(dat.ID$Year.select[dat.ID$Stock==Stock])==F) {

 Year.sel <- dat.ID$Year.select[dat.ID$Stock==Stock]

 } else {Year.sel <- NA}

# Put data into vectors

StartYear <- min(dat.raw$Year)

EndYear <- max(dat.raw$Year)

AllYear <- dat.raw$Year

AllLength <- dat.raw$Length

if(dat.ID$mm.user==FALSE) AllLength <- AllLength/10

AllFreq <- dat.raw$CatchNo

Years <- sort(unique(AllYear))

nYears <- length(Years)

# if data are simulated, add noise and n.sim more years

if(substr(Stock,start=nchar(Stock)-2,stop=nchar(Stock))=="Sim") {

 n.L.sim <- length(AllLength)

 AllYearSim <- AllYear

 AllLengthSim <- AllLength

 AllFreqSim <- rlnorm(n=n.L.sim,mean=log(AllFreq),sd=0.1)

 if(!(Stock %in% c("CodfFSim","CodRecSim"))) { # CodfFSim and CodRecSim are simulations that should run for only one year

 for(i in 1 : (n.sim-1)) {

 AllYearSim <- append(AllYearSim,AllYear+i)

 AllLengthSim <- append(AllLengthSim,AllLength)

 AllFreqSim <- append(AllFreqSim,rlnorm(n=n.L.sim,mean=log(AllFreq),sd=0.1))

 }

 AllYear <- AllYearSim

 AllLength <- AllLengthSim

 AllFreq <- AllFreqSim

 Years <- sort(unique(AllYear))

 nYears <- length(Years)

 EndYear <- Years[nYears] }

} # end of simulation loop

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

# plot LF for all years to detect potential problems

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

for(z in 1:ceiling(nYears/6)) {

 #modification by Gianpaolo 09 07 17

 if(grepl("win",tolower(Sys.info()['sysname']))) {windows(12,8)

 } else if(grepl("linux",tolower(Sys.info()['sysname']))) {X11(12,8)

 } else {quartz(12,8)}

 par(mfrow=c(2,3))

 for(v in 1 : 6) {

 w <- v+(z-1)\*6

 if(w > nYears) break()

 df.p <- data.frame(AllYear[AllYear==Years[w]&AllFreq>0],AllLength[AllYear==Years[w]&AllFreq>0],AllFreq[AllYear==Years[w]&AllFreq>0])

 names(df.p) <- c("Year","Length","Freq")

 LF.p <- AG(dat=df.p) # function to aggregate data in case bins are not unique

 plot(x=LF.p$Length,y=LF.p$Freq,xlim=c(0,Lmax),xlab="",ylab="Freq",bty="l",main=Years[w],cex=0.5)

 }

}

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

# Print warning if MergeLF is used

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

if(dat.ID[dat.ID$Stock==Stock]$MergeLF==TRUE) {

 cat("Attention: LFs in subsequent years are merged and the first year is identical with the second")

}

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

# Print years and Lmax across all data for early orientation

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

cat("\n Lmax =",Lmax,", median Lmax =",Lmax.med,"cm, for potential setting of Linf.user in ID file \n\n")

cat(" Years in data set (for potential cut & paste into Years.user in ID file):\n", paste(Years,collapse=","),"\n")

cat("If error without hint occurs, copy years into Years.user and delete next year to be processed from string\n\n")

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

# Create matrix to store annual estimates

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

Ldat <- data.frame(Stock=rep(Stock,nYears),Year=rep(NA,nYears),

 Linf=rep(NA,nYears),

 Linf.lcl=rep(NA,nYears),

 Linf.ucl=rep(NA,nYears),

 Lc=rep(NA,nYears), # for trawl selection

 Lc.lcl=rep(NA,nYears),

 Lc.ucl=rep(NA,nYears),

 Lmean=rep(NA,nYears),

 r.alpha=rep(NA,nYears),

 r.alpha.lcl=rep(NA,nYears),

 r.alpha.ucl=rep(NA,nYears),

 r.GLmean=rep(NA,nYears),r.SD=rep(NA,nYears), # for gill net selection

 MK=rep(NA,nYears),

 MK.lcl=rep(NA,nYears),

 MK.ucl=rep(NA,nYears),

 FK=rep(NA,nYears),

 FK.lcl=rep(NA,nYears),

 FK.ucl=rep(NA,nYears),

 ZK=rep(NA,nYears),

 ZK.lcl=rep(NA,nYears),

 ZK.ucl=rep(NA,nYears),

 FM=rep(NA,nYears),

 FM.lcl=rep(NA,nYears),

 FM.ucl=rep(NA,nYears),

 r.Lopt=rep(NA,nYears),

 BB0=rep(NA,nYears),

 BB0.lcl=rep(NA,nYears),

 BB0.ucl=rep(NA,nYears),

 YR=rep(NA,nYears),

 YR.lcl=rep(NA,nYears),

 YR.ucl=rep(NA,nYears),

 perc.mat=rep(NA,nYears),

 L95=rep(NA,nYears))

Lfit <- matrix(list(),nYears,3)

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

# Use aggregated LF data for estimation of Linf (and overall Z/K)

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

df <- data.frame(AllYear,AllLength,AllFreq)

names(df) <- c("Year","Length","Freq")

LF.all <- AG(dat=df) # function to aggregate data by year

# standardize to max Freq

LF.all$Freq = LF.all$Freq/max(LF.all$Freq)

# remove leading empty records

LF.all <- LF.all[which(LF.all$Freq>0)[1] : length(LF.all$Length),]

# remove trailing empty records

LF.all <- LF.all[1 : which(LF.all$Length==max(LF.all$Length[LF.all$Freq>0])),]

# get number of records in LF.all

n.LF.all <- length(LF.all$Length)

# If no Linf is provided by the user (preferred), determine Linf from fully selected LF:

# Freq=Nstart\*exp(ZK\*(log(1-L/Linf)-log(1-Lstart/Linf)))

# Nstart is canceled out when dividing both sides by their sums

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

# determine start values of selection ogive to find first fully selected length class Lstart

 L10 <- LF.all$Length[which(LF.all$Freq>0.1)[1]] # use length at 10% of peak frequency as proxy for L10

 L90 <- LF.all$Length[which(LF.all$Freq>0.9)[1]] # use length at 90% of peak frequency as proxy for L90

 Lc.st <- ifelse(is.na(dat.ID$Lc.user)==TRUE,(L10 + L90)/2,dat.ID$Lc.user) # use mean of L10 and L90 as proxy for Lc, else user input

 alpha.st <- -log(1/LF.all$Freq[which(LF.all$Freq>0.1)[1]])/(L10-Lc.st) # use rearranged logistic curve to estimate slope alpha

 # determine start values for Linf and Z/K

 Linf.st <- ifelse(is.na(dat.ID$Linf.user)==F,dat.ID$Linf.user,Lmax.med) # use Linf.user or median Lmax across years as start value for Linf in nls analysis

 Lmean.st <- sum(LF.all$Length[LF.all$Length>=Lc.st]\*LF.all$Freq[LF.all$Length>=Lc.st])/

 sum(LF.all$Freq[LF.all$Length>=Lc.st])

 MK.st <- ifelse(is.na(dat.ID$MK.user)==TRUE, 1.5,dat.ID$MK.user) # default 1.5

 ZK.st <- (Linf.st-Lmean.st)/(Lmean.st-Lc.st) # the Holt equation

 FK.st <- ifelse((ZK.st-MK.st)>0,ZK.st-MK.st,0.3) # prevent M/K being larger than Z/K

 # get vectors with fully selected length classes for Linf estimation

 if(is.na(dat.ID$Lstart.user)==FALSE) {Lstart <- dat.ID$Lstart.user} else {

 Lstart <- (alpha.st\*Lc.st-log(1/0.95-1))/alpha.st # Length where selection probability is 0.95

 # test if there are enough (>=4) length classes for estimation of aggregated Linf and ZK

 Lstart.i <- which(LF.all>=Lstart)[1]

 Lmax.i <- length(LF.all$Length)

 peak.i <- which.max(LF.all$Freq)

 if(Lstart.i<(peak.i+1)) Lstart <- LF.all$Length[peak.i+1] # make sure fully selected length starts after peak

 if((Lmax.i-Lstart.i)<4) Lstart <- LF.all$Length[Lstart.i-1] # make sure enough length classes are available

 }

 # do not include Lmax to allow Linf < Lmax and to avoid error in nls when Linf-L becomes negative

 L.L <- LF.all$Length[LF.all$Length >= Lstart & LF.all$Length < Linf.st]

 L.Freq <- LF.all$Freq[LF.all$Length>=L.L[1]& LF.all$Length < Linf.st]

 if(length(L.L)<4) {

 #modification by Gianpaolo 09 07 17

 if(grepl("win",tolower(Sys.info()['sysname']))) {windows(6,4)

 } else if(grepl("linux",tolower(Sys.info()['sysname']))) {X11(6,4)

 } else {quartz(6,4)}

 plot(x=LF.all$Length,y=LF.all$Freq, bty="l",main=Stock)

 lines(x=c(Lstart,Lstart),y=c(0,0.9\*max(LF.all$Freq)),lty="dashed")

 text(x=Lstart,y=max(LF.all$Freq),"Lstart")

 lines(x=c(Linf.st,Linf.st),y=c(0,0.9\*max(LF.all$Freq)),lty="dashed")

 text(x=Linf.st,y=max(LF.all$Freq),"Lmax")

 stop("Too few fully selected data points: set Lstart.user\n")}

 # standardize frequencies by dividing by sum of observed frequencies, needed to drop NLstart from equation

 sum.L.Freq <- sum(L.Freq)

 L.Freq <- L.Freq/sum.L.Freq

 # use nls() to find Linf-ZK combination with least residuals

 if(is.na(dat.ID$Linf.user)==TRUE) {

 Linf.mod <- nls(L.Freq ~ ((Linf-L.L)/(Linf-Lstart))^ZK /

 sum(((Linf-L.L)/(Linf-Lstart))^ZK),

 start=list(ZK=ZK.st,Linf=Linf.st),

 lower=c(0.5\*ZK.st,0.999\*Linf.st),

 upper=c(1.5\*ZK.st,1.2\*Linf.st),

 algorithm = "port")

 ZK.nls <- as.numeric(coef(Linf.mod)[1])

 ZK.nls.sd <- as.numeric(coef(summary(Linf.mod))[,2][1])

 ZK.nls.lcl <- ZK.nls-1.96\*ZK.nls.sd

 ZK.nls.ucl <- ZK.nls+1.96\*ZK.nls.sd

 Linf.nls <- as.numeric(coef(Linf.mod)[2])

 Linf.nls.sd <- as.numeric(coef(summary(Linf.mod))[,2][2])

 Linf.lcl <- Linf.nls-1.96\*Linf.nls.sd

 Linf.ucl <- Linf.nls+1.96\*Linf.nls.sd

 } else { # end of loop to determine Linf and ZK.L

 # use given Linf and determine ZK.L

 # use Linf provided by user if given

 Linf.nls <- dat.ID$Linf.user

 Linf.nls.sd <- 0.01\*dat.ID$Linf.user

 ZK.mod <- nls(L.Freq ~ exp(ZK\*(log(1-L.L/Linf.nls)-log(1-L.L[1]/Linf.nls)))/

 sum(exp(ZK\*(log(1-L.L/Linf.nls)-log(1-L.L[1]/Linf.nls)))),

 start=list(ZK=ZK.st),

 lower=c(0.7\*ZK.st),

 upper=c(1.3\*ZK.st),

 algorithm = "port")

 ZK.nls <- as.numeric(coef(ZK.mod)[1])

 ZK.nls.sd <- as.numeric(coef(summary(ZK.mod))[,2][1])

 ZK.nls.lcl <- ZK.nls-1.96\*ZK.nls.sd

 ZK.nls.ucl <- ZK.nls+1.96\*ZK.nls.sd

} # end of loop if Linf is given by user

# get vector of all lengths <= prior Linf to avoid error in equation

AllFreq <- AllFreq[AllLength <= Linf.nls]

AllYear <- AllYear[AllLength <= Linf.nls]

AllLength <- AllLength[AllLength <= Linf.nls]

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

# Start LF analysis by year

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

cat("Running Jags model to fit SL and N distributions for",dat.ID$Species,"\n")

jagsFit<-c() #modification by GP to select the best year

# open window for plotting annual fits

if(grepl("win",tolower(Sys.info()['sysname']))) {windows(12,8,record=TRUE) # code for different OS by GP

} else if(grepl("linux",tolower(Sys.info()['sysname']))) {X11(12,8,record=TRUE)

} else {quartz(12,8,record=TRUE)}

par(mfrow=c(2,3))

i = 0 # start counter

for(Year in Years) {

 i = i+1 # i is the index of Years, which may contain gaps

 # if MergeLF==TRUE and if this is the second or heigher year and no simulation, aggregate LF with previous year LF

 if(dat.ID$MergeLF==TRUE & substr(Stock,start=nchar(Stock)-2,stop=nchar(Stock))!="Sim") {

 if(i==1) {AG.yr <- c(Year,Years[2])} else { # if first year, aggregate with second year

 AG.yr <- c(Years[i-1],Year) }

 } else AG.yr <- Year

 # aggregate data within the year (sometimes there are more than one sample per year)

 df <- data.frame(AllYear[AllYear%in%AG.yr],AllLength[AllYear%in%AG.yr],AllFreq[AllYear%in%AG.yr])

 names(df) <- c("Year","Length","Freq")

 LF.y <- AG(dat=df) # function to aggregate data by year and across years

 LF.y$Freq <- LF.y$Freq/sum(LF.y$Freq) # standardize frequencies

 # remove empty leading and trailing records

 LF.y <- LF.y[which(LF.y$Freq>0)[1] : length(LF.y$Length),]

 LF.y <- LF.y[1 : which.max(LF.y$Length[LF.y$Freq>0]),]

 # get vectors

 L.y <- LF.y$Length

 r.Freq.y <- LF.y$Freq

 # fill remaining zero frequencies with very small number, to avoid error

 r.Freq.y[r.Freq.y==0] <- min(r.Freq.y[r.Freq.y>0],na.rm=T)/100

 # enter data for this year into data frame

 Ldat$Year[i] <- Year

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

# Estimate annual parameters Lc, alpha, M/K, F/K from LF curve with trawl-type selection

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

 # determine priors

 n.L <- length(L.y)

 Linf.pr <- Linf.nls

 Linf.sd.pr <- ifelse(Linf.nls.sd/Linf.nls<0.01,Linf.nls.sd,0.01\*Linf.nls) # restict prior CV of Linf to < 0.01

 MK.pr <- MK.st

 MK.sd.pr <- ifelse(is.na(dat.ID$MK.user)==TRUE,0.15,0.075)

 Pile <- dat.ID$Pile

if(dat.ID$GausSel==FALSE){ # apply trawl-like selection

 Lc.pr <- ifelse(is.na(dat.ID$Lc.user)==TRUE,1.02\*Lc.st,dat.ID$Lc.user) # with 1.02 multiplier to account for systematic small underestimation

 Lc.sd.pr <- ifelse(is.na(dat.ID$Lc.user)==TRUE,0.1\*Lc.pr,0.05\*Lc.pr) # assume narrower SD if Lc is given by user

 r.max.Freq <- max(r.Freq.y,na.rm=T)

 r.alpha.pr <- -log(r.max.Freq/r.Freq.y[which(r.Freq.y>(0.1\*r.max.Freq))[1]])/(L10/Linf.nls-Lc.st/Linf.nls) # relative alpha for standardized data

 r.alpha.sd.pr<- 0.025\*r.alpha.pr

 FK.pr <- ifelse((ZK.nls-MK.st) > 0,ZK.nls-MK.st,0.3) # if Z/K <= M/K assume low F/K = 0.3

 # list of data to pass to JAGS plus list of parameters to estimate

 jags.data <- list ("r.Freq.y","L.y","n.L","Linf.pr","Linf.sd.pr","Lc.pr","Lc.sd.pr","r.alpha.pr","r.alpha.sd.pr","MK.pr","MK.sd.pr",

 "FK.pr","Pile")

 jags.params <- c("r.alpha.d","Lc.d","SL","xN","FK.d","MK.d","Linf.d","pile.fac","Freq.pred")

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

 # LBB JAGS model for trawl-like selection

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

 sink("SLNMod.jags")

cat("

 model {

 r.alpha.d\_tau <- pow(r.alpha.sd.pr, -2)

 r.alpha.d ~ dnorm(r.alpha.pr,r.alpha.d\_tau)

 Lc.d\_tau <- pow(Lc.sd.pr,-2)

 Lc.d ~ dnorm(Lc.pr,Lc.d\_tau) #

 MK.d\_tau <-pow(MK.sd.pr, -2) # strong prior on M/K

 MK.d ~ dnorm(MK.pr, MK.d\_tau)

 Linf.tau <- pow(Linf.sd.pr,-2)

 Linf.d ~ dnorm(Linf.pr,Linf.tau)

 FK.d ~ dlnorm(log(FK.pr),4) # wide prior range for F/K

 SL[1] ~ dlogis(0,1000)

 Freq.pred[1]<-0

 xN[1] <-1

 p.low <- ifelse(Pile==1,0.99,0)

 p.hi <- ifelse(Pile==0,0.01,1)

 pile.fac ~ dunif(p.low,p.hi)

 for(j in 2:n.L) {

 SL[j] <- 1/(1+exp(-r.alpha.d\*(((L.y[j]+L.y[j-1])/2)/Linf.d-Lc.d/Linf.d))) # selection at mid-length of bin

 xN[j] <- xN[j-1]\*((Linf.d-L.y[j])/(Linf.d-L.y[j-1]))^(MK.d+FK.d\*SL[j]) # predicted numbers without pile-up

 cN[j] <- (xN[j-1]-xN[j])/(MK.d+FK.d\*SL[j]) # predicted relative frequency with pile-up correction

 dN[j] <- cN[j]-xN[j] # difference between corrected and uncorrected frequencies

 uN[j] <- xN[j] + dN[j]\*pile.fac # gradual application of correction with pile.fac between 0 and 1

 Freq.pred[j]<-uN[j]\*SL[j] # relative frequencies of vulnerable individuals

 # normalize frequencies by dividing by sum of frequencies; multiply with 10 to avoid small numbers and with 1000 for effective sample size

 r.Freq.pred[j]<- Freq.pred[j]/sum(Freq.pred)\*10\*1000

 }

 #><> LIKELIHOOD FUNCTION

 #><> Fit observed to predicted LF data using a Dirichlet distribution (more robust in JAGS)

 r.Freq.y[2:n.L] ~ ddirch(r.Freq.pred[2:n.L])

 } # END OF MODEL

 ",fill = TRUE)

sink()

MODEL = "SLNMod.jags"

jagsfitSLN <- jags.parallel(data=jags.data, working.directory=NULL, inits=NULL,

 parameters.to.save=jags.params,

 model.file=paste(MODEL),

 n.burnin=300, n.thin=10, n.iter=600, n.chains=3)

jagsFit<-c(jagsFit,jagsfitSLN$BUGSoutput$pD) #modification by GP to select the best year according to the Deviance information criterion

# use median and percentiles

Ldat$Lc[i] <- median(jagsfitSLN$BUGSoutput$sims.list$Lc.d)

Ldat$Lc.lcl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$Lc.d,0.025)

Ldat$Lc.ucl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$Lc.d,0.975)

Ldat$Lmean[i] <- sum(L.y[L.y>=Ldat$Lc[i]]\*r.Freq.y[L.y>=Ldat$Lc[i]])/sum(r.Freq.y[L.y>=Ldat$Lc[i]])

Ldat$r.alpha[i] <- median(jagsfitSLN$BUGSoutput$sims.list$r.alpha.d)

Ldat$r.alpha.lcl[i]<- quantile(jagsfitSLN$BUGSoutput$sims.list$r.alpha.d,0.025)

Ldat$r.alpha.ucl[i]<- quantile(jagsfitSLN$BUGSoutput$sims.list$r.alpha.d,0.975)

Ldat$MK[i] <- median(jagsfitSLN$BUGSoutput$sims.list$MK.d)

Ldat$MK.lcl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$MK.d,0.025)

Ldat$MK.ucl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$MK.d,0.975)

Ldat$FK[i] <- median(jagsfitSLN$BUGSoutput$sims.list$FK.d)

Ldat$FK.lcl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$FK.d,0.025)

Ldat$FK.ucl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$FK.d,0.975)

FMi <- jagsfitSLN$BUGSoutput$sims.list$FK.d/jagsfitSLN$BUGSoutput$sims.list$MK.d

Ldat$FM[i] <- median(FMi)

Ldat$FM.lcl[i] <- quantile(FMi,0.025)

Ldat$FM.ucl[i] <- quantile(FMi,0.975)

ZKi <- jagsfitSLN$BUGSoutput$sims.list$MK.d + jagsfitSLN$BUGSoutput$sims.list$FK.d

Ldat$ZK[i] <- median(ZKi)

Ldat$ZK.lcl[i] <- quantile(ZKi,0.025)

Ldat$ZK.ucl[i] <- quantile(ZKi,0.975)

Ldat$r.Lopt[i] <- 3/(3+Ldat$MK[i])

Ldat$Linf[i] <- median((jagsfitSLN$BUGSoutput$sims.list$Linf.d))

Ldat$Linf.lcl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$Linf.d,0.025)

Ldat$Linf.ucl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$Linf.d,0.975)

} # end of trawl-like selection

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

# Estimate parameters GLmean, SD, F/K, M/K if selection is gillnet-like

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

if(dat.ID$GausSel==TRUE) {

 # determine priors

 # assume length at peak Freq as mean and distance to length at 80% of peak as SD of mean

 GLmean.st <- L.y[which.max(r.Freq.y)]

 # assume SD of Gaussian selection as distance between length at peak and length at 50% of peak

 Lc.pr <- L.y[which(r.Freq.y >= (0.5\*max(r.Freq.y)))][1]

 SD.st <- max(GLmean.st-Lc.pr,0.25\*GLmean.st)

 cat("Running Jags model to fit SL and N distributions for gillnet-like selection\n")

 n.L <- length(L.y)

 jags.data <- list ("n.L","GLmean.st","L.y","SD.st","ZK.nls","r.Freq.y","Linf.pr","Linf.sd.pr","MK.pr")

 jags.params <- c("GLmean.d","SD.d","SL","xN","FK.d","MK.d","Linf.d","Freq.pred")

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

 # JAGS model L-based with integral

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

 sink("SLNMod.jags")

 cat("

 model {

 GLmean.tau <- pow(0.1\*GLmean.st,-2)

 GLmean.d ~ dnorm(GLmean.st,GLmean.tau)

 SD.tau <- pow(0.2\*SD.st,-2)

 SD.d ~ dnorm(SD.st,SD.tau)

 MK.d\_tau <-pow(0.15,-2)

 MK.d ~ dnorm(MK.pr,MK.d\_tau)

 Linf.tau <- pow(Linf.sd.pr,-2)

 Linf.d ~ dnorm(Linf.pr,Linf.tau)

 FK <- (ZK.nls-1.5) # ZK overestimated in gillnet selection, used as upper range

 FK.d ~ dunif(0,FK)

 SL[1]~ dlogis(0,1000)

 Freq.pred[1]<-0

 xN[1]<-1

 for(j in 2:n.L) {

 SL[j]<- exp(-((L.y[j]-GLmean.d)^2/(2\*SD.d^2)))

 xN[j]<-xN[j-1]\*exp((MK.d+FK.d\*SL[j])\*(log(1-L.y[j]/Linf.d)-log(1-L.y[j-1]/Linf.d)))

 cN[j] <- (xN[j-1]-xN[j])/(MK.d+FK.d\*SL[j])

 Freq.pred[j]<-cN[j]\*SL[j]

 #><> add effective sample size (try 100 typical for LF data)

 r.Freq.pred[j]<- Freq.pred[j]/sum(Freq.pred)\*10000

 }

 #><> LIKELIHOOD FUNCTION

 #><> Fit observed to predicted LF data using a Dirichlet distribution (more robust in JAGS)

 r.Freq.y[2:n.L]~ddirch(r.Freq.pred[2:n.L])

 } # END OF MODEL

 ",fill = TRUE)

 sink()

 MODEL = "SLNMod.jags"

 #jagsfitSLN <- jags(jags.data, inits=NULL, jags.params, paste(MODEL), n.chains = Nchains , n.thin =Nthin , n.iter =Niter , n.burnin = Nburnin)

 jagsfitSLN <- jags.parallel(data=jags.data, working.directory=NULL, inits=NULL,

 parameters.to.save=jags.params,

 model.file=paste(MODEL),

 n.burnin=300, n.thin=10, n.iter=1000, n.chains=3)

 jagsFit<-c(jagsFit,jagsfitSLN$BUGSoutput$pD) #modification by GP to select the best year according to the Deviance information criterion

 # use median and percentiles

 Ldat$GLmean[i] <- median(jagsfitSLN$BUGSoutput$sims.list$GLmean.d)

 Ldat$GLmean.lcl[i]<- quantile(jagsfitSLN$BUGSoutput$sims.list$GLmean.d,0.025)

 Ldat$GLmean.ucl[i]<- quantile(jagsfitSLN$BUGSoutput$sims.list$GLmean.d,0.975)

 Ldat$SD[i] <- median(jagsfitSLN$BUGSoutput$sims.list$SD.d)

 Ldat$SD.lcl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$SD.d,0.025)

 Ldat$SD.ucl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$SD.d,0.975)

 Ldat$MK[i] <- median(jagsfitSLN$BUGSoutput$sims.list$MK.d)

 Ldat$MK.lcl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$MK.d,0.025)

 Ldat$MK.ucl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$MK.d,0.975)

 Ldat$FK[i] <- median(jagsfitSLN$BUGSoutput$sims.list$FK.d)

 Ldat$FK.lcl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$FK.d,0.025)

 Ldat$FK.ucl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$FK.d,0.975)

 FMi <- jagsfitSLN$BUGSoutput$sims.list$FK.d/jagsfitSLN$BUGSoutput$sims.list$MK.d

 Ldat$FM[i] <- median(FMi)

 Ldat$FM.lcl[i] <- quantile(FMi,0.025)

 Ldat$FM.ucl[i] <- quantile(FMi,0.975)

 ZKi <- jagsfitSLN$BUGSoutput$sims.list$MK.d + jagsfitSLN$BUGSoutput$sims.list$FK.d

 Ldat$ZK[i] <- median(ZKi)

 Ldat$ZK.lcl[i] <- quantile(ZKi,0.025)

 Ldat$ZK.ucl[i] <- quantile(ZKi,0.975)

 Ldat$r.Lopt[i] <- 3/(3+Ldat$MK[i])

 Ldat$Linf[i] <- median((jagsfitSLN$BUGSoutput$sims.list$Linf.d))

 Ldat$Linf.lcl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$Linf.d,0.025)

 Ldat$Linf.ucl[i] <- quantile(jagsfitSLN$BUGSoutput$sims.list$Linf.d,0.975)

} # end of gillnet loop

# call BH function to estimate B/B0 and YR for the given year [i]

 BH.list <- BH(AllLength=unique(AllLength[AllYear==Year]),Linf=Ldat$Linf[i],MK=Ldat$MK[i],FK=Ldat$FK[i],GausSel=dat.ID$GausSel,

 selpar1=ifelse(dat.ID$GausSel==T,Ldat$GLmean[i]/Ldat$Linf[i],Ldat$Lc[i]/Ldat$Linf[i]),

 selpar2=ifelse(dat.ID$GausSel==T,Ldat$SD[i]/Ldat$Linf[i],Ldat$r.alpha[i]))

 Ldat$BB0[i] <- as.numeric(BH.list[1])

 Ldat$YR[i] <- as.numeric(BH.list[2])

 # Error propagation, assuming that fractional uncertainties add in quadrature

 rel.lcl <- sqrt(((Ldat$FM[i]-Ldat$FM.lcl[i])/Ldat$FM[i])^2+((Ldat$MK[i]-Ldat$MK.lcl[i])/Ldat$MK[i])^2+((Ldat$FK[i]-Ldat$FK.lcl[i])/Ldat$FK[i])^2+((Ldat$Linf[i]-Ldat$Linf.lcl[i])/Ldat$Linf[i])^2)

 rel.ucl <- sqrt(((Ldat$FM.ucl[i]-Ldat$FM[i])/Ldat$FM[i])^2+((Ldat$MK.ucl[i]-Ldat$MK[i])/Ldat$MK[i])^2+((Ldat$FK.ucl[i]-Ldat$FK[i])/Ldat$FK[i])^2+((Ldat$Linf.ucl[i]-Ldat$Linf[i])/Ldat$Linf[i])^2)

 Ldat$BB0.lcl[i] <- Ldat$BB0[i]-Ldat$BB0[i]\*rel.lcl

 Ldat$BB0.ucl[i] <- Ldat$BB0[i]+Ldat$BB0[i]\*rel.ucl

 Ldat$YR.lcl[i] <- Ldat$YR[i]-Ldat$YR[i]\*rel.lcl

 Ldat$YR.ucl[i] <- Ldat$YR[i]+Ldat$YR[i]\*rel.ucl

 # get MSFD D3.3 indicators

 Ldat$L95[i] <- wtd.quantile(x=L.y,weights=r.Freq.y,probs=c(0.95))

 Ldat$perc.mat[i] <- ifelse(is.na(dat.ID$Lm50)==F,sum(r.Freq.y[L.y>dat.ID$Lm50])/sum(r.Freq.y),NA)

# create and store vectors for plotting fit to years

 r.L.y <- L.y[L.y < Ldat$Linf[i]] / Ldat$Linf[i]

 r.Freq.y <- r.Freq.y[L.y < Ldat$Linf[i]]

 Freq.pred <- vector()

 for(k in 1:length(r.L.y)){

 Freq.pred[k] <- median(jagsfitSLN$BUGSoutput$sims.list$Freq.pred[,k])

 }

 Lfit[i,1][[1]] <- r.L.y

 Lfit[i,2][[1]] <- r.Freq.y

 Lfit[i,3][[1]] <- Freq.pred

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

 # plot LBB fit to detect potential problems

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

 plot.year(r.L.y=r.L.y, r.Freq.y=r.Freq.y,r.Lopt=Ldat$r.Lopt[i],

 r.Freq.pred.y = Freq.pred/sum(Freq.pred),

 SL1=ifelse(dat.ID$GausSel==T,Ldat$GLmean[i],Ldat$Lc[i]),

 SL2=ifelse(dat.ID$GausSel==T,Ldat$SD[i],Ldat$r.alpha[i]),

 MK=Ldat$MK[i],FK=Ldat$FK[i],Linf=Ldat$Linf[i],main=Years[i])

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

 # Check for unrealistic fits

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

 if(dat.ID$GausSel==FALSE && (Ldat$ZK[i]>25 || Ldat$ZK[i] < 0.9 || (Ldat$ZK[i]/median(Ldat$MK,na.rm=TRUE)) < 0.9 ||

 Ldat$r.Lopt[i] > 1 || Ldat$r.Lopt[i] < 0.3 || (Ldat$Lc[i]/median(Ldat$Lc,na.rm=TRUE)) > 1.8 ||

 (Ldat$Lc[i]/median(Ldat$Lc,na.rm=TRUE)) < 0.4 || Ldat$MK[i] <0)) {

 if(Ldat$MK[i] <0) {

 cat(red("\n",bold("WARNING!!")),"Unable to determine proper Lc, remove peaks of early juveniles

 by setting Lcut.user or removing such years or set MK.user=1.5 or remove",Years[i],"\n\n")} else {

 cat(red("\n",bold("WARNING!!")),"Year",Years[i],"data unsuitable for LBB analysis,

 use Years.user in ID file to specify suitable years.","\n\n")}

 stop("Unsuitable data")}

} # end of annual loop

# get some reference points as median of time series

Linf.med <- median(Ldat$Linf)

Linf.lcl <- median(Ldat$Linf.lcl)

Linf.ucl <- median(Ldat$Linf.ucl)

if(dat.ID$GausSel==F) {

 Lc.med <- median(Ldat$Lc)

 r.alpha.med <- median(Ldat$r.alpha) } else {

 GLmean.med <- median(Ldat$GLmean)

 SD.med <- median(Ldat$SD) }

MK.med <- median(Ldat$MK)

MK.lcl <- median(Ldat$MK.lcl)

MK.ucl <- median(Ldat$MK.ucl)

FK.med <- median(Ldat$FK)

FK.lcl <- median(Ldat$FK.lcl)

FK.ucl <- median(Ldat$FK.ucl)

FM.med <- median(Ldat$FM)

FM.lcl <- median(Ldat$FM.lcl)

FM.ucl <- median(Ldat$FM.ucl)

ZK.med <- median(Ldat$ZK)

ZK.lcl <- median(Ldat$ZK.lcl)

ZK.ucl <- median(Ldat$ZK.ucl)

r.Lopt.med <- median(Ldat$r.Lopt)

Lopt.med <- r.Lopt.med\*Linf.med

Lc\_opt.med <- Linf.med\*(2+3\*FM.med)/((1+FM.med)\*(3+MK.med))

BB0.med <- median(Ldat$BB0)

BB0.lcl <- median(Ldat$BB0.lcl)

BB0.ucl <- median(Ldat$BB0.ucl)

YR.med <- median(Ldat$YR)

YR.lcl <- median(Ldat$YR.lcl)

YR.ucl <- median(Ldat$YR.ucl)

BFM1B0.list <- BH(AllLength=unique(AllLength),Linf=Linf.med,MK=MK.med,FK=MK.med,GausSel=dat.ID$GausSel,

 selpar1=ifelse(dat.ID$GausSel==T,r.Lopt.med,5/(2\*(3+MK.med))),

 selpar2=ifelse(dat.ID$GausSel==T,SD.med/Linf.med,r.alpha.med))

BFM1B0 <- as.numeric(BFM1B0.list[1])

YRFM1 <- as.numeric(BFM1B0.list[2])

# mean length if F=M

if(dat.ID$GausSel==F) {

 LmeanFM <- (2\*Lc.med\*MK.med+Linf.med)/(2\*MK.med+1)} else {

 LmeanFM <- (2\*Lc.pr\*MK.med+Linf.med)/(2\*MK.med+1) }

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

# Apply smoothing if desired

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

if(smooth.ts==TRUE && nYears>=3) {

 Linf.ts <- ma(Ldat$Linf)

 Lmean.ts <- ma(Ldat$Lmean)

 Lc.ts <- ma(Ldat$Lc)

 Lc.lcl.ts <- ma(Ldat$Lc.lcl)

 Lc.ucl.ts <- ma(Ldat$Lc.ucl)

 r.alpha.ts <- ma(Ldat$r.alpha)

 r.alpha.lcl.ts <- ma(Ldat$r.alpha.lcl)

 r.alpha.ucl.ts <- ma(Ldat$r.alpha.ucl)

 r.Lopt.ts <- ma(Ldat$r.Lopt)

 L95.ts <- ma(Ldat$L95)

 perc.mat.ts <- ma(Ldat$perc.mat)

 FK.ts <- ma(Ldat$FK)

 FK.lcl.ts <- ma(Ldat$FK.lcl)

 FK.ucl.ts <- ma(Ldat$FK.ucl)

 FM.ts <- ma(Ldat$FM)

 FM.lcl.ts <- ma(Ldat$FM.lcl)

 FM.ucl.ts <- ma(Ldat$FM.ucl)

 ZK.ts <- ma(Ldat$ZK)

 ZK.lcl.ts <- ma(Ldat$ZK.lcl)

 ZK.ucl.ts <- ma(Ldat$ZK.ucl)

 YR.ts <- ma(Ldat$YR)

 YR.lcl.ts <- ma(Ldat$YR.lcl)

 YR.ucl.ts <- ma(Ldat$YR.ucl)

 BB0.ts <- ma(Ldat$BB0)

 BB0.lcl.ts <- ma(Ldat$BB0.lcl)

 BB0.ucl.ts <- ma(Ldat$BB0.ucl)

 if(dat.ID$GausSel==T) {

 GLmean.ts <- ma(Ldat$GLmean)

 GLmean.lcl.ts <- ma(Ldat$GLmean.lcl)

 GLmean.ucl.ts <- ma(Ldat$GLmean.ucl)

 SD.ts <- ma(Ldat$SD)

 }

} else {

 Linf.ts <- Ldat$Linf

 Lmean.ts <- Ldat$Lmean

 Lc.ts <- Ldat$Lc

 Lc.lcl.ts <- Ldat$Lc.lcl

 Lc.ucl.ts <- Ldat$Lc.ucl

 r.alpha.ts <- Ldat$r.alpha

 r.alpha.lcl.ts <- Ldat$r.alpha.lcl

 r.alpha.ucl.ts <- Ldat$r.alpha.ucl

 r.Lopt.ts <- Ldat$r.Lopt

 L95.ts <- Ldat$L95

 perc.mat.ts <- Ldat$perc.mat

 FK.ts <- Ldat$FK

 FK.lcl.ts <- Ldat$FK.lcl

 FK.ucl.ts <- Ldat$FK.ucl

 FM.ts <- Ldat$FM

 FM.lcl.ts <- Ldat$FM.lcl

 FM.ucl.ts <- Ldat$FM.ucl

 ZK.ts <- Ldat$ZK

 ZK.lcl.ts <- Ldat$ZK.lcl

 ZK.ucl.ts <- Ldat$ZK.ucl

 YR.ts <- Ldat$YR

 YR.lcl.ts <- Ldat$YR.lcl

 YR.ucl.ts <- Ldat$YR.ucl

 BB0.ts <- Ldat$BB0

 BB0.lcl.ts <- Ldat$BB0.lcl

 BB0.ucl.ts <- Ldat$BB0.ucl

 if(dat.ID$GausSel==T) {

 GLmean.ts <- Ldat$GLmean

 GLmean.lcl.ts <- Ldat$GLmean.lcl

 GLmean.ucl.ts <- Ldat$GLmean.ucl

 SD.ts <- Ldat$SD

 }

}

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

# Start printing results to screen

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

# print priors to screen

cat("\n----------------------------------------------------------------------\n")

cat("LBB results for ",bold(italic(dat.ID$Species)),", stock ",bold(Stock),", ",StartYear,"-",EndYear,ifelse(dat.ID$GausSel==T,", Gaussian selection",""),sep="","\n")

cat("Files:",ID.File,", ",dat.ID$File,sep="","\n")

cat("-----------------------------------------------------------------------\n")

cat("Linf prior= ",Linf.pr,", SD=",format(Linf.sd.pr,digits=2)," cm ",ifelse(is.na(dat.ID$Linf.user)==TRUE,"","(user-defined), "),

 "Lmax=",Lmax,", median Lmax=",Lmax.med,sep="","\n")

cat("Z/K prior = ",format(ZK.nls,digits=2),", SD=", format(ZK.nls.sd,digits=2),", M/K prior=", MK.pr, ", SD=",MK.sd.pr,

 ifelse(is.na(dat.ID$MK.user)==TRUE,"","(user-defined)"),sep="","\n")

if(dat.ID$GausSel==F) {

 cat("F/K prior =", FK.pr, "(wide range with tau=4 in log-normal distribution)\n")

 cat("Lc prior = ",Lc.pr,", SD=",format(Lc.sd.pr,digits=2)," cm",

 ifelse(is.na(dat.ID$Lc.user)==TRUE,""," (user-defined)"),

 ", alpha prior=",r.alpha.pr,", SD=",format(0.1\*r.alpha.pr,digits=2),

 ", Lm50=", dat.ID$Lm50,ifelse(dat.ID$mm.user==F," cm"," mm"),sep="","\n") }

if(dat.ID$Pile != 0) {

 cat("Pile-up correction applied with weight", format(ifelse(dat.ID$Pile==1.0,1.0,

 median(jagsfitSLN$BUGSoutput$sims.list$pile.fac)),nsmall=2),"\n")}

cat("\n")

cat("General reference points (median across years): \n")

cat("Linf = ",Linf.med," (",Linf.lcl,"-",Linf.ucl,

 ifelse(dat.ID$mm.user==F,") cm",") mm"), sep="", "\n")

cat("Lopt = ",format(Lopt.med,digits=2),ifelse(dat.ID$mm.user==F," cm,"," mm,")," Lopt/Linf=",format(r.Lopt.med,digits=2),sep="","\n")

cat("Lc\_opt = ",format(Lc\_opt.med,digits=2),ifelse(dat.ID$mm.user==F," cm,"," mm,"),

 " Lc\_opt/Linf=",format(Lc\_opt.med/Linf.med,digits=2),

 ", Lmean if F=M ",LmeanFM,ifelse(dat.ID$mm.user==F," cm"," mm"),sep="","\n")

cat("M/K = ",MK.med," (",MK.lcl,"-",MK.ucl,")",sep="","\n")

cat("F/M = ",FM.med," (",FM.lcl,"-",FM.ucl,"),"," F/K=",FK.med," (",FK.lcl,"-",FK.ucl,"),",

 " Z/K=",ZK.med," (",ZK.lcl,"-",ZK.ucl,")",sep="","\n")

cat("B/B0 = ",format(BB0.med,digits=2)," (",format(BB0.lcl,digits=2),"-",format(BB0.ucl,digits=2),")",

 ifelse(dat.ID$GausSel==F,", B/B0 F=M Lc=Lc\_opt ",", B/B0 F=M Lmean=Lopt "),format(BFM1B0,digits=2),sep="","\n")

if(BB0.lcl < -0.4 || BB0.ucl > 2) {

 cat(bold("WARNING: Uncertainty in B/B0 estimate is much too wide, data are unsuitable for stock assessment!\n"))

 stop("Data are unsuitable")

}

cat("Y/R' = ",format(YR.med,digits=2)," (",format(YR.lcl,digits=2),"-",format(YR.ucl,digits=2),")",

 ifelse(BB0.med < 0.25,"(reduced: B/B0<0.25),",", "),

 ifelse(dat.ID$GausSel==F,"Y/R' F=M Lc=Lc\_opt ","Y/R' F=M Lmean=Lopt "),format(YRFM1,digits=2),sep="","\n\n")

cat("Estimates for",EndYear,ifelse(smooth.ts==T,"(mean of last 3 years with data):",":"),"\n")

last <- which(Ldat$Year==EndYear)

if(dat.ID$GausSel==F){

 cat("Lc50 =",Lc.ts[last],paste("(",format(Lc.lcl.ts[last],digits=3),

 "-",format(Lc.ucl.ts[last],digits=3),ifelse(dat.ID$mm.user==F,") cm, Lc/Linf=",") mm, Lc/Linf"),

 format(Lc.ts[last]/Linf.ts[last],digits=2)," (",format(Lc.lcl.ts[last]/Linf.ts[last],digits=2),"-",

 format(Lc.ucl.ts[last]/Linf.ts[last],digits=2),")",sep=""),"\n")

 cat("Lc95 = ",format((r.alpha.ts[last]/Linf.ts[last]\*Lc.ts[last]-log(1/0.95-1))/(r.alpha.ts[last]/Linf.ts[last]),digits=3),

 ", alpha=",format(r.alpha.ts[last]/Linf.ts[last],digits=3)," (",format(r.alpha.lcl.ts[last]/Linf.ts[last],digits=3),"-",

 format(r.alpha.ucl.ts[last]/Linf.ts[last],digits=3),")",sep="","\n")

 cat("Lmean/Lopt= ",format(Lmean.ts[last]/(r.Lopt.ts[last]\*Linf.ts[last]),digits=2),

 ", Lc/Lc\_opt=",format(Lc.ts[last]/Lc\_opt.med,digits=2),

 ", L95th=", format(L95.ts[last],digits=3),ifelse(dat.ID$mm.user==F," cm,"," mm,"),

 " L95th/Linf=",format(L95.ts[last]/Linf.ts[last],digits=2),

 ", Mature=",format(Ldat$perc.mat[last]\*100,digits=2),"%",sep="","\n")

 } else if(dat.ID$GausSel==T){

 cat("GLmean/Linf=",format(GLmean.ts[last]/Linf.ts[last],digits=2),",SD/Linf =",SD.ts[last]/Linf.ts[last],"\n")

 cat("GLmean =",GLmean.ts[last],",SD =",SD.ts[last],"\n")

 }

cat("F/M = ",format(FM.ts[last],digits=2)," (",format(FM.lcl.ts[last],digits=2),"-",format(FM.ucl.ts[last],digits=2),"), F/K=",

 format(FK.ts[last],digits=2)," (",format(FK.lcl.ts[last],digits=2),"-",format(FK.ucl.ts[last],digits=2),"), Z/K=",

 format(ZK.ts[last],digits=2)," (",format(ZK.lcl.ts[last],digits=2),"-",format(ZK.ucl.ts[last],digits=2),")",sep="","\n")

cat("Y/R' = ",format(YR.ts[last],digits=2)," (",format(YR.lcl.ts[last],digits=2),"-",format(YR.ucl.ts[last],digits=2),")",

 ifelse(BB0.med < 0.25,"(reduced because B/B0 < 0.25)",""),sep="","\n")

bestfityr = which(jagsFit == min(jagsFit))

cat("B/B0 = ",format(BB0.ts[last],digits=2)," (",format(BB0.lcl.ts[last],digits=2),"-",

 format(BB0.ucl.ts[last],digits=2),"),",

 " best LF fit year ",Years[bestfityr],"=",format(BB0.ts[bestfityr],nsmall=2),

 " (",format(BB0.lcl.ts[bestfityr],digits=2),"-",format(BB0.ucl.ts[bestfityr],digits=2),")",sep="","\n")

# print B/B0 for selected year

if(is.na(Year.sel)==F) {

 BB0.sl <- BB0.ts[Years==Year.sel]

 BB0.lcl.sl <- BB0.lcl.ts[Years==Year.sel]

 BB0.ucl.sl <- BB0.ucl.ts[Years==Year.sel]

}

cat("B/Bmsy = ",format(BB0.ts[last]/BFM1B0,digits=2)," (",format(BB0.lcl.ts[last]/BFM1B0,digits=2),"-",

 format(BB0.ucl.ts[last]/BFM1B0,digits=2),")",

 ifelse(is.na(Year.sel)==F,

 bold(paste(", selected B/B0 ",Year.sel," = ",format(BB0.sl,digits=2)," (",format(BB0.lcl.sl,digits=2),"-",

 format(BB0.ucl.sl,digits=2),")",sep="")),""),sep="","\n")

if(dat.ID$Comment != "" && is.na(dat.ID$Comment)==F) cat(dat.ID$Comment,"\n")

# point out questionable or impossible results

# negative rates

if(Ldat$MK[last] < 0 | Ldat$FK[i] < 0) cat("Data unsuitable for LF analysis, negative mortality rates are impossible\n")

# Biomass larger than unexploited

if(Ldat$BB0[last] >1.1) cat(red("Data unsuitable for LF analysis, biomass exceeds carrying capacity"),"\n")

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

# Plot aggregated results

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

# plot aggregated histogram with fit to fully selected part

#modification by Gianpaolo 09 07 17

if(grepl("win",tolower(Sys.info()['sysname']))) {windows(12,8)

} else if(grepl("linux",tolower(Sys.info()['sysname']))) {X11(12,8)

} else {quartz(12,8)}

par(mfrow=c(2,3),las=1)

plot(x=LF.all$Length,y=LF.all$Freq, bty="l",xlim=c(0,max(max(LF.all$Length),Linf.nls)),

 ylim=c(0,1.1\*max(LF.all$Freq)),

 main=paste(Stock,", aggregated LF"),xlab=ifelse(dat.ID$mm.user==F,"Length (cm)","Length (mm)"),ylab="Frequency")

Lstart.i <- which(LF.all$Length>=Lstart)[1]

Lstart.Freq <- mean(c(LF.all$Freq[(Lstart.i-1):(Lstart.i+1)]))

if(dat.ID$GausSel==F) {

 lines(x=L.L,y=Lstart.Freq\*exp(ZK.nls\*(log(1-L.L/Linf.nls)-log(1-L.L[1]/Linf.nls))), col="blue", lwd=3)

} else {

 wt <- wtd.mean(LF.all$Length,LF.all$Freq)

 var <- wtd.var(LF.all$Length,LF.all$Freq)

 std <- sqrt(var)

 curve(dnorm(x,mean=wt,sd=std),col="blue",lwd=3,add=T)

}

lines(x=c(Lc.st,Lc.st), y=c(0,1), col="darkgreen")

text(x=Lc.st,y=1, "Lc", col="darkgreen", adj=c(0.5,-0.5))

lines(x=c(Linf.nls,Linf.nls), y=c(0,1), col="darkgreen")

text(x=Linf.nls,y=1, "Linf", col="darkgreen", adj=c(0.5,-0.5))

text(x=0.1\*Linf.nls,y=1,"Priors:")

text(x=0.15\*Linf.nls,y=0.8,paste("Linf=",format(Linf.nls,digits=3),sep=""))

if(dat.ID$GausSel==F) text(x=0.15\*Linf.nls,y=0.6,paste("Z/K=",format(ZK.nls,digits=2),sep=""))

text(x=0.1\*Linf.nls,y=0.4,paste("Lc=",format(Lc.st,digits=3),sep=""))

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

# plot first (or selected) and last year

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

# first or selected

 ys <- ifelse(is.na(Year.sel)==T || Year.sel == Years[nYears],1,which(Years==Year.sel))

 plot.year(r.L.y=Lfit[ys,1][[1]], r.Freq.y=Lfit[ys,2][[1]],r.Lopt=Ldat$r.Lopt[ys],

 r.Freq.pred.y = Lfit[ys,3][[1]]/sum(Lfit[ys,3][[1]]),

 SL1=ifelse(dat.ID$GausSel==T,Ldat$GLmean[ys],Ldat$Lc[ys]),

 SL2=ifelse(dat.ID$GausSel==T,Ldat$SD[ys],Ldat$r.alpha[ys]),

 MK=Ldat$MK[ys],FK=Ldat$FK[ys],Linf=Ldat$Linf[ys],main=Years[ys])

# last

 plot.year(r.L.y=Lfit[nYears,1][[1]], r.Freq.y=Lfit[nYears,2][[1]],r.Lopt=Ldat$r.Lopt[nYears],

 r.Freq.pred.y = Lfit[nYears,3][[1]]/sum(Lfit[nYears,3][[1]]),

 SL1=ifelse(dat.ID$GausSel==T,Ldat$GLmean[nYears],Ldat$Lc[nYears]),

 SL2=ifelse(dat.ID$GausSel==T,Ldat$SD[nYears],Ldat$r.alpha[nYears]),

 MK=Ldat$MK[nYears],FK=Ldat$FK[nYears],Linf=Ldat$Linf[nYears],main=Years[nYears])

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

# Plot time series of Lc and Lmean

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

if(nYears > 1) {

if(dat.ID$GausSel==F){

plot(x=Ldat$Year,y=Lmean.ts, bty="l",type="l",

 xlim=c(Ldat$Year[1],Ldat$Year[nYears]),

 xaxt="n",

 ylim=c(0,max(c(1.1\*Lopt.med,max(Lmean.ts,na.rm=T),max(Ldat$Lc.ucl),na.rm=T))),lwd=2,

 xlab="Year",ylab = paste("Length",ifelse(dat.ID$mm.user==F,"(cm)","(mm)")),main="Lmean vs Lopt & Lc vs Lc\_opt")

axis(1,at=Ldat$Year)

lines(x=Ldat$Year,y=Lc.ts,lwd=1,lty="dashed")

#lines(x=Ldat$Year,y=Ldat$Lc.lcl,lty="dotted")

#lines(x=Ldat$Year,y=Ldat$Lc.ucl,lty="dotted")

lines(x=Ldat$Year,y=rep(Lc\_opt.med,nYears),col="darkgreen", lty="dashed") # line for Lc\_opt

text(x=Ldat$Year[nYears],y=Lc\_opt.med,"Lc\_opt", adj=c(1,-0.5), col="darkgreen")

lines(x=Ldat$Year,y=rep(Lopt.med,nYears),col="darkgreen") # line for Lopt

text(x=Ldat$Year[nYears],y=Lopt.med,"Lopt", adj=c(1,-0.5), col="darkgreen")

lines(x=Ldat$Year,y=rep(LmeanFM,nYears),col="darkgreen",lty="dotted") # line for F=M

text(x=Ldat$Year[nYears],y=LmeanFM,"F=M",adj=c(1,-0.5), col="darkgreen")

if(is.na(dat.ID$Lm50)==F){

 lines(x=Ldat$Year,y=rep(dat.ID$Lm50,nYears),col="darkgreen",lty="dotdash") # line for Lm50

 text(x=Ldat$Year[nYears],y=dat.ID$Lm50,"Lm50", adj=c(1,-0.5), col="darkgreen")

}

}

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

# Plot time series of GLmean relative to Lopt

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

 if(dat.ID$GausSel==T){

 plot(x=Ldat$Year,y=GLmean.ts, bty="l",type="l",

 xlim=c(Ldat$Year[1],Ldat$Year[nYears]),

 xaxt="n",

 ylim=c(0,max(1.1\*median(Ldat$r.Lopt)\*Linf.med,max(GLmean.ts),na.rm=T)),lwd=2,

 xlab="Year",ylab = "Lenght (cm)",main="Lmean vs Lopt")

 axis(1,at=Ldat$Year)

# lines(x=Ldat$Year,y=(GLmean.lcl.ts),lty="dotted")

# lines(x=Ldat$Year,y=GLmean.ucl.ts),lty="dotted")

 lines(x=Ldat$Year,y=rep(Lopt.med,nYears),col="darkgreen") # line for Lopt

 text(x=Ldat$Year[nYears],y=Lopt.med,"Lopt", adj=c(1,-0.5), col="darkgreen")

 lines(x=Ldat$Year,y=rep(LmeanFM,nYears),col="darkgreen",lty="dotted") # line for F=M

 text(x=Ldat$Year[nYears],y=LmeanFM,"F=M",adj=c(1,-0.5), col="darkgreen")

}

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

# Plot time series of F/M

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

plot(x=Ldat$Year,y=FM.ts,

 ylim=c(0,max(max(FM.ucl.ts),1.05)),

 bty="l",type = "l", lwd=1.5, xaxt="n",

 main="previous F/M",xlab="Year",ylab="F/M")

axis(1,at=Ldat$Year)

lines(x=Ldat$Year,y=FM.lcl.ts,lty="dotted")

lines(x=Ldat$Year,y=FM.ucl.ts,lty="dotted")

abline(h=1.0,col="darkgreen")

text(x=Ldat$Year[nYears],y=1,"F=M", adj=c(0.8,-0.5), col="darkgreen")

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

# Plot time series of B/B0

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

plot(x=Ldat$Year,y=BB0.ts,ylim=c(0,min(c(1.1,max(c(0.6,BB0.ucl.ts,1.1\*BFM1B0))))),

 bty="l",type = "l", lwd=1.5, xaxt="n",

 main="exploited B / B0",xlab="Year",ylab="B / B0")

axis(1,at=Ldat$Year)

lines(x=Ldat$Year,y=BB0.lcl.ts,lty="dotted")

lines(x=Ldat$Year,y=BB0.ucl.ts,lty="dotted")

abline(h=1.0,col="darkgreen") # B0

text(x=Ldat$Year[nYears],y=1,"B0", adj=c(0.8,-0.5), col="darkgreen")

lines(x=Ldat$Year,y=rep(BFM1B0,nYears),lty="dashed", col="darkgreen")

text(x=Ldat$Year[nYears-1],y=BFM1B0,"B F=M, Lc=opt", adj=c(0.8,-0.5),col="darkgreen")

lines(x=Ldat$Year,y=rep(BFM1B0/2,nYears),lty="dotted", col="red")

text(x=Ldat$Year[nYears-1],y=BFM1B0/2,"proxy 0.5 Bmsy", adj=c(0.8,-0.5),col="red")

# plot B/B0 range of selected year

if(length(dat.ID$Year.select[dat.ID$Stock==Stock]) != 0 && is.na(dat.ID$Year.select[dat.ID$Stock==Stock])==F) {

 lines(x=c(dat.ID$Year.select[dat.ID$Stock==Stock],dat.ID$Year.select[dat.ID$Stock==Stock]),

 y=c(BB0.lcl.sl,BB0.ucl.sl),col="blue")

}

} # end of loop for plotting time series

##AMSY-code

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

## AMSY

## Original code written by Rainer Froese in January - April 2019

## Additons by Henning Winker:

## 1. Implemented process error on process equation

## 2. Process error is implemented as sigme.R = c(0.05,0.07,0.1,0.15) for Very low, Low, Medium, High.

## 3. Schaefer function now greps sigma.R as defined

## 4. Observation error implemented as CV (log.sd of CPUE)

## 5. Implemented Kobe prototype with terminal F/Fmsy taken as mean of previous 3 yrs.

## 6. Added save.plot option from CMSY

## 7. Added automatic package installer

## Additions by Gianpaolo Coro:

## 1. improved estimate of prior kq

## 2. retrospective analysis

## Addition by RF: MVN

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

# Install required packages if not available

list.of.packages <- c("gplots", "coda","mvtnorm","crayon") #><> mvt not mtv

new.packages <- list.of.packages[!(list.of.packages %in% installed.packages()[,"Package"])]

if(length(new.packages)) install.packages(new.packages)

library(coda)

library("gplots")

library(mvtnorm) # used for Kobe plot, ignore version error

library(crayon) # to display bold and italics in console

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

# Some general settings ----

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

# set.seed(999) # use for comparing results between runs

rm(list=ls(all=FALSE)) # clear previous variables etc

options(digits=3) # displays all numbers with three significant digits as default

graphics.off() # close graphics windows from previous sessions

setwd(dirname(rstudioapi::getActiveDocumentContext()$path)) # set working directory to source file location

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

# Required settings, File names

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

id\_file <- "EU\_Stocks\_ID\_8.csv" #"FirstAss\_ID\_4.csv" # "SimCPUE\_ID\_8.csv" # "CMSY\_ID\_11.csv" # name of file containing stock-specific info and settings for the analysis

outfile <- paste("Out\_",format(Sys.Date(),format="%B%d%Y\_"),id\_file,sep="") # default name for output file

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

# Select stock to be analyzed ----

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

stocks <-"ALB"

# If the input files contain more than one stock, specify below the stock to be analyzed

# If the line below is commented out (#), all stocks in the input file will be analyzed

stocks <- "ALB" #"HL\_VL" #c("anb-78ab") # c("HH\_VL","HL\_VL","HLH\_VL","LH\_VL","LHL\_VL","LL\_VL") # "anb-78ab" #c("HL\_H","HL\_M","HL\_L","HL\_VL") #"HH\_L" #"LHL\_L" #"Micr\_pou\_AD" #"Myxine glutinosa" #"Eut\_gurn\_Balt" # "rjc.27.3a47d" #"PNSK" # "Ille\_coi\_AD" # "WSTM"

# Read data

cinfo <- read.csv(id\_file, header=T, dec=".", stringsAsFactors = FALSE)

cat("File", id\_file, "read successfully","\n")

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

# General settings for the analysis ----

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

smooth.cpue <- T # set to TRUE to apply ksmooth with minimum bandwidth of 3, increased at low r

filter <- TRUE # set to TRUE for Monte Carlo filtering; if FALSE, incease max.viable to 20000

cor.log.rk <- -0.607 #-0.871 # empirical value of log r-k correlation in 140 stocks analyzed with BSM

sigma.r <- c(0.05,0.07,0.1,0.15) # very low, low, medium, high # overall process error for productivity or r

sigma.cpue <- 0.3 # observation error for cpue

n.p <- 50000 # number of r-kq pairs to be analysed; will be doubled if too few viable pairs are found

n.trial <- 30 # times each year is calculated with new random error terms for r and cpue

min.viable <- 20 # minimum number of viable r-kq pairs to be accepted for analysis

max.viable <- 5000 # maximum number of viable r-kq pairs to reduce processing time; set to 20000 if filter==FALSE

creep.graph <- F # plot graph for effort creep correction, if used

do.plots <- T # retrospective analysis does not work if FALSE

write.output <- T # set to TRUE if table with results in output file is wanted

kobe.plot <- T # HW set to TRUE so produce additional kobe status plot

save.plots <- T # set to TRUE to save graphs to JPEG files

close.plots <- T # set to TRUE to close on-screen plots, to avoid "too many open devices" error in batch-processing;

retros <- F # retrospective analysis, requires do.plots <- TRUE

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

# FUNCTIONS ----

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

# Monte Carlo filtering with Schaefer Function ----

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

SchaeferCPUE<-function(yr, cpue, ri, kqi, sigR, filter){

 # create matrix for results

 mdat <- matrix(ncol = (2\*nyr+1))

 colnames(mdat) <- c("rv","kqv",paste("c",yr[1:(nyr-1)],sep=""),paste("b",yr[1:nyr],sep=""))

 for(i in 1:length(ri)) { # for all submitted r-kq pairs

 for(trial in 1:n.trial) { # rerun every r-kq pair several times because error terms per year are random

 # max one succesful run across all years is returned per trial

 cqt <- vector()

 cpuet <- vector()

 FFmsy <- vector()

 break.flag <- FALSE

 for(t in 1:(nyr-1)) { # for all years except the last one, for which catch cannot be calculated

 # assign random error terms to surplus production and to cpue

 err <- exp(rnorm(1,0,sigR)) # set annual error for productivity

 cpuet[t] <- cpue[t]\*exp(rnorm(1,0,sigma.cpue)) # assign error to cpue

 if(cpuet[t]<=0) {cpuet[t] <- 0.01\*kqi[i]} # make sure cpuet is not zero or negative

 # calculate catch

 if(cpuet[t]/kqi[i] >= 0.25) {

 cqt[t] <- (cpuet[t] + cpuet[t] \* ri[i] \* (1-cpuet[t]/kqi[i]))\*err - cpue[t+1] } else {

 cqt[t] <- (cpuet[t] + cpuet[t] \* ri[i] \* (1-cpuet[t]/kqi[i])\*(4\*cpuet[t]/kqi[i]))\*err - cpue[t+1] } # reduce r linearly below 0.25 kq

 # use moving average to smooth hectic catch predictions

 if(t == 2) {cqt[t] <- mean(c(cqt[t-1],cqt[t])) }

 if(t > 2) {cqt[t] <- mean(c(cqt[t-2],cqt[t-1],cqt[t])) }

 # calculate MSYq and F/Fmsy

 MSYq <- ri[i]\*kqi[i]/4

 FFmsy[t] <- 2\*cqt[t]/(ri[i]\*cpue[t])

 if(filter==TRUE) {

 ## Test compatibility of r-kq pairs with general prior popdyn knowledge

 ## If one test fails, break the loop and go to the next trial

 # (1) Exclude r-kq pair if catch is negative (cqt[t] < 0)

 mult.kqi <- ifelse(res=="Very low",-0.06,ifelse(res=="Low",-0.02,0)) #relax rule for Very low and Low resilience

 if(cqt[t] < mult.kqi\*kqi[i]) {break.flag<-TRUE;break}

 # (2) Exclude r-kq pair if catch exceeds biomass (cqt[t] > cpue[t])

 # if lowest cpue is close to zero, skip this test

 if(min(cpue.raw) > 0.1\*max.cpue) {

 # in highly productive species, catch may exceed average annual biomass

 mult.cpue <- ifelse(res=="High",1.4,ifelse(res=="Medium",1,ifelse(res=="Low",0.5,0.25)))

 if(cqt[t] > (mult.cpue\*cpuet[t])) {break.flag<-TRUE;break} }

 # (3) Exclude r-kq pair if catch exceeds MSY

 # some overshooting of MSY is possible

 mult.msy <- ifelse(res=="Very low",10,ifelse(res=="Low",5,ifelse(res=="Medium",3,2)))

 if(cqt[t] > mult.msy\*MSYq) {break.flag<-TRUE;break}

 # (4) Exclude r-k pairs if F/Fmsy is highly unrealistic (negative or much too high)

 FFlow <- ifelse(res=="Very low",-25,-3)

 FFhi <- ifelse(res=="Very low",12,5)

 if(t > 1 && (FFmsy[t-1] < FFlow || FFmsy[t-1] > FFhi)) {break.flag<-TRUE;break}

 # (5) if relative cpue in the year of the B/k prior is outside of the prior range, discard trial

 # relax rule if lower B/k prior range is <= 0.01

 if(prior.Bk[1] <= 0.01) { prior.Bk[1] <- 0.0001}

 if(t==Bk.yr.i && (cpuet[Bk.yr.i]/kqi[i] < prior.Bk[1] || cpuet[Bk.yr.i]/kqi[i] > prior.Bk[2])) {break.flag<-TRUE;break }

 } # end of condition for filtering

 } # end of t-loop through years

 # if t-loop was broken and flag==TRUE do not test further, do not plot points, do not store results

 if(break.flag==TRUE) { next }

 # assign error to last cpue and repeat filter (7) if applicable to last year

 cpuet[nyr] <- cpue[nyr]\*exp(rnorm(1,0,sigma.cpue))

 if(cpuet[nyr]<=0) {cpuet[nyr] <- 0.01\*kqi[i]} # make sure cpuet is not zero or negative

 if(filter==TRUE && Bk.yr.i==nyr && (cpuet[nyr]/kqi[i] < prior.Bk[1] || cpuet[nyr]/kqi[i] > prior.Bk[2])) { next }

 # If all tests are passed, add viable r-kq pair and predicted catch to matrix

 mdat <- rbind(mdat,c(ri[i],kqi[i],cqt[1:(nyr-1)],cpuet[1:(nyr)]))

 # plot viable r-kq pairs

 if(do.plots==T) {

 points(x=ri[i],y=kqi[i], col="grey20", pch=".", cex=2.5) }

 } # end of trial-loop for trials per r-kq pair

 if(length(mdat[,1])>max.viable) { break} # end searching for viable pairs if n > max.viable

 } # end of i-loop through r-kq pairs

 mdat <- na.omit(mdat)

 return(mdat)

} # end of SchaeferCPUE function

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

# Function to create multivariate-normal distribution for r-k

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

mvn <- function(n,mean.log.r,sd.log.r,mean.log.kq,sd.log.kq) {

 cov.log.rk <- cor.log.rk\*sd.log.r\*sd.log.kq # covariance with empirical correlation and prior variances covar.log.rk = matrix(NA, ncol=2,nrow=2) # contract covariance matrix

 covar.log.rk <- matrix(NA, ncol=2,nrow=2) # covariance matrix

 covar.log.rk[1,1] <- sd.log.r^2 # position [1,1] is variance of log.r

 covar.log.rk[2,2] <- sd.log.kq^2 # position [2,2] is variance of log.k

 covar.log.rk[1,2] = covar.log.rk[2,1] = cov.log.rk # positions [1,2] and [2,1] are correlations

 mu.log.rk <- (c(mean.log.r,mean.log.kq)) # vector of log.means

 mvn.log.rk <- rmvnorm(n,mean=mu.log.rk,sigma=covar.log.rk,method="svd")

 return(mvn.log.rk)

}

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

# END OF FUNCTIONS

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

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

# Create table for output to csv file

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

if(write.output==T && substr(id\_file,1,3)=="Sim"){ # output for simulated data

 outheaders = data.frame("Stock","r.true", "r.est","r.lcl","r.ucl",

 "kq.true","kq.est","kq.lcl","kq.ucl",

 "MSYq.true","MSYq.est","MSYq.lcl","MSYq.ucl",

 "FFmsy.true","FFmsy.est","FFmsy.lcl","FFmsy.ucl",

 "BBmsy.true","BBmsy.est","BBmsy.lcl","BBmsy.ucl")

 write.table(outheaders,file=outfile, append = T, sep=",",row.names=F,col.names=F) }

if(write.output==TRUE && is.null(cinfo$MSY.BSM)==F) { # assuming all BSM fields are available

 outheaders = data.frame("Stock","r.BSM","lcl","ucl","r.est","lcl","ucl","k.BSM","lcl","ucl","k.est","lcl","ucl",

 "BBmsy.BSM","lcl","ucl","BBmsy.est","lcl","ucl",

 "FFmsy.BSM","lcl","ucl",

 "FFmsy.est","lcl","ucl")

 write.table(outheaders,file=outfile, append = T, sep=",",row.names=F,col.names=F)

}

if(write.output==T && substr(id\_file,1,3)!="Sim" && is.null(cinfo$MSY.BSM)==T){

 outheaders = data.frame("Stock","Fmsy.est","Fmsy.lcl","Fmsy.ucl",

 "FFmsy.est","FFmsy.lcl","FFmsy.ucl",

 "BBmsy.est","BBmsy.lcl","BBmsy.ucl")

 write.table(outheaders,file=outfile, append = T, sep=",",row.names=F,col.names=F) }

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

# Start output to screen

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

cat("------------------------------------------------------------\n")

cat("AMSY Analysis,", date(),"\n")

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

# Analyze stock(s)

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

if(is.na(stocks[1])==TRUE){

 stocks <- as.character(cinfo$Stock) # Analyze stocks in sequence of ID file

 # stocks <- cinfo$Stock[cinfo$Stock >= "fle-2425"] # Analyze stocks in sequence of ID file

 # stocks <- sort(as.character(cinfo$Stock)) # Analyze stocks in alphabetic order

}

# analyze one stock after the other

for(stock in stocks) {

 #retrospective analysis

 if(retros==T && (cinfo$EndYear[cinfo$Stock==stock]-cinfo$Bk.yr[cinfo$Stock==stock])<3) {

 retros.nyears<-0 #retrospective analysis

 cat("Warning: Retrospective analysis not meaningful and omitted if B/k prior is in the final year(s)\n") } else {

 retros.nyears<-ifelse(retros==T,3,0) #retrospective analysis

 }

 FFmsy.retrospective<-list() #retrospective analysis

 BBmsy.retrospective<-list() #retrospective analysis

 years.retrospective<-list() #retrospective analysis

 for (retrosp.step in 0:retros.nyears){ #retrospective analysis

 cat("------------------------------------------------------------\n")

 cat("Stock ",bold(stock),", ", bold(italic(as.character(cinfo$ScientificName[cinfo$Stock==stock]))),", ",

 as.character(cinfo$EnglishName[cinfo$Stock==stock]),sep="","\n")

 # read data for stock

 cpue\_file <- cinfo$CPUE\_File[cinfo$Stock==stock]

 cdat <- read.csv(cpue\_file, header=T, dec=".", stringsAsFactors = FALSE)

 # assign data from cinfo to vectors

 n <- n.p

 res <- as.character(cinfo$Resilience[cinfo$Stock==stock])

 res.i <- which(c("Very low","Low","Medium","High")%in%res) # determines process error strength

 if(length(res.i)==0) {stop("Spelling error in resilience in ID file\n")}

 start.yr <- as.numeric(cinfo$StartYear[cinfo$Stock==stock])

 end.yr <- as.numeric(cinfo$EndYear[cinfo$Stock==stock])

 end.yr <- end.yr-retrosp.step #retrospective analysis

 r.low <- as.numeric(cinfo$r.low[cinfo$Stock==stock])

 r.hi <- as.numeric(cinfo$r.hi[cinfo$Stock==stock])

 user.log.r <- ifelse(is.na(r.low)==F & is.na(r.hi)==F,TRUE,FALSE)

 Bk.yr <- as.numeric(cinfo$Bk.yr[cinfo$Stock==stock])

 Bk.pr <- as.character(cinfo$Bk.pr[cinfo$Stock==stock])

 Bk.pr.low <- as.numeric(cinfo$Bk.pr.low[cinfo$Stock==stock])

 Bk.pr.hi <- as.numeric(cinfo$Bk.pr.hi[cinfo$Stock==stock])

 e.creep <- as.numeric(cinfo$e.creep[cinfo$Stock==stock])

 comment <- as.character(cinfo$Comment[cinfo$Stock==stock])

 Fmsy.ass <- as.numeric(cinfo$Fmsy.ass[cinfo$Stock==stock])

 Bmsy.ass <- as.numeric(cinfo$Bmsy.ass[cinfo$Stock==stock])

 source <- as.character(cinfo$Source[cinfo$Stock==stock])

 # check for common errors

 if(length(r.low)==0){

 cat("ERROR: Could not find the stock in the ID input file - check that the stock names match in ID and CPUE files and that commas are used (not semi-colon)")

 return (NA) }

 if(length(cdat$Year[cdat$Stock==stock])==0){

 cat("ERROR: Could not find the stock in the CPUE file - check that the stock names match in ID and CPUE files and that commas are used (not semi-colon)")

 return (NA) }

 if(start.yr < cdat$Year[cdat$Stock==stock][1]){

 cat("ERROR: start year in ID file before first year in CPUE file\n")

 return (NA)}

 if(!(Bk.pr %in% c("Near unexploited","More than half","About half","Small","Very small"))){

 cat("ERROR: Prior for stock size not in: Near unexploited, More than half, About half, Small, Very small\n")

 return (NA)}

 if(Bk.yr < start.yr || Bk.yr > end.yr){

 cat("ERROR: Year for B/k prior outside range of years\n")

 return (NA)}

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

 # Determine initial ranges for r

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

 # initial range of r from input file

 if(is.na(r.low)==F & is.na(r.hi)==F) {

 prior.r <- c(r.low,r.hi)

 } else {

 # initial range of r based on resilience

 if(res == "High") {

 prior.r <- c(0.6,1.5)} else if(res == "Medium") {

 prior.r <- c(0.2,0.8)} else if(res == "Low") {

 prior.r <- c(0.05,0.5)} else { # i.e. res== "Very low"

 prior.r <- c(0.015,0.1)}

 }

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

 # extract data on stock

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

 yr <- as.numeric(cdat$Year[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr])

 nyr <- length(yr) # number of years in the time series

 Bk.yr.i <- which(yr==Bk.yr)

 cpue.raw <- as.numeric(cdat$CPUE[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr])

 # get catch from full assessments or simulations if available, for comparison

 if(is.null(cdat$Catch[cdat$Stock==stock][1])==F && is.na(cdat$Catch[cdat$Stock==stock][1])==F){

 C.ass <- cdat$Catch[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr] } else {C.ass <- NA}

 # apply correction for effort-creep to commercial(!) CPUE if indicated by user

 if(is.na(e.creep)==FALSE) {

 cpue.cor <- cpue.raw

 for(i in 1:(length(cpue.raw)-1)) {

 cpue.cor[i+1] <- cpue.raw[i+1]\*(1-e.creep/100)^i # equation for decay in %; first cpue without correction

 }

 if(creep.graph==TRUE) {

 windows(8,6)

 plot(x=yr,y=cpue.raw,ylim=c(0,max(cpue.raw)),type="l",bty="l",xlab="Year",ylab="CPUE")

 lines(x=yr,y=cpue.cor,col="red")

 text(x=yr[length(yr)/2],y=max(cpue.raw),paste(stock," CPUE corrected for effort creep of ",e.creep," %",sep=""),col="red")

 }

 cpue.raw <- cpue.cor

 }

 d.cpue.raw <- max(diff(cpue.raw)/cpue.raw[1:(nyr-1)])

 if(smooth.cpue==T||d.cpue.raw > 1.5) {

 smooth.flag <- TRUE

 bw <- log(2)/exp(mean(log(prior.r))) # use population doubling time as bandwidth

 bw <- ifelse(bw < 3,3,bw) # enforce minimum bandwidth of 3

 cpue <- ksmooth(x=yr,y=cpue.raw,kernel="normal",n.points=length(yr),bandwidth=bw)$y } else {

 smooth.flag <- FALSE

 cpue <- cpue.raw }

 # use median of 3 largest cpue as max cpue

 max.cpue <- sort(cpue)[length(cpue)-1]

 min.cpue <- min(cpue)

 if(length(Fmsy.ass)>0 && is.null(Fmsy.ass)==F && is.na(Fmsy.ass)==F && is.null(cdat$F[1])==F) {

 FFmsy.ass <- as.numeric(cdat$F[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr]/Fmsy.ass)

 if(is.null(cdat$FLow[1])==F) {FFmsy.ass.lcl <- as.numeric(cdat$FLow[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr]/Fmsy.ass) }

 if(is.null(cdat$FHi[1])==F) {FFmsy.ass.ucl <- as.numeric(cdat$FHi[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr]/Fmsy.ass) }

 } else {Fmsy.ass <- NA; FFmsy.ass <- NA;FFmsy.ass.lcl <- NA;FFmsy.ass.ucl <- NA}

 if(length(Bmsy.ass)>0 && is.null(Bmsy.ass)==F && is.na(Bmsy.ass)==F && is.null(cdat$B[1])==F) {

 BBmsy.ass <- as.numeric(cdat$B[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr]/Bmsy.ass)

 if(is.null(cdat$BLow[1])==F) {BBmsy.ass.ucl <- as.numeric(cdat$BLow[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr]/Bmsy.ass) }

 if(is.null(cdat$BHi[1])==F) {BBmsy.ass.lcl <- as.numeric(cdat$BHi[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr]/Bmsy.ass) }

 } else {Bmsy.ass <- NA; BBmsy.ass <- NA;BBmsy.ass.lcl <- NA;BBmsy.ass.ucl <- NA}

 if(length(yr)==0){

 cat("ERROR: Could not find the stock in the CPUE input files - Please check that the stock ID is written correctly")

 return (NA) }

 if(length(yr) != (end.yr-start.yr+1)) {

 cat("ERROR: indicated year range is of different length than years in CPUE file\n")

 return (NA)}

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

 # Determine ranges for relative biomass

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

 # initial range of B/k from input file

 if(is.na(Bk.pr.low)==F & is.na(Bk.pr.hi)==F) {

 prior.Bk <- c(Bk.pr.low,Bk.pr.hi)

 } else {

 if(Bk.pr == "Near unexploited") {

 prior.Bk <- c(0.75,1.0)} else if(Bk.pr == "More than half") {

 prior.Bk <- c(0.5,0.85)} else if(Bk.pr == "About half") {

 prior.Bk <- c(0.35,0.65)} else if(Bk.pr == "Small") {

 prior.Bk <- c(0.15,0.4)} else { # i.e. Bk.pr== "Very small"

 prior.Bk <- c(0.01,0.2)}

 }

 # us relative range of B/k as relative range for kq

 mean.prior.Bk <- mean(prior.Bk)

 rr.prior.Bk <- (mean.prior.Bk-prior.Bk[1])/mean.prior.Bk

 prior.kq.low.1 <- (1-rr.prior.Bk)\*cpue[Bk.yr.i]/mean.prior.Bk

 prior.kq.hi.1 <- (1+rr.prior.Bk)\*cpue[Bk.yr.i]/mean.prior.Bk

 # kq must be > max cpue unless near unexploited

 prior.kq.low.2 <- ifelse(prior.kq.low.1 < max.cpue,ifelse(mean.prior.Bk >= 0.85,0.9\*max.cpue,max.cpue),prior.kq.low.1)

 # increase lower kq prior if cpue is small and flat

 if((max(cpue)/min(cpue))<2) {

 prior.kq.low <- ifelse(mean.prior.Bk < 0.3,2\*prior.kq.low.2,

 ifelse(mean.prior.Bk < 0.6,1.5\*prior.kq.low.2,prior.kq.low.2))

 } else {prior.kq.low <- prior.kq.low.2 }

 # kq.hi at least 30-50% larger than kq.low, depending on Bk prior

 if(mean.prior.Bk >= 0.6) {

 prior.kq.hi.2 <- ifelse(prior.kq.hi.1 < (1.3\*prior.kq.low),1.3\*prior.kq.low,prior.kq.hi.1) } else {

 prior.kq.hi.2 <- ifelse(prior.kq.hi.1 < (1.5\*prior.kq.low),1.5\*prior.kq.low,prior.kq.hi.1) }

 # if upper prior kq is too hi, limit to 3 times lower range

 prior.kq.hi <- ifelse(prior.kq.hi.2 > (3\*prior.kq.low),3\*prior.kq.low,prior.kq.hi.2)

 prior.kq <- c(prior.kq.low,prior.kq.hi)

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

 # Sampling of r-k space

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

 # turn numerical ranges into log-normal distributions

 mean.log.r=mean(log(prior.r))

 sd.log.r=(log(prior.r[2])-log(prior.r[1]))/4 # assume range covers 4 SD

 mean.log.kq <- mean(log(prior.kq))

 sd.log.kq <- (log(prior.kq[2])-log(prior.kq[1]))/4 # assume range covers 4 SD

 mvn.log.rk <- mvn(n=n,mean.log.r=mean.log.r,sd.log.r=sd.log.r,mean.log.kq=mean.log.kq,sd.log.kq=sd.log.kq)

 ri1 <- exp(mvn.log.rk[,1])

 kqi1 <- exp(mvn.log.rk[,2])

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

 # print prior info on screen

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

 cat("CPUE data for years ",yr[1]," - ",yr[nyr],", CPUE range ",min.cpue," - ",max(cpue),", smooth = ",smooth.flag,sep="","\n")

 cat("Prior for r = ",res,", ", r.low," - ",r.hi,sep="","\n")

 if(is.na(r.low)==T) {

 cat("Used prior range for r = ", prior.r[1]," - ",prior.r[2],sep="","\n") } else {

 cat("Used prior range for r = ", quantile(ri1,0.01)," - ",quantile(ri1,0.99),sep="","\n") }

 cat("Prior for ",Bk.yr," stock status = ", Bk.pr,", ",Bk.pr.low," - ",Bk.pr.hi,sep="","\n")

 cat("Used ",Bk.yr," prior B/B0 range = ",prior.Bk[1]," - ",prior.Bk[2],", prior B/Bmsy = ",2\*prior.Bk[1]," - ",2\*prior.Bk[2],sep="","\n")

 cat("Used prior range for kq = ",prior.kq[1]," - ",prior.kq[2]," [original range = ",prior.kq.low.1," - ",prior.kq.hi.1,"]\n",sep="")

 if(is.na(Fmsy.ass)==F) {cat("Assessment Fmsy =",Fmsy.ass,"\n")}

 if(is.na(FFmsy.ass[1])==F) {cat("Assessment F/Fmsy =",FFmsy.ass[nyr-1],ifelse(is.na(FFmsy.ass.lcl[1])==F,

 paste(",",format(FFmsy.ass.lcl[nyr-1],digits = 2),"-",format(FFmsy.ass.ucl[nyr-1],digits=2),"")),

 "(",yr[nyr-1],")\n")}

 if(is.na(Bmsy.ass)==F) {cat("Assessment proxy Bmsy =",Bmsy.ass,"\n")}

 if(is.na(BBmsy.ass[1])==F) {cat("Assessment proxy B/Bmsy =",BBmsy.ass[nyr],ifelse(is.na(BBmsy.ass.lcl[1])==F,

 paste(",",format(BBmsy.ass.lcl[nyr],digits = 2),"-",format(BBmsy.ass.ucl[nyr],digits=2),"")),

 "(",yr[nyr],")\n")

 cat("Source:",source,"\n")}

 if(is.null(cinfo$MSY.BSM[cinfo$Stock==stock])==F && is.na(cinfo$MSY.BSM[cinfo$Stock==stock])==F) { # assume all info from BSM analysis is available

 cat("BSM r =", cinfo$r.BSM[cinfo$Stock==stock],",",cinfo$r.BSM.lcl[cinfo$Stock==stock],

 "-",cinfo$r.BSM.ucl[cinfo$Stock==stock],"\n")

 cat("BSM k =", cinfo$k.BSM[cinfo$Stock==stock]\*1000,",",cinfo$k.BSM.lcl[cinfo$Stock==stock]\*1000,

 "-",cinfo$k.BSM.ucl[cinfo$Stock==stock]\*1000,"\n")

 cat("BSM MSY =", cinfo$MSY.BSM[cinfo$Stock==stock]\*1000,",",cinfo$MSY.BSM.lcl[cinfo$Stock==stock]\*1000,

 "-",cinfo$MSY.BSM.ucl[cinfo$Stock==stock]\*1000, "\n")

 cat("BSM last B/Bmsy =", cinfo$B\_Bmsy[cinfo$Stock==stock],",",cinfo$B\_Bmsy.lcl[cinfo$Stock==stock],

 "-",cinfo$B\_Bmsy.ucl[cinfo$Stock==stock], "\n")

 cat("BSM last F/Fmsy =", cinfo$F\_Fmsy[cinfo$Stock==stock],",",cinfo$F\_Fmsy.lcl[cinfo$Stock==stock],

 "-",cinfo$F\_Fmsy.ucl[cinfo$Stock==stock], "\n")

 }

 cat("Comment:",comment,"\n")

 if(is.na(source)==F) {cat("Source:",source,"\n")}

 cat("\n")

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

 # Plot CPUE data and prior CPUE\_msy

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

 if(close.plots==T) {graphics.off()} # close previous plots, e.g. in batch processing

 if(do.plots==T) {

 # check for operating system, open separate window for graphs if Windows

 if(grepl("win",tolower(Sys.info()['sysname']))) {windows(14,9)}

 par(mfrow=c(2,3))

 # (a): plot CPUE with prior for CPUEmsy

 plot(x=yr, y=cpue,

 ylim=c(0,max(ifelse(substr(id\_file,1,3)=="Sim",1.1\*cinfo$true.MSYq,0),1.2\*max(cpue),

 1.2\*max(cpue.raw),prior.kq[2])),

 type ="l", bty="l", main=paste("(a) CPUE", stock), xlab="Year",

 ylab="CPUE", lwd=0.5, cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.5)

 lines(x=yr,y=cpue.raw,lwd=2)

 # arrows(x0=Bk.yr,y0=(1+cv.prior.Bk)\*cpue[Bk.yr.i],x1=Bk.yr,y1=(1-cv.prior.Bk)\*cpue[Bk.yr.i],length=0.05,angle = 90, code=3)# plot prior range for B/k

 arrows(x0=Bk.yr,y0=prior.kq[1],x1=Bk.yr,y1=prior.kq[2],length=0.05,angle = 90, code=3,col="blue")# plot prior range for B/k

 lines(x=c(yr[1],yr[nyr]),y=c(prior.kq.low/2,prior.kq.low/2),lty="dotted")

 lines(x=c(yr[1],yr[nyr]),y=c(prior.kq.hi/2,prior.kq.hi/2),lty="dotted")

 text(x=yr[nyr-as.integer(0.15\*nyr)],y=(prior.kq.low+prior.kq.hi)/4,"Bmsy\_q")

 # (b): plot r-k graph

 plot(x=ri1, y=kqi1, xlim = c(0.95\*quantile(ri1,0.001),1.2\*quantile(ri1,0.999)),

 ylim = c(0.95\*quantile(kqi1,0.001),1.2\*quantile(kqi1,0.999)),

 log="xy", xlab="r", ylab="kq", main="(b) Finding viable r-kq", pch=".", cex=2, bty="l",

 col="grey90", cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.5)

 lines(x=c(prior.r[1],prior.r[2],prior.r[2],prior.r[1],prior.r[1]), # plot original prior range

 y=c(prior.kq[1],prior.kq[1],prior.kq[2],prior.kq[2],prior.kq[1]),

 lty="dotted")

 } # end of do.plots loop

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

 # Call AMSY-Schaefer function to filter r-kq space for viable r-kq pairs

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

 cat("Monte Carlo filtering of r-kq space with",n,"points and",n.trial,"error patterns. \n")

 MCA1 <- SchaeferCPUE(yr=yr, cpue=cpue, ri=ri1, kqi=kqi1, sigR=sigma.r[res.i], filter=filter) #><> correct PE input

 n.viable <- length(MCA1[,"rv"])

 cat("Viable r-kq pairs =",n.viable,"\n")

 if((n.viable)<min.viable) {

 cat("Too few r-kq pairs after filtering, repeating analysis with 2 times more pairs, extended prior ranges, and increased smoothing:\n")

 mvn.log.rk2 <- mvn(n=2\*n,mean.log.r=mean.log.r,sd.log.r=1.2\*sd.log.r,mean.log.kq=mean.log.kq,sd.log.kq=1.2\*sd.log.kq)

 ri2 <- exp(mvn.log.rk2[,1])

 kqi2 <- exp(mvn.log.rk2[,2])

 points(x=ri2, y=kqi2,pch=".",cex=2,col="grey90") # plot new potential r-k pairs

 lines(x=c(prior.r[1],prior.r[2],prior.r[2],prior.r[1],prior.r[1]), # re-plot original prior range

 y=c(prior.kq[1],prior.kq[1],prior.kq[2],prior.kq[2],prior.kq[1]),

 lty="dotted")

 points(MCA1,col="grey20", pch=".", cex=2.5) # re-plot points found so far

 text(x=prior.r[2],y=1.15\*prior.kq[2],"extended")

 cpue <- ksmooth(x=yr,y=cpue.raw,kernel="normal",n.points=length(yr),bandwidth=5)$y

 MCA2 <- SchaeferCPUE(yr=yr, cpue=cpue, ri=ri2, kqi=kqi2, sigR=sigma.r[res.i], filter=filter)

 MCA <- rbind(MCA1,MCA2)

 n.viable <- length(MCA[,"rv"])

 cat("Viable r-kq pairs =",n.viable,"\n") } else {MCA <- MCA1}

 if((n.viable)<10) {

 cat("Too few r-kq pairs after filtering, doing analysis without filters:\n")

 MCA <- SchaeferCPUE(yr=yr, cpue=cpue, ri=ri2, kqi=kqi2, sigR=sigma.r[res.i], filter=FALSE)

 text(x=prior.r[1],y=0.9\*prior.kq[1],"no filters")

 }

 rv <- MCA[,"rv"]

 kqv <- MCA[,"kqv"]

 MSYqv <- rv \* kqv / 4

 MSYq.est <- median(MSYqv)

 MSYq.lcl <- as.numeric(quantile(MSYqv,0.025))

 MSYq.ucl <- as.numeric(quantile(MSYqv,0.975))

 n.v <- length(MSYqv)

 kqv.est <- median(kqv)

 kqv.lcl <- as.numeric(quantile(kqv,0.025))

 kqv.ucl <- as.numeric(quantile(kqv,0.975))

 rv.est <- 4\*MSYq.est/kqv.est # rv corresponding to median(kqv)

 rv.lcl <- as.numeric(quantile(rv,0.025))

 rv.ucl <- as.numeric(quantile(rv,0.975))

 cqt.sel <- matrix(nrow=length(rv),ncol=nyr-1)

 colnames(cqt.sel) <- c(yr[1:nyr-1])

 for(j in 1:(nyr-1)) {

 cqt.sel[,j] <- MCA[,j+2]}

 cqt.median <- apply(cqt.sel,2,median)

 cqt.lcl <- apply(cqt.sel,2,quantile,probs=0.025)

 cqt.ucl <- apply(cqt.sel,2,quantile,probs=0.975)

 cpuet.sel <- matrix(nrow=length(rv),ncol=nyr)

 colnames(cpuet.sel) <- c(yr[1:nyr])

 for(j in 1:(nyr)) {

 cpuet.sel[,j] <- MCA[,j+2+nyr-1]}

 cpuet.median <- apply(cpuet.sel,2,median)

 cpuet.lcl <- apply(cpuet.sel,2,quantile,probs=0.025)

 cpuet.ucl <- apply(cpuet.sel,2,quantile,probs=0.975)

 BBmsy.end <- cpuet.median[nyr]/(kqv.est/2)

 BBmsy.end.lcl <- cpuet.lcl[nyr]/(kqv.est/2)

 BBmsy.end.ucl <- cpuet.ucl[nyr]/(kqv.est/2)

 Ft <- cqt.median[1:(nyr-1)]/cpuet.median[1:(nyr-1)]

 FFmsy <- Ft/(0.5\*rv.est)

 FFmsy.end <- FFmsy[nyr-1]

 Ft.lcl <- cqt.lcl[1:(nyr-1)]/cpuet.median[1:(nyr-1)]

 FFmsy.lcl <- Ft.lcl/(0.5\*rv.est)

 FFmsy.end.lcl <- FFmsy.lcl[nyr-1]

 Ft.ucl <- cqt.ucl[1:(nyr-1)]/cpuet.median[1:(nyr-1)]

 FFmsy.ucl <- Ft.ucl/(0.5\*rv.est)

 FFmsy.end.ucl <- FFmsy.ucl[nyr-1]

 if(substr(id\_file,1,3)=="Sim"){ # if dealing with simulated data, get the "true" values

 MSYq.true <- cinfo$true.MSYq[cinfo$Stock==stock]

 r.true <- as.numeric(cinfo$true.r[cinfo$Stock==stock])

 kq.true <- as.numeric(cinfo$true.kq[cinfo$Stock==stock])

 cqt.true <- as.numeric(cdat$Catch[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr])

 Ft.true <- cqt.true/cpuet.median

 FFmsy.true <- Ft.true/(0.5\*cinfo$true.r[cinfo$Stock==stock])

 FFmsy.end.true <- FFmsy.true[nyr-1]

 BBmsy.end.true <- cinfo$true.Bk.end[cinfo$Stock==stock]\*2

 } else {FFmsy.true <- NA; FFmsy.end.true <- NA; cqt.true <- NA}

 cat("\n Results:",

 "\n viable r-kq pairs = ",n.v,

 "\n median kq = ",kqv.est,", ",kqv.lcl," - ",kqv.ucl,

 "\n median MSYq = ",MSYq.est,", ",MSYq.lcl," - ",MSYq.ucl,

 "\n r (4 MSYq/kq) = ",rv.est,", ",rv.lcl," - ",rv.ucl,

 "\n Fmsy (r/2) = ",rv.est/2,", ",rv.lcl/2," - ",rv.ucl/2,

 "\n F/Fmsy = ",FFmsy.end,", ",FFmsy.end.lcl," - ",FFmsy.end.ucl," (",yr[nyr-1],")",

 ifelse(substr(id\_file,1,3)=="Sim",paste(", true:",format(FFmsy.end.true,digits = 3)),""),

 "\n B/Bmsy = ",BBmsy.end,", ",BBmsy.end.lcl," - ",BBmsy.end.ucl," (",yr[nyr],")",

 ifelse(substr(id\_file,1,3)=="Sim",paste(", true:",format(BBmsy.end.true,digits = 3)),""),

 "\n",sep="")

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

 # Plot results

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

 if(do.plots==T) {

 # (b): continued..

 # Add estimated r-k with confidence limits, for comparison with prior range

 points(x=rv.est,y=kqv.est,col="red3",lwd=2) # most probably r-kq pair estimate

 lines(x=c(rv.lcl,rv.ucl),y=c(kqv.est,kqv.est),col="red3",lwd=2) # confidence limits

 lines(x=c(rv.est,rv.est),y=c(kqv.lcl,kqv.ucl),col="red3",lwd=2) # confidence limits

 # Add true r-kq to simulated data in plot B:

 if(substr(id\_file,1,3)=="Sim") points(x=cinfo$true.r[cinfo$Stock==stock],y=cinfo$true.kq[cinfo$Stock==stock],col="blue",cex=2,lwd=2)

 if(substr(id\_file,1,4)=="CMSY") points(x=cinfo$r.BSM[cinfo$Stock==stock],y=1000\*cinfo$k.BSM[cinfo$Stock==stock],col="blue",cex=2,lwd=2)

 # (c): Analysis of viable r-k plot

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

 plot(x=MCA[,"rv"], y=MCA[,"kqv"],

 xlim=c(min(c(rv,cinfo$true.r[cinfo$Stock==stock],rv.lcl),na.rm=T),max(c(rv,cinfo$true.r[cinfo$Stock==stock],rv.ucl),na.rm=T)),

 ylim=c(min(c(kqv,cinfo$true.kq[cinfo$Stock==stock],kqv.lcl),na.rm=T),max(c(kqv,cinfo$true.kq[cinfo$Stock==stock],kqv.ucl),na.rm=T)),

 pch=".", cex=2.5, col="grey20", log="xy", bty="l",

 xlab="r", ylab="kq", main="(c) Analysis of viable r-kq", cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.5)

 # points(x=rv,y=kqv,pch=".", cex=2.5, col="indianred1") # selected points for deriving estimates

 points(x=rv.est,y=kqv.est,col="red3",lwd=2) # most probably r-kq pair estimate

 lines(x=c(rv.lcl,rv.ucl),y=c(kqv.est,kqv.est),col="red3",lwd=2) # confidence limits

 lines(x=c(rv.est,rv.est),y=c(kqv.lcl,kqv.ucl),col="red3",lwd=2) # confidence limits

 if(substr(id\_file,1,3)=="Sim") points(x=cinfo$true.r[cinfo$Stock==stock],y=cinfo$true.kq[cinfo$Stock==stock],col="blue",cex=2,lwd=2) # true r-kq pair

 # plot r-k from BSM if available

 if(is.null(cinfo$r.BSM[cinfo$Stock==stock])==F) {

 abline(v=cinfo$r.BSM[cinfo$Stock==stock],lty="dashed",col="blue")

 }

 # plot r-k confidence limits if CPUE test file is available

 if(substr(id\_file,1,4)=="CMSY") {

 points(x=cinfo$r.BSM[cinfo$Stock==stock],y=cinfo$k.BSM[cinfo$Stock==stock]\*1000,col="blue",lwd=2) # most probably r-kq pair estimate

 lines(x=c(cinfo$r.BSM.lcl[cinfo$Stock==stock],cinfo$r.BSM.ucl[cinfo$Stock==stock]),

 y=c(cinfo$k.BSM[cinfo$Stock==stock]\*1000,cinfo$k.BSM[cinfo$Stock==stock]\*1000),col="blue",lwd=2) # confidence limits

 lines(x=c(cinfo$r.BSM[cinfo$Stock==stock],cinfo$r.BSM[cinfo$Stock==stock]),

 y=c(cinfo$k.BSM.lcl[cinfo$Stock==stock]\*1000,cinfo$k.BSM.ucl[cinfo$Stock==stock]\*1000),col="blue",lwd=2) # confidence limits

 }

 # (d) Pred. rel. catch plot

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

 # get data from full assessments if available

 C\_MSY.ass <- NA

 if(is.na(C.ass[1])==F && is.na(Fmsy.ass)==F && is.na(Bmsy.ass)==F) {

 MSY.ass <- Fmsy.ass\*Bmsy.ass

 C\_MSY.ass <- C.ass/MSY.ass

 }

 # get data from BSM if available

 if(is.null(cinfo$MSY.BSM[cinfo$Stock==stock])==F) {

 C\_MSY.ass <- C.ass/(cinfo$MSY.BSM[cinfo$Stock==stock]\*1000)

 }

 # determine height of y-axis in plot

 max.y <- max(c(cqt.ucl/MSYq.est,1.1,cqt.true/MSYq.est,C\_MSY.ass), na.rm=T)

 # Main plot of relative CMSY catch up to last year because the schaefer equation is not reliable in nyr

 plot(x=yr[1:(nyr-1)],y=cqt.median[1:(nyr-1)]/MSYq.est, lwd=2, xlab="Year", ylab="C/MSY", type="l",

 ylim=c(0,max.y), bty="l", main="(d) Catch/MSY", cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.5)

 lines(x=yr[2:(nyr-1)], y=cqt.lcl[2:(nyr-1)]/MSYq.est,type="l",lty="dotted")

 lines(x=yr[2:(nyr-1)], y=cqt.ucl[2:(nyr-1)]/MSYq.est,type="l",lty="dotted")

 lines(x=c(yr[1],yr[nyr-1]),y=c(1,1), lty="dashed", lwd=1.5) # line indicating MSY

 text(x=yr[nyr-as.integer(0.1\*nyr)],y=1.1, "MSY")

 # plot true catch \* q from simulations

 if(substr(id\_file,1,3)=="Sim") {

 cqt.true <- as.numeric(cdat$Catch[cdat$Stock==stock & cdat$Year >= start.yr & cdat$Year <= end.yr])

 lines(x=yr[1:(nyr-1)], y=cqt.true[1:(nyr-1)]/MSYq.true,col="blue") }

 # plot catch/MSY from BSM of full assessments

 if(is.na(C.ass[1])==F && is.null(cinfo$r.BSM[cinfo$Stock==stock])==F) {

 lines(x=yr[1:(nyr-1)], y=C\_MSY.ass[1:(nyr-1)],col="blue")

 }

 # plot (e): F/Fmsy

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

 max.y <- max(c(1.2,FFmsy.ucl,FFmsy.true,FFmsy.ass.ucl),na.rm=T)

 plot(x=yr[1:(nyr-1)],y=FFmsy,

 ylim=c(0,max.y),

 lwd=2, xlab="Year", ylab="F/Fmsy", type="l",

 bty="l", main="(e) F/Fmsy", cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.5)

 lines(x=yr[2:(nyr-1)],y=FFmsy.lcl[2:(nyr-1)],lty="dotted") # lcl of F/Fmsy

 lines(x=yr[2:(nyr-1)],y=FFmsy.ucl[2:(nyr-1)],lty="dotted") # ucl of F/Fmsy

 lines(x=c(yr[1],yr[nyr-1]),y=c(1,1),lty="dashed") # indicates F/Fmsy = 1

 if(substr(id\_file,1,3)=="Sim") {lines(x=yr[1:(nyr-1)],y=FFmsy.true[1:(nyr-1)], col="blue")} # true F/Fmsy from simulations

 if(is.na(FFmsy.ass[1])==F) {lines(x=yr[1:(nyr-1)],y=FFmsy.ass[1:(nyr-1)], col="blue")} # F/Fmsy from assessments

 if(is.na(FFmsy.ass.ucl[1])==F) {lines(x=yr[1:(nyr-1)],y=FFmsy.ass.ucl[1:(nyr-1)], lty="dotted",col="blue")} # F/Fmsy from assessments

 if(is.na(FFmsy.ass.lcl[1])==F) {lines(x=yr[1:(nyr-1)],y=FFmsy.ass.lcl[1:(nyr-1)], lty="dotted",col="blue")} # F/Fmsy from assessments

 # plot F/Fmsy from BSM

 if(is.null(cinfo$F\_Fmsy[cinfo$Stock==stock])==F) {

 points(x=yr[nyr-1],y=cinfo$F\_Fmsy[cinfo$Stock==stock],col="blue")

 lines(x=c(yr[nyr-1],yr[nyr-1]),y=c(cinfo$F\_Fmsy.lcl[cinfo$Stock==stock],

 cinfo$F\_Fmsy.ucl[cinfo$Stock==stock]),col="blue")

 }

 # plot F/Fmsy from BSM if CMSY file

 if(is.na(C.ass[1])==F && is.null(cinfo$r.BSM[cinfo$Stock==stock])==F && substr(id\_file,1,4)=="CMSY") {

 F.BSM <- C.ass[1:(nyr-1)] / cpue[1:(nyr-1)]

 FFmsy.BSM <- F.BSM / (cinfo$r.BSM[cinfo$Stock==stock]/2)

 lines(x=yr[1:(nyr-1)], y=FFmsy.BSM[1:(nyr-1)],col="blue")

 }

 # plot (f): B/Bmsy

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

 Bkt <- cpuet.median/(0.5\*kqv.est)

 Bmsy.true <- cinfo$true.kq[cinfo$Stock==stock]/kqv.est

 if(is.na(C.ass[1])==F && is.null(cinfo$r.BSM[cinfo$Stock==stock])==F && substr(id\_file,1,4)=="CMSY") {

 BBmsy.BSM <- cpue.raw / (cinfo$k.BSM[cinfo$Stock==stock]\*1000 / 2) } else { BBmsy.BSM <- NA }

 max.y <- max(c(Bkt, kqv.ucl/kqv.est,Bmsy.true,cpue.raw/(kqv.est/2),cpuet.ucl/(0.5\*kqv.est),BBmsy.ass.ucl,BBmsy.BSM),na.rm=T)

 plot(x=yr,y=Bkt,type="l",ylim=c(0,max.y),

 bty="l",lwd=2,xlab="Year",ylab="B/Bmsy",

 main="(f) B/Bmsy", cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.5)

 lines(x=yr[2:nyr],y=cpuet.lcl[2:nyr]/(0.5\*kqv.est),lty="dotted")

 lines(x=yr[2:nyr],y=cpuet.ucl[2:nyr]/(0.5\*kqv.est),lty="dotted")

 lines(x=c(yr[1],yr[nyr]),y=c(1,1),lty="dashed")

 lines(x=c(yr[1],yr[nyr]),y=c(kqv.lcl/kqv.est,kqv.lcl/kqv.est),lty="dotted")

 lines(x=c(yr[1],yr[nyr]),y=c(kqv.ucl/kqv.est,kqv.ucl/kqv.est),lty="dotted")

 lines(x=c(yr[1],yr[nyr]),y=c(0.5,0.5),lty="longdash",col="red")

 if(substr(id\_file,1,3)=="Sim") {

 lines(x=c(yr[1],yr[nyr]),y=c(Bmsy.true,Bmsy.true),lty="dashed",col="blue")

 lines(x=yr,y=cpue.raw/(cinfo$true.kq[1]/2),col="blue")}

 if(is.na(BBmsy.ass[1])==F) {lines(x=yr,y=BBmsy.ass,col="blue")}

 if(is.na(BBmsy.ass.ucl[1])==F) {lines(x=yr,y=BBmsy.ass.ucl,lty="dotted",col="blue")}

 if(is.na(BBmsy.ass.lcl[1])==F) {lines(x=yr,y=BBmsy.ass.lcl,lty="dotted",col="blue")}

 # plot B/Bmsy from BSM

 if(is.null(cinfo$B\_Bmsy[cinfo$Stock==stock])==F) {

 points(x=yr[nyr],y=cinfo$B\_Bmsy[cinfo$Stock==stock],col="blue")

 lines(x=c(yr[nyr],yr[nyr]),y=c(cinfo$B\_Bmsy.lcl[cinfo$Stock==stock],

 cinfo$B\_Bmsy.ucl[cinfo$Stock==stock]),col="blue")

 }

 # plot B/Bmsy from BSM if CMSY file

 if(is.na(C.ass[1])==F && is.null(cinfo$r.BSM[cinfo$Stock==stock])==F && substr(id\_file,1,4)=="CMSY") {

 lines(x=yr, y=BBmsy.BSM,col="blue")

 }

 #if(is.na(C.ass[1])==F && is.null(cinfo$r.BSM[cinfo$Stock==stock])==F) {

 # lines(x=yr, y=BBmsy.BSM,col="blue")

 #}

 if (save.plots==TRUE & do.plots==TRUE) {

 jpgfile<-paste(stock,"\_AMSY.jpg",sep="")

 dev.copy(jpeg,jpgfile,

 width = 1024,

 height = 768,

 units = "px",

 pointsize = 18,

 quality = 95,

 res=80,

 antialias="cleartype")

 dev.off()

 }

 if(kobe.plot==T){

 # open window for plot of four panels

 if(grepl("win",tolower(Sys.info()['sysname']))) {windows(7,7)}

 par(mfrow=c(1,1))

 # make margins narrower

 par(mar=c(4.1,4.1,2.1,2.1))

 bbmsy = (cpuet.sel[,nyr]/(0.5\*kqv.est))

 ffmsy = ((apply(cqt.sel[,(nyr-4):(nyr-1)],1,median)/cpuet.sel[,nyr])/(0.5\*rv.est))

 log.bbmsy = log(bbmsy[ffmsy>0]) # Prevents NA warning

 log.ffmsy = log(ffmsy[ffmsy>0]) # Prevents NA warning

 # get mean after all the CMSY subsetting (can't match with biomass sbmsetting)

 mu.kobe = c(median(log.ffmsy),median(log.bbmsy))

 # Get covariance of the 2 vectors

 cov.kobe = cov(cbind(log.ffmsy,log.bbmsy))

 # Generate 10000 new random deviates from a MVN

 log.kobe.mvn = rmvnorm(10000 ,mean = mu.kobe,sigma = cov.kobe)

 kobe.mvn = exp(log.kobe.mvn)

 # Generate 10000 new random deviates from a MVN

 x.F\_Fmsy =exp(log.kobe.mvn[,1])

 y.b\_bmsy =exp(log.kobe.mvn[,2])

 kernelF <- ci2d(y.b\_bmsy,x.F\_Fmsy,nbins=151,factor=2.2,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])))

 max.y <- max(c(2, quantile(x.F\_Fmsy,0.96),na.rm =T))

 max.x <- max(max(2,quantile(y.b\_bmsy,0.999)))

 #Create plot

 plot(1000,1000,type="b", xlim=c(0,max.x), ylim=c(0,max.y),lty=3,xlab="",ylab="", bty="l", cex.main = 2, cex.lab = 1.35, cex.axis = 1.35,xaxs = "i",yaxs="i")

 mtext(expression(B/B[MSY]),side=1, line=2.7, cex=1.4)

 mtext(expression(F/F[MSY]),side=2, line=2.7, cex=1.4)

 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)

 zf2 = c(1,100)

 zb1 = c(1,100)

 zf1 = c(0,1)

 polygon(c(zb1,rev(zb1)),c(0,0,1,1),col="green",border=0)

 polygon(c(zb2,rev(zb2)),c(0,0,1,1),col="yellow",border=0)

 polygon(c(1,100,100,1),c(1,1,100,100),col="orange",border=0)

 polygon(c(0,1,1,0),c(1,1,100,100),col="red",border=0)

 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(Bkt,c(FFmsy,median(x.F\_Fmsy)),pch=16,cex=1)

 lines(c1,c2,lty=3,lwd=0.7)

 lines(c2,c1,lty=3,lwd=0.7)

 lines(Bkt,c(FFmsy,median(x.F\_Fmsy)), lty=1,lwd=1.)

 points(Bkt[1],c(FFmsy,median(x.F\_Fmsy))[1],col=1,pch=22,bg="white",cex=1.5)

 points(Bkt[which(yr==median(yr))],c(FFmsy,median(x.F\_Fmsy))[which(yr==median(yr))],col=1,pch=21,bg="white",cex=1.5)

 points(Bkt[nyr],c(FFmsy,median(x.F\_Fmsy))[nyr],col=1,pch=24,bg="white",cex=1.5)

 # Get Propability

 Pr.green = sum(ifelse(y.b\_bmsy>1 & x.F\_Fmsy<1,1,0))/length(y.b\_bmsy)\*100

 Pr.red = sum(ifelse(y.b\_bmsy<1 & x.F\_Fmsy>1,1,0))/length(y.b\_bmsy)\*100

 Pr.yellow = sum(ifelse(y.b\_bmsy<1 & x.F\_Fmsy<1,1,0))/length(y.b\_bmsy)\*100

 Pr.orange = sum(ifelse(y.b\_bmsy>1 & x.F\_Fmsy>1,1,0))/length(y.b\_bmsy)\*100

 sel.years = c(median(yr))

 legend('topright',

 c(paste(start.yr),paste(median(yr)),paste(end.yr),"50% C.I.","80% C.I.","95% C.I.",paste0(round(c(Pr.red,Pr.yellow,Pr.orange,Pr.green),1),"%")),

 lty=c(1,1,1,rep(-1,8)),pch=c(22,21,24,rep(22,8)),pt.bg=c(rep("white",3),"cornsilk2","grey","cornsilk4","red","yellow","orange","green"),

 col=1,lwd=1.1,cex=1.1,pt.cex=c(rep(1.3,3),rep(1.7,3),rep(2.2,4)),bty="n",y.intersp = 1.)

 } # End Kobe

 if (save.plots==TRUE & kobe.plot==TRUE)

 {

 jpgfile<-paste(stock,"\_KOBE.jpg",sep="")

 dev.copy(jpeg,jpgfile,

 width = 1024\*0.7,

 height = 1024\*0.7,

 units = "px",

 pointsize = 18,

 quality = 95,

 res=80,

 antialias="cleartype")

 dev.off()

 }

 FFmsy.retrospective[[retrosp.step+1]]<-FFmsy #retrospective analysis

 BBmsy.retrospective[[retrosp.step+1]]<-Bkt #retrospective analysis

 years.retrospective[[retrosp.step+1]]<-yr #retrospective analysis

 } # end of do.plots loop

 } #retrospective analysis - end loop

 #retrospective analysis plots

 if (retros.nyears>0 && do.plots==T){

 if(grepl("win",tolower(Sys.info()['sysname']))) {windows(12,7)}

 par(mfrow=c(1,2))

 allyears<-years.retrospective[[1]]

 nyrtotal<-length(allyears)

 plot(x=allyears[1:(nyrtotal-1)],y=FFmsy.retrospective[[1]], main=as.character(stock), ylim=c(0,max(FFmsy.retrospective[[1]],na.rm=T)), lwd=2, xlab="Year", ylab="F/Fmsy", type="l", bty="l", cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.5)

 lines(x=allyears[1:(nyrtotal-2)],y=FFmsy.retrospective[[2]], type = "o", pch=15, col="red")

 lines(x=allyears[1:(nyrtotal-3)],y=FFmsy.retrospective[[3]], type = "o", pch=16, col="green")

 lines(x=allyears[1:(nyrtotal-4)],y=FFmsy.retrospective[[4]], type = "o", pch=17, col="blue")

 legend("bottomleft", legend = c("Reference",allyears[nyrtotal-2],allyears[nyrtotal-3],allyears[nyrtotal-4]),

 col=c("black","red", "green", "blue"), lty=1, pch=c(-1,15,16,17))

 plot(x=allyears[1:(nyrtotal)],y=BBmsy.retrospective[[1]],main=as.character(stock), ylim=c(0,max(BBmsy.retrospective[[1]],na.rm=T)), lwd=2, xlab="Year", ylab="B/Bmsy", type="l", bty="l",cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.5)

 lines(x=allyears[1:(nyrtotal-1)],y=BBmsy.retrospective[[2]], type = "o", pch=15, col="red")

 lines(x=allyears[1:(nyrtotal-2)],y=BBmsy.retrospective[[3]], type = "o", pch=16, col="green")

 lines(x=allyears[1:(nyrtotal-3)],y=BBmsy.retrospective[[4]], type = "o", pch=17, col="blue")

 legend("bottomleft", legend = c("Reference",allyears[nyrtotal-1],allyears[nyrtotal-2],allyears[nyrtotal-3]),

 col=c("black","red", "green", "blue"), lty=1, pch=c(-1,15,16,17))

 } #retrospective analysis plots - end

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

 # write results to file

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

 if(write.output==TRUE && substr(id\_file,1,3)=="Sim") {

 output = data.frame(as.character(stock),

 r.true,rv.est,rv.lcl,rv.ucl,

 kq.true,kqv.est,kqv.lcl,kqv.ucl,

 cinfo$true.MSYq[cinfo$Stock==stock],

 MSYq.est,MSYq.lcl,MSYq.ucl,

 FFmsy.end.true,

 FFmsy.end,FFmsy.end.lcl,FFmsy.end.ucl,

 BBmsy.end.true,

 BBmsy.end,BBmsy.end.lcl,BBmsy.end.ucl )

 write.table(output, file=outfile, append = T, sep = ",",

 dec = ".", row.names = FALSE, col.names = FALSE)

 }

 if(write.output==TRUE && is.null(cinfo$MSY.BSM[cinfo$Stock==stock])==F) { # assuming all BSM fields are available

 output = data.frame(as.character(stock),

 cinfo$r.BSM[cinfo$Stock==stock],cinfo$r.BSM.lcl[cinfo$Stock==stock],cinfo$r.BSM.ucl[cinfo$Stock==stock],

 rv.est,rv.lcl,rv.ucl,

 cinfo$k.BSM[cinfo$Stock==stock]\*1000,cinfo$k.BSM.lcl[cinfo$Stock==stock]\*1000,cinfo$k.BSM.ucl[cinfo$Stock==stock]\*1000,

 kqv.est,kqv.lcl,kqv.ucl,

 cinfo$B\_Bmsy[cinfo$Stock==stock],cinfo$B\_Bmsy.lcl[cinfo$Stock==stock],cinfo$B\_Bmsy.ucl[cinfo$Stock==stock],

 BBmsy.end,BBmsy.end.lcl,BBmsy.end.ucl,

 cinfo$F\_Fmsy[cinfo$Stock==stock],cinfo$F\_Fmsy.lcl[cinfo$Stock==stock],cinfo$F\_Fmsy.ucl[cinfo$Stock==stock],

 FFmsy[nyr-1],FFmsy.lcl[nyr-1],FFmsy.ucl[nyr-1])

 write.table(output, file=outfile, append = T, sep = ",",dec = ".", row.names = FALSE, col.names = FALSE)

 } # end of BSM option to write results to file

 if(write.output==T && substr(id\_file,1,3)!="Sim" && is.null(cinfo$MSY.BSM)==T){ # regular assessment

 output = data.frame(as.character(stock),rv.est/2,rv.lcl/2,rv.ucl/2,

 kqv.est,kqv.lcl,kqv.ucl,

 FFmsy[nyr-1],FFmsy.lcl[nyr-1],FFmsy.ucl[nyr-1],

 cpue[nyr]/kqv.est,cpue[nyr]/kqv.ucl,cpue[nyr]/kqv.lcl)

 write.table(output, file=outfile, append = T, sep = ",",

 dec = ".", row.names = FALSE, col.names = FALSE)

 } # end of regular output

} # end loop on stocks

##CMSY and BSM-code

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

## CMSY and BSM analysis

## Written by Rainer Froese, Gianpaolo Coro and Henning Winker

## Version of November 2016

## Note that time series excluding 2004 - 2010 will give an error in dataframe; set write.output <- F to avoid that error

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

library(R2jags) # Interface with JAGS

library(coda)

library("parallel")

library("foreach")

library("doParallel")

library("gplots")

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

# Some general settings

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

# set.seed(999) # use for comparing results between runs

rm(list=ls(all=TRUE)) # clear previous variables etc

options(digits=3) # displays all numbers with three significant digits as default

graphics.off() # close graphics windows from previous sessions

FullSchaefer <- F # initialize variable; automatically set to TRUE if enough abundance data are available

n.chains <- ifelse(detectCores() > 2,3,2) # set 3 chains in JAGS if more than 2 cores are available

ncores\_for\_computation=detectCores() # cores to be used for parallel processing of CMSY

cl <- makeCluster(ncores\_for\_computation)

registerDoParallel(cl, cores = ncores\_for\_computation)

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

# Required settings, File names

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

catch\_file <- "O\_Stocks\_Catch\_15\_Med.csv" # name of file containing "stock", "yr", "ct", and optional "bt"

id\_file <- "O\_Stocks\_ID\_18\_Med.csv" # name of file containing stock-specific info and settings for the analysis

outfile <- paste("Out\_",format(Sys.Date(),format="%B%d%Y\_"),id\_file,sep="") # default name for output file

outfile.txt <- paste(outfile,".txt", sep="")

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

# Select stock to be analyzed

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

stocks <-"ALB"

# If the input files contain more than one stock, specify below the stock to be analyzed

# If the line below is commented out (#), all stocks in the input file will be analyzed

stocks <- "ALB" # c("SEPIOFF\_CY","MICRPOU\_IS","EPINGUA\_IS","CHAMGAL\_SA","CORYHIP\_SA","ILLECOI\_SA")

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

# General settings for the analysis

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

dataUncert <- 0.1 # set observation error as uncertainty in catch - default is SD=0.1

sigmaR <- 0.1 # overall process error for CMSY; SD=0.1 is the default

n <- 10000 # initial number of r-k pairs

n.new <- n # initialize n.new

ni <- 3 # iterations for r-k-startbiomass combinations, to test different variability patterns; no improvement seen above 3

nab <- 2 # default=5; minimum number of years with abundance data to run BSM

mgraphs <- T # set to TRUE to produce additional graphs for management

save.plots <- T # set to TRUE to save graphs to JPEG files

close.plots <- F # set to TRUE to close on-screen plots after they are saved, to avoid "too many open devices" error in batch-processing

write.output <- T # set to TRUE if table with results in output file is wanted; expects years 2004-2010 to be available

force.cmsy <- F # set to TRUE if CMSY results are to be preferred over BSM results

select.yr <- NA # option to display F, B, F/Fmsy and B/Bmsy for a certain year; default NA

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

# FUNCTIONS

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

# Monte Carlo filtering with Schaefer Function

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

SchaeferParallelSearch<-function(ni, nyr,sigR,duncert,ct,int.yr,intbio, startbt, ki,i, ri,int.yr.i, nstartbt, yr, end.yr, endbio, npoints, pt){

 ptm<-proc.time()

 # create vectors for viable r, k and bt

 inmemorytable <- vector()

 # parallelised for the points in the r-k space

 inmemorytable <- foreach (i = 1 : npoints, .combine='rbind', .packages='foreach', .inorder=TRUE) %dopar%{

 nsbt = length(startbt)

 VP <- FALSE

 for(nj in 1:nsbt) {

 # create empty vector for annual biomasses

 bt <- vector()

 j<-startbt[nj]

 # set initial biomass, including 0.1 process error to stay within bounds

 bt[1]=j\*ki[i]\*exp(rnorm(1,0, 0.1\*sigR)) ## set biomass in first year

 # repeat test of r-k-startbt combination to allow for different random error

 for(re in 1:ni) {

 #loop through years in catch time series

 for (t in 1:nyr) { # for all years in the time series

 xt=rnorm(1,0, sigR) # set new process error for every year

 zlog.sd = sqrt(log(1+(duncert)^2))

 zt=rlnorm(1,meanlog = 0, sdlog = zlog.sd) # model the catch error as a log normal distribution.

 # calculate biomass as function of previous year's biomass plus surplus production minus catch

 bt[t+1] <- ifelse(bt[t]/ki[i] >= 0.25,

 bt[t]+ri[i]\*bt[t]\*(1-bt[t]/ki[i])\*exp(xt)-ct[t]\*zt,

 bt[t]+(4\*bt[t]/ki[i])\*ri[i]\*bt[t]\*(1-bt[t]/ki[i])\*exp(xt)-ct[t]\*zt) # assuming reduced r at B/k < 0.25

 # if biomass < 0.01 k, discard r-k-startbt combination

 if(bt[t+1] < 0.01\*ki[i]) {

 break

 } # stop looping through years, go to next upper level

 # intermediate year check

 if ((t+1)==int.yr.i && (bt[t+1]>(intbio[2]\*ki[i]) || bt[t+1]<(intbio[1]\*ki[i]))) {

 break

 }

 } # end of loop of years

 # if loop was broken or last biomass falls outside of expected ranges

 # do not store results, go directly to next startbt

 if(t < nyr || bt[yr==end.yr] > (endbio[2]\*ki[i]) || bt[yr==end.yr] < (endbio[1]\*ki[i]) ) {

 next

 } else {

 #each vector will be finally appended to the others found by the threads - this is done by the .combine='rbind' option

 inmemorytablerow<-c(i,j,ri[i],ki[i],bt[1:(nyr+1)]/ki[i])

 if (length(inmemorytablerow)==(4+nyr+1)){

 if (VP==FALSE)

 {

 inmemorytable<-inmemorytablerow

 }

 else

 {

 inmemorytable<-rbind(inmemorytable,inmemorytablerow)

 }

 VP<-TRUE

 }

 }

 } # end of repetition for random error

 } # end of j-loop of initial biomasses

 # instruction necessary to make the foreach loop see the variable:

 if (length(inmemorytable)==0)

 {inmemorytable<-vector(length=4+nyr+1)\*NA}

 else

 {inmemorytable}

 }#end loop on points

 #create the output matrix

 mdat <- matrix(data=NA, nrow = npoints\*nstartbt, ncol = 2+nyr+1)

 npointsmem = dim(inmemorytable)[1]

 npointscols = dim(inmemorytable)[2]

 #reconstruction of the processing matrix after the parallel search

 if (npointsmem>0 && npointscols>0){

 for (idxr in 1:npointsmem){

 i = inmemorytable[idxr,1]

 if (!is.na(i)){

 j = inmemorytable[idxr,2]

 mdatindex<-((i-1)\*nstartbt)+which(startbt==j)

 mdat[mdatindex,1] <- inmemorytable[idxr,3]

 mdat[mdatindex,2] <- inmemorytable[idxr,4]

 mdat[mdatindex,3:(2+nyr+1)] <- inmemorytable[idxr,5:(4+nyr+1)]

 if(pt==T) points(x=ri[i], y=ki[i], pch=".", cex=4, col="gray")

 }

 }

 }

 ptm<-proc.time()-ptm

 mdat <- na.omit(mdat)

 return(mdat)

}

SchaeferMC <- function(ri, ki, startbio, int.yr, intbio, endbio, sigR, pt, duncert, startbins, ni) {

 # create vector for initial biomasses

 startbt <- seq(from =startbio[1], to=startbio[2], by = (startbio[2]-startbio[1])/startbins)

 nstartbt <- length(startbt)

 npoints <- length(ri)

 # get index of intermediate year

 int.yr.i <- which(yr==int.yr)

 #loop through r-k pairs with parallel search

 mdat<-SchaeferParallelSearch(ni, nyr,sigR,duncert,ct,int.yr,intbio, startbt, ki, i, ri, int.yr.i, nstartbt, yr, end.yr, endbio, npoints,pt)

 cat("\n")

 return(list(mdat))

} # end of SchaeferMC function

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

# Function for moving average

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

ma <- function(x){

 x.1 <- filter(x,rep(1/3,3),sides=1)

 x.1[1] <- x[1]

 x.1[2] <- (x[1]+x[2])/2

 return(x.1)

}

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

# END OF FUNCTIONS

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

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

# Start output to screen

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

cat("-------------------------------------------\n")

cat("CMSY Analysis,", date(),"\n")

cat("-------------------------------------------\n")

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

# Read data and assign to vectors

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

# create headers for data table file

if(write.output==T){

 outheaders = data.frame("Group","Region", "Subregion","Name","SciName","Stock","start.yr","end.yr","btype",

 "MaxCatch","LastCatch","MSY\_BSM","lcl","ucl","r\_BSM","lcl","ucl",

 "k\_BSM","lcl","ucl","q\_BSM","lcl","ucl","rel\_B\_BSM","lcl","ucl","rel\_F\_BSM",

 "r\_CMSY","lcl","ucl","k\_CMSY","lcl","ucl","MSY\_CMSY","lcl","ucl",

 "rel\_B\_CMSY","2.5th","97.5th","rel\_F\_CMSY",

 "F\_msy","lcl","ucl","curF\_msy","lcl","ucl",

 "MSY","lcl","ucl","Bmsy","lcl","ucl",

 "B","lcl","ucl","B\_Bmsy","lcl","ucl",

 "F","lcl","ucl","F\_Fmsy","lcl","ucl",

 "sel\_B","sel\_B\_Bmsy","sel\_F","sel\_F\_Fmsy",

 "c00","c01","c02","c03","c04","c05","c06","c07","c08","c09","c10","c11","c12","c13","c14","c15",

 "F.Fmsy00","F.Fmsy01","F.Fmsy02","F.Fmsy03","F.Fmsy04","F.Fmsy05","F.Fmsy06","F.Fmsy07","F.Fmsy08","F.Fmsy09","F.Fmsy10","F.Fmsy11","F.Fmsy12","F.Fmsy13","F.Fmsy14","F.Fmsy15",

 "B00","B01","B02","B03","B04","B05","B06","B07","B08","B09","B10","B11","B12","B13","B14","B15")

 write.table(outheaders,file=outfile, append = T, sep=",",row.names=F,col.names=F)

}

cat("Parallel processing will use",ncores\_for\_computation,"cores\n")

# Read data

cdat <- read.csv(catch\_file, header=T, dec=".", stringsAsFactors = FALSE)

cinfo <- read.csv(id\_file, header=T, dec=".", stringsAsFactors = FALSE)

cat("Files", catch\_file, ",", id\_file, "read successfully","\n")

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

# Analyze stock(s)

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

if(is.na(stocks[1])==TRUE){

 stocks <- as.character(cinfo$Stock) # Analyze stocks in sequence of ID file

 # stocks <- sort(as.character(cinfo$Stock)) # Analyze stocks in alphabetic order

 # stocks <- as.character(cinfo$Stock[cinfo$Subregion=="Sardinia"]) # Analyze stocks in Region

}

# analyze one stock after the other

for(stock in stocks) {

 cat("Processing",stock,",", as.character(cinfo$ScientificName[cinfo$Stock==stock]),"\n")

 # assign data from cinfo to vectors

 res <- as.character(cinfo$Resilience[cinfo$Stock==stock])

 start.yr <- as.numeric(cinfo$StartYear[cinfo$Stock==stock])

 end.yr <- as.numeric(cinfo$EndYear[cinfo$Stock==stock])

 r.low <- as.numeric(cinfo$r.low[cinfo$Stock==stock])

 r.hi <- as.numeric(cinfo$r.hi[cinfo$Stock==stock])

 user.log.r <- ifelse(is.na(r.low)==F & is.na(r.hi)==F,TRUE,FALSE)

 stb.low <- as.numeric(cinfo$stb.low[cinfo$Stock==stock])

 stb.hi <- as.numeric(cinfo$stb.hi[cinfo$Stock==stock])

 int.yr <- as.numeric(cinfo$int.yr[cinfo$Stock==stock])

 intb.low <- as.numeric(cinfo$intb.low[cinfo$Stock==stock])

 intb.hi <- as.numeric(cinfo$intb.hi[cinfo$Stock==stock])

 endb.low <- as.numeric(cinfo$endb.low[cinfo$Stock==stock])

 endb.hi <- as.numeric(cinfo$endb.hi[cinfo$Stock==stock])

 btype <- as.character(cinfo$btype[cinfo$Stock==stock])

 force.cmsy <- ifelse(force.cmsy==T,T,cinfo$force.cmsy[cinfo$Stock==stock])

 comment <- as.character(cinfo$Comment[cinfo$Stock==stock])

 # set global defaults for uncertainty

 duncert <- dataUncert

 sigR <- sigmaR

 # check for common errors

 if (length(btype)==0){

 cat("ERROR: Could not find the stock in the ID input file - check that the stock names match in ID and Catch files and that commas are used (not semi-colon)")

 return (NA) }

 if(start.yr < cdat$yr[cdat$Stock==stock][1]){

 cat("ERROR: start year in ID file before first year in catch file\n")

 return (NA)}

 # extract data on stock

 yr <- as.numeric(cdat$yr[cdat$Stock==stock & cdat$yr >= start.yr & cdat$yr <= end.yr])

 if (length(yr)==0){

 cat("ERROR: Could not find the stock in the Catch input files - Please check that the code is written correctly")

 return (NA)

 }

 ct.raw <- as.numeric(cdat$ct[cdat$Stock==stock & cdat$yr >= start.yr & cdat$yr <= end.yr])/1000 ## assumes that catch is given in tonnes, transforms to '000 tonnes

 if(btype=="biomass" | btype=="CPUE" ) {

 bt <- as.numeric(cdat$bt[cdat$Stock==stock & cdat$yr >= start.yr & cdat$yr <= end.yr])/1000 ## assumes that biomass is in tonnes, transforms to '000 tonnes

 } else {bt <- NA}

 if(is.na(mean(ct.raw))){

 cat("ERROR: Missing value in Catch data; fill or interpolate\n")

 }

 nyr <- length(yr) # number of years in the time series

 # change catch to 3 years moving average where value is average of past 3 years

 ct <- ma(ct.raw)

 # initialize vectors for viable r, k, bt, and all in a matrix

 mdat.all <- matrix(data=vector(),ncol=2+nyr+1)

 # initialize other vectors anew for each stock

 current.attempts <- NA

 # use start.yr if larger than select year

 if(is.na(select.yr)==F) {

 sel.yr <- ifelse(start.yr > select.yr,start.yr,select.yr)

 } else sel.yr <- NA

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

 # Determine initial ranges for parameters and biomass

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

 # initial range of r from input file

 if(is.na(r.low)==F & is.na(r.hi)==F) {

 start.r <- c(r.low,r.hi)

 } else {

 # initial range of r based on resilience

 if(res == "High") {

 start.r <- c(0.6,1.5)} else if(res == "Medium") {

 start.r <- c(0.2,0.8)} else if(res == "Low") {

 start.r <- c(0.05,0.5)} else { # i.e. res== "Very low"

 start.r <- c(0.015,0.1)}

 }

 # get index of years with lowest and highest catch between start+3 and end-3 years

 min.yr.i <- which.min(ct[4:(length(ct)-3)])+3

 max.yr.i <- which.max(ct[4:(length(ct)-3)])+3

 min.ct <- ct[min.yr.i]

 max.ct <- ct[max.yr.i]

 # use initial biomass range from input file if stated

 if(is.na(stb.low)==F & is.na(stb.hi)==F) {

 startbio <- c(stb.low,stb.hi)

 } else {

 # if start year < 1960 assume high biomass

 if(start.yr < 1960) {startbio <- c(0.5,0.9)} else {

 # else use medium prior biomass range

 startbio <- c(0.2,0.6)} }

 # use year and biomass range for intermediate biomass from input file

 if(is.na(intb.low)==F & is.na(intb.hi)==F) {

 int.yr <- int.yr

 intbio <- c(intb.low,intb.hi)

 # if contrast in catch is low, use initial range again in mid-year

 } else if(min(ct)/max(ct) > 0.6) {

 int.yr <- as.integer(mean(c(start.yr, end.yr)))

 intbio <- startbio

 # else if year of minimum catch is after max catch then use min catch

 } else if(min.yr.i > max.yr.i) {

 int.yr <- yr[min.yr.i-1]

 if(startbio[1]>=0.5 & (int.yr-start.yr) < (end.yr-int.yr) &

 (min.ct/max.ct) > 0.3) intbio <- c(0.2,0.6) else intbio <- c(0.01,0.4)

 # else use max catch

 } else {

 # assume that biomass range in year before maximum catch was high or medium

 int.yr <- yr[max.yr.i-1]

 intbio <- if((startbio[1]>=0.5 & (int.yr-start.yr) < (end.yr-int.yr))| # if initial biomass is high, assume same for intermediate

 # ((min.ct/max.ct < 0.3 & (max.yr.i - min.yr.i) < 25))) c(0.5,0.9) else c(0.2,0.6) }

 (((max.ct-min.ct)/max.ct)/(max.yr.i-min.yr.i) > 0.04)) c(0.5,0.9) else c(0.2,0.6) } # if incease is steep, assume high, else medium

 # end of intbio setting

 # final biomass range from input file

 if(is.na(endb.low)==F & is.na(endb.hi)==F) {

 endbio <- c(endb.low,endb.hi)

 } else {

 # else use mean final catch/max catch to estimate final biomass

 rawct.ratio=ct.raw[nyr]/max(ct)

 endbio <- if(ct[nyr]/max(ct) > 0.8) {c(0.4,0.8)} else if(rawct.ratio < 0.5) {c(0.01,0.4)} else {c(0.2,0.6)}

 # if default endbio is low (0.01-0.4), check whether the upper bound should be lower than 0.4 for depleted stocks

 if(endbio[2]==0.4){

 if(rawct.ratio< 0.05) {endbio[2] <- 0.1} else

 if(rawct.ratio< 0.15) {endbio[2] <- 0.2} else

 if(rawct.ratio< 0.35) {endbio[2] <- 0.3} else {endbio[2] <- 0.4}

 }

 } # end of final biomass setting

 # initial prior range of k values, assuming min k will be larger than max catch / prior for r

 if(mean(endbio) <= 0.5) {

 start.k <- c(max(ct)/start.r[2],4\*max(ct)/start.r[1])} else {

 start.k <- c(2\*max(ct)/start.r[2],12\*max(ct)/start.r[1])}

 # start.k <- c(start.k[1],3000)

 cat("startbio=",startbio,ifelse(is.na(stb.low)==T,"default","expert"),

 ", intbio=",int.yr,intbio,ifelse(is.na(intb.low)==T,"default","expert"),

 ", endbio=",endbio,ifelse(is.na(endb.low)==T,"default","expert"),"\n")

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

 # Uniform sampling of the r-k space

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

 # get random set of r and k from log space distribution

 ri1 = exp(runif(n, log(start.r[1]), log(start.r[2])))

 ki1 = exp(runif(n, log(start.k[1]), log(start.k[2])))

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

 # Plot data and progress

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

 # check for operating system, open separate window for graphs if Windows

 if(grepl("win",tolower(Sys.info()['sysname']))) {windows(14,9)}

 par(mfrow=c(2,3))

 # plot catch

 plot(x=yr, y=ct.raw,

 ylim=c(0,max(ifelse(substr(id\_file,1,3)=="Sim",

 1.1\*true.MSY,0),1.2\*max(ct.raw))),

 type ="l", bty="l", main=paste("A: ",stock,"catch"), xlab="Year", ylab="Catch", lwd=2)

 lines(x=yr,y=ct,col="blue", lwd=1)

 points(x=yr[max.yr.i], y=max.ct, col="red", lwd=2)

 points(x=yr[min.yr.i], y=min.ct, col="red", lwd=2)

 # plot r-k graph

 plot(x=ri1, y=ki1, xlim = start.r, ylim = start.k, log="xy", xlab="r", ylab="k",

 main="B: Finding viable r-k", pch=".", cex=3, bty="l", col="gray95")

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

 # 1 - Call CMSY-SchaeferMC function to preliminary explore the r-k space

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

 cat("First Monte Carlo filtering of r-k space with ",n," points...\n")

 MCA <- SchaeferMC(ri=ri1, ki=ki1, startbio=startbio, int.yr=int.yr, intbio=intbio, endbio=endbio, sigR=sigR,

 pt=T, duncert=dataUncert, startbins=10, ni=ni)

 mdat.all <- rbind(mdat.all,MCA[[1]])

 rv.all <- mdat.all[,1]

 kv.all <- mdat.all[,2]

 btv.all <- mdat.all[,3:(2+nyr+1)]

 # count viable trajectories and r-k pairs

 n.viable.b <- length(mdat.all[,1])

 n.viable.pt <- length(unique(mdat.all[,1]))

 cat("Found ",n.viable.b," viable trajectories for", n.viable.pt," r-k pairs\n")

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

 # 2 - if the lower bound of k is too high, reduce it by half and rerun

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

 if(length(kv.all[kv.all < 1.1\*start.k[1] & rv.all < mean(start.r)]) > 10) {

 cat("Reducing lower bound of k, resampling area with",n,"additional points...\n")

 start.k <- c(0.5\*start.k[1],start.k[2])

 ri1 = exp(runif(n, log(start.r[1]), log(start.r[2])))

 ki1 = exp(runif(n, log(start.k[1]), log(start.k[2])))

 MCA <- SchaeferMC(ri=ri1, ki=ki1, startbio=startbio, int.yr=int.yr, intbio=intbio, endbio=endbio, sigR=sigR,

 pt=T, duncert=dataUncert, startbins=10, ni=ni)

 mdat.all <- rbind(mdat.all,MCA[[1]])

 rv.all <- mdat.all[,1]

 kv.all <- mdat.all[,2]

 btv.all <- mdat.all[,3:(2+nyr+1)]

 n.viable.b <- length(mdat.all[,1])

 n.viable.pt <- length(unique(mdat.all[,1]))

 cat("Found altogether",n.viable.b," viable trajectories for", n.viable.pt," r-k pairs\n")

 }

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

 # 3 - if few points were found then resample and shrink the log k space

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

 if (n.viable.b <= 1000){

 log.start.k.new <- log(start.k)

 max.attempts <- 3

 current.attempts <- 1

 startbins <- 10

 while (n.viable.b <= 1000 && current.attempts <= max.attempts){

 if(n.viable.pt > 0) {

 log.start.k.new[1] <- mean(c(log(start.k[1]), min(log(kv.all))))

 log.start.k.new[2] <- mean(c(log.start.k.new[2], max(log(kv.all)))) }

 n.new <- n\*current.attempts #add more points

 ri1 = exp(runif(n.new, log(start.r[1]), log(start.r[2])))

 ki1 = exp(runif(n.new, log.start.k.new[1], log.start.k.new[2]))

 cat("Shrinking k space: repeating Monte Carlo in the interval [",exp(log.start.k.new[1]),",",exp(log.start.k.new[2]),"]\n")

 cat("Attempt ",current.attempts," of ",max.attempts," with ",n.new," additional points...","\n")

 if(current.attempts==2 & n.viable.b < 50){

 duncert <- 2\*dataUncert

 sigR <- 2\*sigmaR

 startbins <- 20

 cat("Doubling startbins, catch and process error, and number of variability patterns \n")

 }

 MCA <- SchaeferMC(ri=ri1, ki=ki1, startbio=startbio, int.yr=int.yr, intbio=intbio, endbio=endbio, sigR=sigR,

 pt=T, duncert=duncert, startbins=startbins, ni=2\*ni)

 mdat.all <- rbind(mdat.all,MCA[[1]])

 rv.all <- mdat.all[,1]

 kv.all <- mdat.all[,2]

 btv.all <- mdat.all[,3:(2+nyr+1)]

 n.viable.b <- length(mdat.all[,1])

 n.viable.pt <- length(unique(mdat.all[,1]))

 cat("Found altogether",n.viable.b," viable trajectories for", n.viable.pt," r-k pairs\n")

 current.attempts=current.attempts+1 #increment the number of attempts

 }

 if(n.viable.b < 5) {

 cat("Only",n.viable.pt,"viable r-k pairs found, check data and settings \n")

 next

 }

 }

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

 # 4 - if tip of viable r-k pairs is 'thin', do extra sampling there

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

 if(length(rv.all[rv.all > 0.9\*start.r[2]]) < 5) {

 l.sample.r <- quantile(rv.all,0.6)

 add.points <- ifelse(is.na(current.attempts)==T,n,ifelse(current.attempts==2,2\*n,ifelse(length(rv.all)>500,3\*n,6\*n)))

 cat("Final sampling in the tip area above r =",l.sample.r,"with",add.points,"additional points...\n")

 log.start.k.new <- c(log(0.8\*min(kv.all)),log(max(kv.all[rv.all > l.sample.r])))

 ri1 = exp(runif(add.points, log(l.sample.r), log(start.r[2])))

 ki1 = exp(runif(add.points, log.start.k.new[1], log.start.k.new[2]))

 MCA <- SchaeferMC(ri=ri1, ki=ki1, startbio=startbio, int.yr=int.yr, intbio=intbio, endbio=endbio, sigR=sigR,

 pt=T, duncert=duncert, startbins=10, ni=ni)

 mdat.all <- rbind(mdat.all,MCA[[1]])

 rv.all <- mdat.all[,1]

 kv.all <- mdat.all[,2]

 btv.all <- mdat.all[,3:(2+nyr+1)]

 n.viable.b <- length(mdat.all[,1])

 n.viable.pt <- length(unique(mdat.all[,1]))

 cat("Found altogether",n.viable.b," viable trajectories for", n.viable.pt," r-k pairs\n")

 }

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

 # Bayesian analysis of catch & biomass (or CPUE) with Schaefer model

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

 FullSchaefer <- F

 if(btype != "None" & length(bt[is.na(bt)==F])>=nab) {

 FullSchaefer <- T

 # set inits for r-k in lower right corner of log r-k space

 init.r <- start.r[1]+0.8\*(start.r[2]-start.r[1])

 init.k <- start.k[1]+0.1\*(start.k[2]-start.k[1])

 # vector with no penalty (=0) if predicted biomass is within viable range, else a penalty of 10 is set

 pen.bk = pen.F = rep(0,length(ct))

 #><>><>

 # Add biomass priors

 b.yrs = c(1,length(start.yr:int.yr),length(start.yr:end.yr))

 b.prior = rbind(matrix(c(startbio[1],startbio[2],intbio[1],intbio[2],endbio[1],endbio[2]),2,3),rep(0,3)) # last row includes the 0 pen

 #><>><>

 cat("Running MCMC analysis....\n")

 if(btype == "biomass" ) {

 # Data to be passed on to JAGS

 jags.data <- c('ct','bt','nyr', 'start.r','startbio','start.k',

 'init.r','init.k', 'pen.bk','pen.F','b.yrs','b.prior')

 # Parameters to be returned by JAGS

 jags.save.params <- c('r','k','P') #

 # JAGS model

 Model = "model{

 # to avoid crash due to 0 values

 eps<- 0.01

 penm[1] <- 0 # no penalty for first biomass

 Pmean[1] <- log(alpha)

 P[1] ~ dlnorm(Pmean[1],itau2)

 for (t in 2:nyr) {

 Pmean[t] <- ifelse(P[t-1] > 0.25,

 log(max(P[t-1] + r\*P[t-1]\*(1-P[t-1]) - ct[t-1]/k,eps)), # Process equation

 log(max(P[t-1] + 4\*P[t-1]\*r\*P[t-1]\*(1-P[t-1]) - ct[t-1]/k,eps))) # assuming reduced r at B/k < 0.25

 P[t] ~ dlnorm(Pmean[t],itau2) # Introduce process error

 penm[t] <- ifelse(P[t]<(eps+0.001),log(k\*P[t])-log(k\*(eps+0.001)),ifelse(P[t]>1,log(k\*P[t])-log(k\*(0.99)),0)) # penalty if Pmean is outside viable biomass

 }

 # ><> Biomass priors/penalties are enforced as follows

 for (i in 1:3) {

 penb[i] <- ifelse(P[b.yrs[i]]<b.prior[1,i],log(k\*P[b.yrs[i]])-log(k\*b.prior[1,i]),ifelse(P[b.yrs[i]]>b.prior[2,i],log(k\*P[b.yrs[i]])-log(k\*b.prior[2,i]),0))

 b.prior[3,i] ~ dnorm(penb[i],100)

 }

 for (t in 1:nyr){

 Fpen[t] <- ifelse(ct[t]>(0.9\*k\*P[t]),ct[t]-(0.9\*k\*P[t]),0) #><> Penalty term on F > 1, i.e. ct>B

 pen.F[t] ~ dnorm(Fpen[t],1000)

 pen.bk[t] ~ dnorm(penm[t],10000)

 Bm[t] <- log(P[t]\*k);

 bt[t] ~ dlnorm(Bm[t],isigma2);

 }

 # priors

 # search in the alpha space from the center of the range. Allow high variability

 log.alpha <- log((startbio[1]+startbio[2])/2)

 sd.log.alpha <- (log.alpha-log(startbio[1]))/5

 tau.log.alpha <- pow(sd.log.alpha,-2)

 alpha ~ dlnorm(log.alpha,tau.log.alpha)

 # search in the k space from 20% of the range

 log.km <- log(start.k[1]+0.2\*(start.k[2]-start.k[1]))

 sd.log.k <- (log.km-log(start.k[1]))/4

 tau.log.k <- pow(sd.log.k,-2)

 k ~ dlnorm(log.km,tau.log.k)

 # define process (tau) and observation (sigma) variances as inversegamma priors

 itau2 ~ dgamma(2,0.01)

 tau2 <- 1/itau2

 tau <- pow(tau2,0.5)

 isigma2 ~ dgamma(2,0.01)

 sigma2 <- 1/isigma2

 sigma <- pow(sigma2,0.5)

 log.rm <- mean(log(start.r))

 sigma.log.r <- abs(log.rm - log(start.r[1]))/2

 tau.log.r <- pow(sigma.log.r,-2)

 r ~ dlnorm(log.rm,tau.log.r)

 } " # end of JAGS model for btype=="biomass"

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

 # Schaefer model for Catch & CPUE

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

 } else {

 # get prior for q from stable catch/biomass period, min 5 years; get range of years from input file

 q.start <- cinfo$q.start[cinfo$Stock==stock]

 q.end <- cinfo$q.end[cinfo$Stock==stock]

 if(is.na(q.start)==F & is.na(q.end)==F) {

 mean.last.ct <-mean(ct[yr >= q.start & yr <= q.end], na.rm=T) # get mean catch of indicated years

 mean.last.cpue <-mean(bt[yr >= q.start & yr <= q.end], na.rm=T) # get mean of CPUE of indicated years

 } else {

 # get prior range for q from mean catch and mean CPUE in recent years

 lyr <- ifelse(mean(start.r)>=0.5,5,10) # determine number of last years to use, 5 for normal and 10 for slow growing fish

 mean.last.ct <-mean(ct[(nyr-lyr):nyr],na.rm=T) # get mean catch of last years

 mean.last.cpue <-mean(bt[(nyr-lyr):nyr],na.rm=T) # get mean of CPUE of last years

 }

 gm.start.r <- exp(mean(log(start.r))) # get geometric mean of prior r range

 if(mean(endbio) >= 0.5) { # if biomass is high

 q.1 <- mean.last.cpue\*0.25\*gm.start.r/mean.last.ct

 q.2 <- mean.last.cpue\*0.5\*start.r[2]/mean.last.ct

 } else {

 q.1 <- mean.last.cpue\*0.5\*gm.start.r/mean.last.ct

 q.2 <- mean.last.cpue\*start.r[2]/mean.last.ct

 }

 q.prior <- c(q.1,q.2)

 init.q <- mean(q.prior)

 # Data to be passed on to JAGS

 jags.data <- c('ct','bt','nyr', 'start.r', 'start.k', 'startbio', 'q.prior',

 'init.q','init.r','init.k','pen.bk','pen.F','b.yrs','b.prior')

 # Parameters to be returned by JAGS

 jags.save.params <- c('r','k','q', 'P')

 # JAGS model

 Model = "model{

 # to reduce chance of non-convergence, Pmean[t] values are forced >= eps

 eps<-0.01

 penm[1] <- 0 # no penalty for first biomass

 Pmean[1] <- log(alpha)

 P[1] ~ dlnorm(Pmean[1],itau2)

 for (t in 2:nyr) {

 Pmean[t] <- ifelse(P[t-1] > 0.25,

 log(max(P[t-1] + r\*P[t-1]\*(1-P[t-1]) - ct[t-1]/k,eps)), # Process equation

 log(max(P[t-1] + 4\*P[t-1]\*r\*P[t-1]\*(1-P[t-1]) - ct[t-1]/k,eps))) # assuming reduced r at B/k < 0.25

 P[t] ~ dlnorm(Pmean[t],itau2) # Introduce process error

 penm[t] <- ifelse(P[t]<(eps+0.001),log(q\*k\*P[t])-log(q\*k\*(eps+0.001)),ifelse(P[t]>1,log(q\*k\*P[t])-log(q\*k\*(0.99)),0)) # penalty if Pmean is outside viable biomass

 }

 # ><> Biomass priors/penalties are enforced as follows

 for (i in 1:3) {

 penb[i] <- ifelse(P[b.yrs[i]]<b.prior[1,i],log(q\*k\*P[b.yrs[i]])-log(q\*k\*b.prior[1,i]),ifelse(P[b.yrs[i]]>b.prior[2,i],log(q\*k\*P[b.yrs[i]])-log(q\*k\*b.prior[2,i]),0))

 b.prior[3,i] ~ dnorm(penb[i],100)

 }

 for (t in 1:nyr){

 Fpen[t] <- ifelse(ct[t]>(0.9\*k\*P[t]),ct[t]-(0.9\*k\*P[t]),0) #><> Penalty term on F > 1, i.e. ct>B

 pen.F[t] ~ dnorm(Fpen[t],1000)

 pen.bk[t] ~ dnorm(penm[t],10000)

 cpuem[t] <- log(q\*P[t]\*k);

 bt[t] ~ dlnorm(cpuem[t],isigma2);

 }

 # priors

 log.alpha <- log((startbio[1]+startbio[2])/2) # needed for fit of first biomass

 sd.log.alpha <- (log.alpha-log(startbio[1]))/4

 tau.log.alpha <- pow(sd.log.alpha,-2)

 alpha ~ dlnorm(log.alpha,tau.log.alpha)

 # search in the k space starting from 20% of the range

 log.km <- log(start.k[1]+0.2\*(start.k[2]-start.k[1]))

 sd.log.k <- (log.km-log(start.k[1]))/4

 tau.log.k <- pow(sd.log.k,-2)

 k ~ dlnorm(log.km,tau.log.k)

 # set realistic prior for q

 log.qm <- mean(log(q.prior))

 sd.log.q <- (log.qm-log(q.prior[1]))/4

 tau.log.q <- pow(sd.log.q,-2)

 q ~ dlnorm(log.qm,tau.log.q)

 # define process (tau) and observation (sigma) variances as inversegamma prios

 itau2 ~ dgamma(4,0.01)

 tau2 <- 1/itau2

 tau <- pow(tau2,0.5)

 isigma2 ~ dgamma(2,0.01)

 sigma2 <- 1/isigma2

 sigma <- pow(sigma2,0.5)

 log.rm <- mean(log(start.r))

 sigma.log.r <- abs(log.rm - log(start.r[1]))/2

 tau.log.r <- pow(sigma.log.r,-2)

 r ~ dlnorm(log.rm,tau.log.r)

 } " # end of JAGS model for CPUE

 } # end of else loop for Schaefer with CPUE

 # Write JAGS model to file

 cat(Model, file="r2jags.bug")

 if(btype=="biomass") {

 j.inits <- function(){list("r"=rnorm(1,mean=init.r,sd=0.2\*init.r),

 "k"=rnorm(1,mean=init.k,sd=0.1\*init.k),

 "itau2"=1000,

 "isigma2"=1000)}} else {

 j.inits <- function(){list("r"=rnorm(1,mean=init.r,sd=0.2\*init.r),

 "k"=rnorm(1,mean=init.k,sd=0.1\*init.k),

 "q"=rnorm(1,mean=init.q,sd=0.2\*init.q),

 "itau2"=1000,

 "isigma2"=1000)}}

 # run model

 jags\_outputs <- jags.parallel(data=jags.data,

 working.directory=NULL, inits=j.inits,

 parameters.to.save=jags.save.params,

 model.file="r2jags.bug", n.chains = n.chains,

 n.burnin = 30000, n.thin = 10,

 n.iter = 60000)

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

 # Results from JAGS Schaefer

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

 r\_raw <- as.numeric(mcmc(jags\_outputs$BUGSoutput$sims.list$r))

 k\_raw <- as.numeric(mcmc(jags\_outputs$BUGSoutput$sims.list$k))

 # Importance sampling: only accept r-k pairs where r is near the prior range

 r\_out <- r\_raw[r\_raw > 0.5\*start.r[1] & r\_raw < 1.5 \* start.r[2]]

 k\_out <- k\_raw[r\_raw > 0.5\*start.r[1] & r\_raw < 1.5 \* start.r[2]]

 mean.log.r.jags <- mean(log(r\_out))

 sd.log.r.jags <- sd(log(r\_out))

 r.jags <- exp(mean.log.r.jags)

 lcl.r.jags <- exp(mean.log.r.jags - 1.96\*sd.log.r.jags)

 ucl.r.jags <- exp(mean.log.r.jags + 1.96\*sd.log.r.jags)

 mean.log.k.jags <- mean(log(k\_out))

 sd.log.k.jags <- sd(log(k\_out))

 k.jags <- exp(mean.log.k.jags)

 lcl.k.jags <- exp(mean.log.k.jags - 1.96\*sd.log.k.jags)

 ucl.k.jags <- exp(mean.log.k.jags + 1.96\*sd.log.k.jags)

 MSY.posterior <- r\_out\*k\_out/4 # simpler

 mean.log.MSY.jags <- mean(log(MSY.posterior))

 sd.log.MSY.jags <- sd(log(MSY.posterior))

 MSY.jags <- exp(mean.log.MSY.jags)

 lcl.MSY.jags <- exp(mean.log.MSY.jags - 1.96\*sd.log.MSY.jags)

 ucl.MSY.jags <- exp(mean.log.MSY.jags + 1.96\*sd.log.MSY.jags)

 if(btype=="CPUE") {

 q\_out <- as.numeric(mcmc(jags\_outputs$BUGSoutput$sims.list$q))

 mean.log.q <- mean(log(q\_out))

 sd.log.q <- sd(log(q\_out))

 mean.q <- exp(mean.log.q)

 lcl.q <- exp(mean.log.q-1.96\*sd.log.q)

 ucl.q <- exp(mean.log.q+1.96\*sd.log.q)

 F.bt.cpue <- mean.q\*ct.raw/bt

 Fmsy.cpue <- r.jags/2

 }

 # get F from observed biomass

 if(btype == "biomass") {

 F.bt <- ct.raw/bt

 Fmsy.bt <- r.jags/2

 }

 # get relative biomass P=B/k as predicted by BSM, including predictions for years with NA abundance

 all.P <- jags\_outputs$BUGSoutput$sims.list$P # matrix with P distribution by year

 quant.P <- apply(all.P,2,quantile,c(0.025,0.5,0.975),na.rm=T)

 # get k, r posterior ><>

 all.k <- jags\_outputs$BUGSoutput$sims.list$k # matrix with P distribution by year

 all.r <- jags\_outputs$BUGSoutput$sims.list$r # matrix with P distribution by year

 # get B/Bmys posterior

 all.b\_bmsy=NULL

 for(t in 1:ncol(all.P)){

 all.b\_bmsy <- cbind(all.b\_bmsy,all.P[,t]\*2)}

 # get F/Fmys posterior ><>

 all.F\_Fmsy=NULL

 for(t in 1:ncol(all.P)){

 all.F\_Fmsy<- cbind(all.F\_Fmsy,(ct.raw[t]/(all.P[,t]\*all.k))/ifelse(all.P[,t]>0.25,all.r/2,all.r/2\*4\*all.P[,t]))}

 } # end of MCMC Schaefer loop

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

 # get results from CMSY

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

 # get estimate of most probable r as 75th percentile of mid log.r-classes

 # get unique combinations of r-k

 unique.rk <- unique(mdat.all[,1:2])

 # get remaining viable log.r and log.k

 log.rs <- log(unique.rk[,1])

 log.ks <- log(unique.rk[,2])

 # get vectors with numbers of r and mid values in classes

 # determine number of classes as a function of r-width

 r.width <- (max(unique.rk[,1])-start.r[1])/(start.r[2]-start.r[1])

 classes <- ifelse(r.width>0.8,100,ifelse(r.width>0.5,50,ifelse(r.width>0.3,25,12)))

 hist.log.r <- hist(x=log.rs, breaks=classes, plot=F)

 log.r.counts <- hist.log.r$counts

 log.r.mids <- hist.log.r$mids

 # get most probable log.r as 75th percentile of mids with counts > 0

 log.r.est <- as.numeric(quantile(log.r.mids[which(log.r.counts > 0)],0.75))

 median.log.r <- as.numeric(quantile(x=log.r.mids[which(log.r.counts > 0)], 0.50))

 lcl.log.r <- as.numeric(quantile(x=log.r.mids[which(log.r.counts > 0)], 0.5125))

 ucl.log.r <- as.numeric(quantile(x=log.r.mids[which(log.r.counts > 0)], 0.9875))

 sd.log.r.est <- (ucl.log.r - log.r.est) / 1.96

 r.est <- exp(log.r.est)

 lcl.r.est <- exp(log.r.est-1.96\*sd.log.r.est)

 ucl.r.est <- exp(log.r.est+1.96\*sd.log.r.est)

 # get r-k pairs above median of mids

 rem <- which(unique.rk[,1] > exp(median.log.r))

 rem.log.r <- log(unique.rk[,1][rem])

 rem.log.k <- log(unique.rk[,2][rem])

 # do linear regression of log k ~ log r with slope fixed to -1 (from Schaefer)

 reg <- lm(rem.log.k ~ 1 + offset(-1\*rem.log.r))

 int.reg <- as.numeric(reg[1])

 sd.reg <- sd(resid(reg))

 # get estimate of log(k) from y where x = log.r.est

 log.k.est <- int.reg + (-1) \* log.r.est

 # get estimates of ucl of log.k.est from y + SD where x = ucl.log.r

 ucl.log.k <- int.reg + (-1) \* lcl.log.r + sd.reg

 # get estimates of sd.log.k.est from upper confidence limit of log.k.est

 sd.log.k.est <- (ucl.log.k - log.k.est) / 1.96

 lcl.log.k <- log.k.est - 1.96\*sd.log.k.est

 ucl.log.k <- log.k.est + 1.96\*sd.log.k.est

 k.est <- exp(log.k.est)

 lcl.k.est <- exp(lcl.log.k)

 ucl.k.est <- exp(ucl.log.k)

 # get MSY from remaining log r-k pairs

 log.MSY.est <- mean(rem.log.r + rem.log.k - log(4))

 sd.log.MSY.est <- sd(rem.log.r + rem.log.k - log(4))

 lcl.log.MSY.est <- log.MSY.est - 1.96\*sd.log.MSY.est

 ucl.log.MSY.est <- log.MSY.est + 1.96\*sd.log.MSY.est

 MSY.est <- exp(log.MSY.est)

 lcl.MSY.est <- exp(lcl.log.MSY.est)

 ucl.MSY.est <- exp(ucl.log.MSY.est)

 # get predicted biomass vectors as median and quantiles

 # only use biomass trajectories from r-k pairs within the confidence limits

 rem.btv.all <- mdat.all[which(mdat.all[,1] > lcl.r.est & mdat.all[,1] < ucl.r.est

 & mdat.all[,2] > lcl.k.est & mdat.all[,2] < ucl.k.est),3:(2+nyr+1)]

 median.btv <- apply(rem.btv.all,2, median)

 median.btv.lastyr <- median.btv[length(median.btv)-1]

 nextyr.bt <- median.btv[length(median.btv)]

 lcl.btv <- apply(rem.btv.all,2, quantile, probs=0.025)

 q.btv <- apply(rem.btv.all,2, quantile, probs=0.25)

 ucl.btv <- apply(rem.btv.all,2, quantile, probs=0.975)

 lcl.median.btv.lastyr <- lcl.btv[length(lcl.btv)-1]

 ucl.median.btv.lastyr <- ucl.btv[length(lcl.btv)-1]

 lcl.nextyr.bt <- lcl.btv[length(lcl.btv)]

 ucl.nextyr.bt <- ucl.btv[length(lcl.btv)]

 # get F derived from predicted CMSY biomass

 F.CMSY <- ct.raw/(median.btv[1:nyr]\*k.est)

 Fmsy.CMSY <- r.est/2 # Fmsy from CMSY

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

 # Get results for management

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

 if(FullSchaefer==F | force.cmsy==T) { # if only CMSY is available or shall be used

 MSY <-MSY.est; lcl.MSY<-lcl.MSY.est; ucl.MSY<-ucl.MSY.est

 Bmsy <-k.est/2; lcl.Bmsy<-lcl.k.est/2; ucl.Bmsy<-ucl.k.est/2

 Fmsy <-r.est/2; lcl.Fmsy<-lcl.r.est/2; ucl.Fmsy<-ucl.r.est/2

 B.Bmsy<-2\*median.btv[1:nyr];lcl.B.Bmsy<-2\*lcl.btv[1:nyr];ucl.B.Bmsy<-2\*ucl.btv[1:nyr]

 if(is.na(sel.yr)==F){B.Bmsy.sel<-2\*median.btv[yr==sel.yr]}

 } else {

 MSY <-MSY.jags; lcl.MSY<-lcl.MSY.jags; ucl.MSY<-ucl.MSY.jags

 Bmsy <-k.jags/2; lcl.Bmsy<-lcl.k.jags/2; ucl.Bmsy<-ucl.k.jags/2

 Fmsy <-r.jags/2; lcl.Fmsy<-lcl.r.jags/2; ucl.Fmsy<-ucl.r.jags/2

 B.Bmsy<-2\*quant.P[2,];lcl.B.Bmsy<-2\*quant.P[1,];ucl.B.Bmsy<-2\*quant.P[3,]

 if(is.na(sel.yr)==F) {B.Bmsy.sel<-2\*quant.P[2,][yr==sel.yr]}

 }

 B <-B.Bmsy\*Bmsy;lcl.B<-lcl.B.Bmsy\*Bmsy;ucl.B<-ucl.B.Bmsy\*Bmsy

 B.last <-B[nyr];lcl.B.last<-lcl.B[nyr];ucl.B.last<-ucl.B[nyr]

 B.Bmsy.last<-B.Bmsy[nyr];lcl.B.Bmsy.last<-lcl.B.Bmsy[nyr];ucl.B.Bmsy.last<-ucl.B.Bmsy[nyr]

 Fm <- ct.raw/B;lcl.F<-ct.raw/ucl.B;ucl.F<-ct.raw/lcl.B

 Fmsy.vec <- ifelse(B.Bmsy>0.5,Fmsy,Fmsy\*2\*B.Bmsy)

 lcl.Fmsy.vec <- ifelse(B.Bmsy>0.5,lcl.Fmsy,lcl.Fmsy\*2\*B.Bmsy)

 ucl.Fmsy.vec <- ifelse(B.Bmsy>0.5,ucl.Fmsy,ucl.Fmsy\*2\*B.Bmsy)

 F.Fmsy <- Fm/Fmsy.vec; lcl.F.Fmsy<-lcl.F/Fmsy.vec; ucl.F.Fmsy<-ucl.F/Fmsy.vec

 F.last <-Fm[nyr];lcl.F.last<-lcl.F[nyr];ucl.F.last<-ucl.F[nyr]

 Fmsy.last <-Fmsy.vec[nyr];lcl.Fmsy.last<-lcl.Fmsy.vec[nyr];ucl.Fmsy.last<-ucl.Fmsy.vec[nyr]

 F.Fmsy.last<-F.Fmsy[nyr];lcl.F.Fmsy.last<-lcl.F.Fmsy[nyr];ucl.F.Fmsy.last<-ucl.F.Fmsy[nyr]

 if(is.na(sel.yr)==F){

 B.sel<-B.Bmsy.sel\*Bmsy

 F.sel<-ct.raw[yr==sel.yr]/B.sel

 F.Fmsy.sel<-F.sel/Fmsy.vec[yr==sel.yr]

 }

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

 # print input and results to screen

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

 cat("---------------------------------------\n")

 cat("Species:", cinfo$ScientificName[cinfo$Stock==stock], ", stock:",stock,"\n")

 cat(cinfo$Name[cinfo$Stock==stock], "\n")

 cat("Region:",cinfo$Region[cinfo$Stock==stock],",",cinfo$Subregion[cinfo$Stock==stock],"\n")

 cat("Catch data used from years", min(yr),"-", max(yr),", abundance =", btype, "\n")

 cat("Prior initial relative biomass =", startbio[1], "-", startbio[2],ifelse(is.na(stb.low)==T,"default","expert"), "\n")

 cat("Prior intermediate rel. biomass=", intbio[1], "-", intbio[2], "in year", int.yr,ifelse(is.na(intb.low)==T,"default","expert"), "\n")

 cat("Prior final relative biomass =", endbio[1], "-", endbio[2],ifelse(is.na(endb.low)==T,"default","expert"), "\n")

 cat("Prior range for r =", format(start.r[1],digits=2), "-", format(start.r[2],digits=2),ifelse(is.na(r.low)==T,"default","expert,"),

 ", prior range for k =", start.k[1], "-", start.k[2],"\n")

 # if Schaefer and CPUE, print prior range of q

 if(FullSchaefer==T & btype=="CPUE") {

 cat("Prior range of q =",q.prior[1],"-",q.prior[2],"\n")

 }

 # results of CMSY analysis

 cat("\nResults of CMSY analysis \n")

 cat("-------------------------\n")

 cat("Altogether", n.viable.b, "viable trajectories for", n.viable.pt," r-k pairs were found \n")

 cat("r =", r.est,", 95% CL =", lcl.r.est, "-", ucl.r.est,", k =", k.est,", 95% CL =", lcl.k.est, "-", ucl.k.est,"\n")

 cat("MSY =", MSY.est,", 95% CL =", lcl.MSY.est, "-", ucl.MSY.est,"\n")

 cat("Relative biomass in last year =", median.btv.lastyr, "k, 2.5th perc =", lcl.median.btv.lastyr,

 ", 97.5th perc =", ucl.median.btv.lastyr,"\n")

 cat("Exploitation F/(r/2) in last year =", (F.CMSY/Fmsy.CMSY)[nyr],"\n\n")

 # print results from full Schaefer if available

 if(FullSchaefer==T) {

 cat("Results from Bayesian Schaefer model (BSM) using catch &",btype,"\n")

 cat("------------------------------------------------------------\n")

 if(btype == "CPUE") cat("q =", mean.q,", lcl =", lcl.q, ", ucl =", ucl.q,"\n")

 cat("r =", r.jags,", 95% CL =", lcl.r.jags, "-", ucl.r.jags,", k =", k.jags,", 95% CL =", lcl.k.jags, "-", ucl.k.jags,"\n")

 cat("MSY =", MSY.jags,", 95% CL =", lcl.MSY.jags, "-", ucl.MSY.jags,"\n")

 cat("Relative biomass in last year =", quant.P[2,][nyr], "k, 2.5th perc =",quant.P[1,][nyr],

 ", 97.5th perc =", quant.P[3,][nyr],"\n")

 cat("Exploitation F/(r/2) in last year =", (ct.raw[nyr]/(quant.P[2,][nyr]\*k.jags))/(r.jags/2) ,"\n\n")

 }

 # print results to be used in management

 cat("Results for Management (based on",ifelse(FullSchaefer==F | force.cmsy==T,"CMSY","BSM"),"analysis) \n")

 cat("-------------------------------------------------------------\n")

 if(force.cmsy==T) cat("Mangement results based on CMSY because abundance data seem unrealistic\n")

 cat("Fmsy =",Fmsy,", 95% CL =",lcl.Fmsy,"-",ucl.Fmsy,"(if B > 1/2 Bmsy then Fmsy = 0.5 r)\n")

 cat("Fmsy =",Fmsy.last,", 95% CL =",lcl.Fmsy.last,"-",ucl.Fmsy.last,"(r and Fmsy are linearly reduced if B < 1/2 Bmsy)\n")

 cat("MSY =",MSY,", 95% CL =",lcl.MSY,"-",ucl.MSY,"\n")

 cat("Bmsy =",Bmsy,", 95% CL =",lcl.Bmsy,"-",ucl.Bmsy,"\n")

 cat("Biomass in last year =",B.last,", 2.5th perc =", lcl.B.last, ", 97.5 perc =",ucl.B.last,"\n")

 cat("B/Bmsy in last year =",B.Bmsy.last,", 2.5th perc =", lcl.B.Bmsy.last, ", 97.5 perc =",ucl.B.Bmsy.last,"\n")

 cat("Fishing mortality in last year =",F.last,", 2.5th perc =", lcl.F.last, ", 97.5 perc =",ucl.F.last,"\n")

 cat("Exploitation F/Fmsy =",F.Fmsy.last,", 2.5th perc =", lcl.F.Fmsy.last, ", 97.5 perc =",ucl.F.Fmsy.last,"\n")

 # show stock status and exploitation for optional selected year

 if(is.na(sel.yr)==F) {

 cat("\nStock status and exploitation in",sel.yr,"\n")

 cat("Biomass =",B.sel, ", B/Bmsy =",B.Bmsy.sel,", F =",F.sel,", F/Fmsy =",F.Fmsy.sel,"\n") }

 # indicate if less than 5 years of biomass or CPUE are available

 if(btype !="None" & length(bt[is.na(bt)==F])<nab) {

 cat("Less than",nab,"years with abundance data available, shown on second axis\n")

 }

 cat("Comment:", comment,"\n")

 cat("----------------------------------------------------------\n")

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

 # Plot results

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

 # Analysis of viable r-k plot

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

 max.y <- max(c(ifelse(FullSchaefer==T,ucl.k.jags,NA), max(kv.all)),

 ifelse(substr(id\_file,1,3)=="Sim",1.2\*true.k,max(kv.all)),

 na.rm=T)

 min.y <- min(c(ifelse(FullSchaefer==T,lcl.k.jags,NA), 0.9\*min(kv.all)),

 ifelse(substr(id\_file,1,3)=="Sim",0.8\*true.k,0.9\*min(kv.all)),

 na.rm=T)

 plot(x=rv.all, y=kv.all, xlim=start.r,

 ylim=c(min.y,max.y),

 pch=16, col="gray",log="xy", bty="l",

 xlab="r", ylab="k", main="C: Analysis of viable r-k")

 # plot r-k pairs from MCMC

 if(FullSchaefer==T) {points(x=r\_out, y=k\_out, pch=16,cex=0.5)}

 # plot best r-k from full Schaefer analysis

 if(FullSchaefer==T) {

 points(x=r.jags, y=k.jags, pch=19, col="red")

 lines(x=c(lcl.r.jags, ucl.r.jags),y=c(k.jags,k.jags), col="red")

 lines(x=c(r.jags,r.jags),y=c(lcl.k.jags, ucl.k.jags), col="red")

 }

 # plot blue dot for CMSY r-k, with 95% CL lines

 points(x=r.est, y=k.est, pch=19, col="blue")

 lines(x=c(lcl.r.est, ucl.r.est),y=c(k.est,k.est), col="blue")

 lines(x=c(r.est,r.est),y=c(lcl.k.est, ucl.k.est), col="blue")

 # Pred. biomass plot

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

 # determine k to use for red line in b/k plot

 if(FullSchaefer==T) {k2use <- k.jags} else {k2use <- k.est}

 # determine hight of y-axis in plot

 max.y <- max(c(ifelse(btype=="biomass",max(bt/k2use,na.rm=T),NA),

 ifelse(btype=="CPUE",max(bt/(mean.q\*k2use),na.rm=T),NA),

 max(ucl.btv),0.6,startbio[2], endbio[2]),

 ifelse(FullSchaefer==T & btype=="biomass",max(bt[is.na(bt)==F]/lcl.k.jags,na.rm=T),NA),

 ifelse(FullSchaefer==T & btype=="CPUE",1.1\*max(bt/(mean.q\*lcl.k.jags),na.rm=T),NA), na.rm=T)

 # Main plot of relative CMSY biomass

 plot(x=yr,y=median.btv[1:nyr], lwd=1.5, xlab="Year", ylab="Relative biomass B/k", type="l",

 ylim=c(0,max.y), bty="l", main="D: Biomass",col="blue")

 lines(x=yr, y=lcl.btv[1:nyr],type="l",lty="dotted",col="blue")

 lines(x=yr, y=ucl.btv[1:nyr],type="l",lty="dotted",col="blue")

 # plot lines for 0.5 and 0.25 biomass

 abline(h=0.5, lty="dashed")

 abline(h=0.25, lty="dotted")

 # plot biomass windows

 lines(x=c(yr[1],yr[1]), y=startbio, col="blue")

 lines(x=c(int.yr,int.yr), y=intbio, col="blue")

 lines(x=c(max(yr),max(yr)), y=endbio, col="blue")

 # if observed biomass is available, plot red biomass line (use non-smoothed bt)

 if(btype=="biomass" & FullSchaefer==T) {

 lines(x=yr, y=bt/k.jags,type="l", col="red", lwd=1)

 lines(x=yr, y=bt/ucl.k.jags,type="l",col="red",lty="dotted")

 lines(x=yr, y=bt/lcl.k.jags,type="l",col="red",lty="dotted")

 }

 # if observed CPUE is available, plot red biomass line

 if(btype=="CPUE" & FullSchaefer==T) {

 lines(x=yr, y=bt/(mean.q\*k.jags),type="l", col="red", lwd=1)

 lines(x=yr, y=bt/(mean.q\*ucl.k.jags),type="l",col="red",lty="dotted")

 lines(x=yr, y=bt/(mean.q\*lcl.k.jags),type="l",col="red",lty="dotted")

 }

 # if biomass or CPUE data are available but fewer than 5 years, plot on second axis

 if(btype != "None" & FullSchaefer==F) {

 par(new=T) # prepares for new plot on top of previous

 plot(x=yr, y=bt, type="l", col="red", lwd=1,

 ann=F,axes=F,ylim=c(0,1.2\*max(bt, na.rm=T))) # forces this plot on top of previous one

 axis(4, col="red", col.axis="red")

 }

 # Exploitation rate plot

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

 # if CPUE data are available but fewer than nab years, plot on second axis

 if(btype == "CPUE") {

 q=1/(max(median.btv[1:nyr][is.na(bt)==F],na.rm=T)\*k.est/max(bt,na.rm=T))

 u.cpue <- q\*ct/bt

 }

 # determine upper bound of Y-axis

 max.y <- max(c(1.5, 1.2\*F.CMSY/Fmsy.CMSY,max(F.CMSY/Fmsy.CMSY),

 ifelse(btype=="biomass" & FullSchaefer==T,max(F.bt[is.na(F.bt)==F]/Fmsy.bt),0),

 ifelse(FullSchaefer==T & btype=="CPUE",max(F.bt.cpue/Fmsy.cpue,na.rm=T),0),

 na.rm=T))

 # plot F from CMSY

 plot(x=yr,y=F.CMSY/Fmsy.CMSY, type="l", bty="l", lwd=1.5, ylim=c(0,max.y), xlab="Year",

 ylab="F / (r/2)", main="E: Exploitation rate", col="blue")

 abline(h=1, lty="dashed")

 # plot F from observed biomass

 if(btype == "biomass" & FullSchaefer==T) lines(x=yr, y=F.bt/Fmsy.bt, col="red")

 # plot F from observed CPUE

 if(FullSchaefer==T & btype == "CPUE") lines(x=yr, y=F.bt.cpue/Fmsy.cpue, col="red")

 # plot F from CPUE on second axis if less than 5 years

 if(FullSchaefer==F & btype == "CPUE") {

 par(new=T) # prepares for new plot on top of previous

 plot(x=yr, y=F.bt.cpue, type="l", col="red", ylim=c(0, 1.2\*max(F.bt.cpue,na.rm=T)),ann=F,axes=F)

 axis(4, col="red", col.axis="red")

 }

 # Parabola plot

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

 max.y <- max(c(max(ct/MSY.est),ifelse(btype=="biomass",max(ct/MSY.jags),NA),1.2),na.rm=T)

 # plot parabola

 x=seq(from=0,to=2,by=0.001)

 y.c <- ifelse(x>0.25,1,ifelse(x>0.125,4\*x,exp(-10\*(0.125-x))\*4\*x)) # correction for low recruitment below half and below quarter of Bmsy

 y=(4\*x-(2\*x)^2)\*y.c

 plot(x=x, y=y, xlim=c(1,0), ylim=c(0,max.y), type="l", bty="l",xlab="Relative biomass B/k",

 ylab="Catch / MSY", main="F: Equilibrium curve")

 # plot catch against CMSY estimates of relative biomass

 points(x=median.btv[1:nyr], y=ct/MSY.est, pch=16, col="blue")

 # plot catch scaled by BSM MSY against observed biomass scaled by BSM k

 if(btype == "biomass") {

 points(x=bt/k.jags, y=ct/MSY.jags, pch=16, cex=0.5, col="red")

 }

 # for CPUE, plot catch scaled by BSM MSY against observed biomass derived as q \* CPUE scaled by BSM k

 if(FullSchaefer==T & btype=="CPUE") {

 points(x=bt/(mean.q\*k.jags), y=ct/MSY.jags, pch=16, cex=0.5, col="red")

 }

 #save analytic chart to JPEG file

 if (save.plots==TRUE)

 {

 jpgfile<-paste(stock,"\_AN.jpeg",sep="")

 dev.copy(jpeg,jpgfile,

 width = 1024,

 height = 768,

 units = "px",

 pointsize = 18,

 quality = 95,

 res=80,

 antialias="cleartype")

 dev.off()

 }

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

 # Plot Management-Graphs if desired

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

 if(mgraphs==T) {

 # open window for plot of four panels

 if(grepl("win",tolower(Sys.info()['sysname']))) {windows(14,12)}

 par(mfrow=c(2,2))

 # make margins narrower

 par(mar=c(3.1,4.1,2.1,2.1))

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

 # plot catch with MSY

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

 max.y <- max(c(1.1\*max(ct.raw),ucl.MSY),na.rm=T)

 plot(x=yr,rep(0,nyr),type="n",ylim=c(0,max.y), bty="l", main=paste("Catch",stock),

 ylab="Catch in 1000 t")

 rect(yr[1],lcl.MSY,yr[nyr],ucl.MSY,col="lightgray", border=NA)

 lines(x=c(yr[1],yr[nyr]),y=c(MSY,MSY),lty="dashed", col="black", lwd=1.5)

 lines(x=yr, y=ct.raw, lwd=2)

 text("MSY",x=end.yr-1.5, y=MSY+MSY\*0.1)

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

 # plot estimated biomass relative to Bmsy

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

 # plot empty frame

 plot(yr, rep(0,nyr),type="n", ylim=c(0,max(c(2, max(ucl.B.Bmsy)))), ylab="B / Bmsy",xlab="Year", main="Biomass", bty="l")

 # plot gray area of uncertainty in predicted biomass

 polygon(c(yr,rev(yr)), c(lcl.B.Bmsy,rev(ucl.B.Bmsy)),col="lightgray", border=NA)

 # plot median biomass

 lines(yr,B.Bmsy,lwd=2)

 # plot lines for Bmsy and 0.5 Bmsy

 lines(x=c(yr[1],yr[nyr]),y=c(1,1), lty="dashed", lwd=1.5)

 lines(x=c(yr[1],yr[nyr]),y=c(0.5,0.5), lty="dotted", lwd=1.5)

 # plot exploitation rate

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

 # plot empty frame

 plot(yr, rep(0,nyr),type="n", ylim=c(0,max(c(2, ifelse(max(ucl.F.Fmsy)<5,max(ucl.F.Fmsy),5)))),

 ylab="F / Fmsy",xlab="Year", main="Exploitation", bty="l")

 # plot gray area of uncertainty in predicted exploitation

 polygon(c(yr,rev(yr)), c(lcl.F.Fmsy,rev(ucl.F.Fmsy)),col="lightgray", border=NA)

 # plot median exploitation rate

 lines(x=yr,y=F.Fmsy,lwd=2)

 # plot line for u.msy

 lines(x=c(yr[1],yr[nyr]),y=c(1,1), lty="dashed", lwd=1.5)

 #><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>><>

 # plot stock-status graph

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

 if(FullSchaefer==T & force.cmsy==F) {x.F\_Fmsy = all.F\_Fmsy[,nyr]

 y.b\_bmsy = all.b\_bmsy[,nyr]} else {

 log.sd.B.Bmsy = (log(ucl.B.Bmsy.last+0.0011)-log(lcl.B.Bmsy.last+0.001))/(2\*1.96)

 log.sd.F.Fmsy = (log(ucl.F.Fmsy.last+0.005)-log(lcl.F.Fmsy.last+0.001))/(2\*1.96)

 x.F\_Fmsy= rlnorm(20000,log(F.Fmsy.last+0.001),log.sd.F.Fmsy)

 y.b\_bmsy =rlnorm(20000,log(B.Bmsy.last+0.001),log.sd.B.Bmsy)

 }

 kernelF <- ci2d(x.F\_Fmsy,y.b\_bmsy,nbins=201,factor=2.2,ci.levels=c(0.50,0.80,0.75,0.90,0.95),show="none")

 c1 <- c(-1,100)

 c2 <- c(1,1)

 max.x1 <- max(c(2, max(kernelF$contours$"0.95"$x,ucl.F.Fmsy.last),na.rm =T))

 max.x <- ifelse(max.x1 > 5,min(max(5,F.Fmsy\*2),8),max.x1)

 max.y <- max(max(2,quantile(y.b\_bmsy,0.96)))

 #Create plot

 plot(1000,1000,type="b", xlim=c(0,max.x), ylim=c(0,max.y),lty=3,xlab="",ylab="", bty="l")

 mtext("F / Fmsy",side=1, line=2)

 mtext("B / Bmsy",side=2, line=2)

 # 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")

 ## Add points and trajectory lines

 lines(c1,c2,lty=3,lwd=0.7)

 lines(c2,c1,lty=3,lwd=0.7)

 lines(F.Fmsy,B.Bmsy, lty=1,lwd=1.)

 points(F.Fmsy,B.Bmsy,cex=0.8,pch=4)

 points(F.Fmsy[1],B.Bmsy[1],col=1,pch=22,bg="white",cex=1.9)

 points(F.Fmsy[which(yr==int.yr)],B.Bmsy[which(yr==int.yr)],col=1,pch=21,bg="white",cex=1.9)

 points(F.Fmsy[nyr],B.Bmsy[nyr],col=1,pch=24,bg="white",cex=1.9)

 ## Add legend

 legend('topright', c(paste(start.yr),paste(int.yr),paste(end.yr),"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(1.3,3),1.7,1.7,1.7),bty="n",y.intersp = 0.9)

 #><> End of Biplot

 } # end of management graphs

 # save management chart to JPEG file

 if (save.plots==TRUE & mgraphs==TRUE)

 {

 jpgfile<-paste(stock,"\_MAN.jpeg",sep="")

 dev.copy(jpeg,jpgfile,

 width = 1024,

 height = 768,

 units = "px",

 pointsize = 18,

 quality = 95,

 res=80,

 antialias="cleartype")

 dev.off()

 }

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

 ## Write some results into csv outfile

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

 if(write.output == TRUE) {

 # write data into csv file

 output = data.frame(as.character(cinfo$Group[cinfo$Stock==stock]),

 as.character(cinfo$Region[cinfo$Stock==stock]),

 as.character(cinfo$Subregion[cinfo$Stock==stock]),

 as.character(cinfo$Name[cinfo$Stock==stock]),

 cinfo$ScientificName[cinfo$Stock==stock],

 stock, start.yr, end.yr, btype,

 max(ct.raw),ct.raw[nyr],

 ifelse(FullSchaefer==T,MSY.jags,NA), # full Schaefer

 ifelse(FullSchaefer==T,lcl.MSY.jags,NA),

 ifelse(FullSchaefer==T,ucl.MSY.jags,NA),

 ifelse(FullSchaefer==T,r.jags,NA),

 ifelse(FullSchaefer==T,lcl.r.jags,NA),

 ifelse(FullSchaefer==T,ucl.r.jags,NA),

 ifelse(FullSchaefer==T,k.jags,NA),

 ifelse(FullSchaefer==T,lcl.k.jags,NA),

 ifelse(FullSchaefer==T,ucl.k.jags,NA),

 ifelse(FullSchaefer==T & btype=="CPUE",mean.q,NA),

 ifelse(FullSchaefer==T & btype=="CPUE",lcl.q,NA),

 ifelse(FullSchaefer==T & btype=="CPUE",ucl.q,NA),

 ifelse(FullSchaefer==T,quant.P[2,][nyr],NA), # last B/k JAGS

 ifelse(FullSchaefer==T,quant.P[1,][nyr],NA),

 ifelse(FullSchaefer==T,quant.P[3,][nyr],NA),

 ifelse(FullSchaefer==T,(ct.raw[nyr]/(quant.P[2,][nyr]\*k.jags))/(r.jags/2),NA), # last F/Fmsy JAGS

 r.est, lcl.r.est, ucl.r.est, # CMSY r

 k.est, lcl.k.est, ucl.k.est, # CMSY k

 MSY.est, lcl.MSY.est, ucl.MSY.est, # CMSY MSY

 median.btv.lastyr, lcl.median.btv.lastyr,ucl.median.btv.lastyr, # CMSY B/k in last year with catch data

 (F.CMSY/Fmsy.CMSY)[nyr],

 Fmsy,lcl.Fmsy,ucl.Fmsy,Fmsy.last,lcl.Fmsy.last,ucl.Fmsy.last,

 MSY,lcl.MSY,ucl.MSY,Bmsy,lcl.Bmsy,ucl.Bmsy,

 B.last, lcl.B.last, ucl.B.last, B.Bmsy.last, lcl.B.Bmsy.last, ucl.B.Bmsy.last,

 F.last, lcl.F.last, ucl.F.last, F.Fmsy.last, lcl.F.Fmsy.last, ucl.F.Fmsy.last,

 ifelse(is.na(sel.yr)==F,B.sel,NA),

 ifelse(is.na(sel.yr)==F,B.Bmsy.sel,NA),

 ifelse(is.na(sel.yr)==F,F.sel,NA),

 ifelse(is.na(sel.yr)==F,F.Fmsy.sel,NA),

 ifelse(yr[1]>2000,NA,ct.raw[yr==2000]),ifelse(yr[1]>2001,NA,ct.raw[yr==2001]),ifelse(yr[1]>2002,NA,ct.raw[yr==2002]), ifelse(yr[1]>2003,NA,ct.raw[yr==2003]),# allow missing 2000-2002

 ct.raw[yr==2004],ct.raw[yr==2005],ct.raw[yr==2006],ct.raw[yr==2007],ct.raw[yr==2008],ct.raw[yr==2009],ct.raw[yr==2010],

 ifelse(yr[nyr]<2011,NA,ct.raw[yr==2011]),ifelse(yr[nyr]<2012,NA,ct.raw[yr==2012]),ifelse(yr[nyr]<2013,NA,ct.raw[yr==2013]),ifelse(yr[nyr]<2014,NA,ct.raw[yr==2014]),ifelse(yr[nyr]<2015,NA,ct.raw[yr==2015]), # allow missing 2011-2015

 ifelse(yr[1]>2000,NA,F.Fmsy[yr==2000]),ifelse(yr[1]>2001,NA,F.Fmsy[yr==2001]),ifelse(yr[1]>2002,NA,F.Fmsy[yr==2002]),ifelse(yr[1]>2003,NA,F.Fmsy[yr==2003]), # allow missing 2000-2002

 F.Fmsy[yr==2004],F.Fmsy[yr==2005],F.Fmsy[yr==2006],F.Fmsy[yr==2007],F.Fmsy[yr==2008],F.Fmsy[yr==2009],F.Fmsy[yr==2010],

 ifelse(yr[nyr]<2011,NA,F.Fmsy[yr==2011]),ifelse(yr[nyr]<2012,NA,F.Fmsy[yr==2012]),ifelse(yr[nyr]<2013,NA,F.Fmsy[yr==2013]),ifelse(yr[nyr]<2014,NA,F.Fmsy[yr==2014]),ifelse(yr[nyr]<2015,NA,F.Fmsy[yr==2015]),# allow missing 2011-2015

 ifelse(yr[1]>2000,NA,B[yr==2000]),ifelse(yr[1]>2001,NA,B[yr==2001]),ifelse(yr[1]>2002,NA,B[yr==2002]),ifelse(yr[1]>2003,NA,B[yr==2003]), # allow missing 2000-2002

 B[yr==2004],B[yr==2005],B[yr==2006],B[yr==2007],B[yr==2008],B[yr==2009],B[yr==2010],

 ifelse(yr[nyr]<2011,NA,B[yr==2011]),ifelse(yr[nyr]<2012,NA,B[yr==2012]),ifelse(yr[nyr]<2013,NA,B[yr==2013]),ifelse(yr[nyr]<2014,NA,B[yr==2014]),ifelse(yr[nyr]<2015,NA,B[yr==2015])) # allow missing 2011-2015

 write.table(output, file=outfile, append = T, sep = ",",

 dec = ".", row.names = FALSE, col.names = FALSE)

 # write screen text into text outfile.txt

 cat("Species:", cinfo$ScientificName[cinfo$Stock==stock], ", stock:",stock,"\n",

 cinfo$Name[cinfo$Stock==stock], "\n",

 "Source:",cinfo$Source[cinfo$Stock==stock],"\n",

 "Region:",cinfo$Region[cinfo$Stock==stock],",",cinfo$Subregion[cinfo$Stock==stock],"\n",

 "Catch data used from years", min(yr),"-", max(yr),", abundance =", btype, "\n",

 "Prior initial relative biomass =", startbio[1], "-", startbio[2],ifelse(is.na(stb.low)==T,"default","expert"), "\n",

 "Prior intermediate rel. biomass=", intbio[1], "-", intbio[2], "in year", int.yr,ifelse(is.na(intb.low)==T,"default","expert"), "\n",

 "Prior final relative biomass =", endbio[1], "-", endbio[2],ifelse(is.na(endb.low)==T,", default","expert"), "\n",

 "Prior range for r =", format(start.r[1],digits=2), "-", format(start.r[2],digits=2),ifelse(is.na(r.low)==T,"default","expert,"),

 ", prior range for k =", start.k[1], "-", start.k[2],

 file=outfile.txt,append=T)

 if(FullSchaefer==T & btype=="CPUE") {

 cat("\n Prior range of q =",q.prior[1],"-",q.prior[2],file=outfile.txt,append=T)

 }

 cat("\n\n Results of CMSY analysis with altogether",n.viable.b, "viable trajectories for", n.viable.pt,"r-k pairs \n",

 "r =", r.est,", 95% CL =", lcl.r.est, "-", ucl.r.est,

 ", k =", k.est,", 95% CL =", lcl.k.est, "-", ucl.k.est,"\n",

 "MSY =", MSY.est,", 95% CL =", lcl.MSY.est, "-", ucl.MSY.est,"\n",

 "Relative biomass last year =", median.btv.lastyr, "k, 2.5th =", lcl.median.btv.lastyr,

 ", 97.5th =", ucl.median.btv.lastyr,"\n",

 "Exploitation F/(r/2) in last year =", (F.CMSY/Fmsy.CMSY)[length(median.btv)-1],"\n",

 file=outfile.txt,append=T)

 if(FullSchaefer==T) {

 cat("\n Results from Bayesian Schaefer model using catch &",btype,"\n",

 "r =", r.jags,", 95% CL =", lcl.r.jags, "-", ucl.r.jags,

 ", k =", k.jags,", 95% CL =", lcl.k.jags, "-", ucl.k.jags,"\n",

 "MSY =", MSY.jags,", 95% CL =", lcl.MSY.jags, "-", ucl.MSY.jags,"\n",

 "Relative biomass in last year =", quant.P[2,][nyr], "k, 2.5th perc =",quant.P[1,][nyr],

 ", 97.5th perc =", quant.P[3,][nyr],"\n",

 "Exploitation F/(r/2) in last year =", (ct.raw[nyr]/(quant.P[2,][nyr]\*k.jags))/(r.jags/2) ,

 file=outfile.txt,append=T)

 if(btype == "CPUE") {cat("\n q =", mean.q,", lcl =", lcl.q, ", ucl =", ucl.q,

 file=outfile.txt,append=T)}

 }

 cat("\n\n Results for Management (based on",ifelse(FullSchaefer==F | force.cmsy==T,"CMSY","BSM"),"analysis) \n",

 "Fmsy =",Fmsy,", 95% CL =",lcl.Fmsy,"-",ucl.Fmsy,"(if B > 1/2 Bmsy then Fmsy = 0.5 r)\n",

 "Fmsy =",Fmsy.last,", 95% CL =",lcl.Fmsy.last,"-",ucl.Fmsy.last,"(r and Fmsy are linearly reduced if B < 1/2 Bmsy)\n",

 "MSY =",MSY,", 95% CL =",lcl.MSY,"-",ucl.MSY,"\n",

 "Bmsy =",Bmsy,", 95% CL =",lcl.Bmsy,"-",ucl.Bmsy,"\n",

 "Biomass in last year =",B.last,", 2.5th perc =", lcl.B.last, ", 97.5 perc =",ucl.B.last,"\n",

 "B/Bmsy in last year =",B.Bmsy.last,", 2.5th perc =", lcl.B.Bmsy.last, ", 97.5 perc =",ucl.B.Bmsy.last,"\n",

 "Fishing mortality in last year =",F.last,", 2.5th perc =", lcl.F.last, ", 97.5 perc =",ucl.F.last,"\n",

 "F/Fmsy =",F.Fmsy.last,", 2.5th perc =", lcl.F.Fmsy.last, ", 97.5 perc =",ucl.F.Fmsy.last,"\n",

 file=outfile.txt,append=T)

 # show stock status and exploitation for optional selected year

 if(is.na(sel.yr)==F) {

 cat("\n Stock status and exploitation in",sel.yr,"\n",

 "Biomass =",B.sel, ", B/Bmsy =",B.Bmsy.sel,", fishing mortality F =",F.sel,", F/Fmsy =",F.Fmsy.sel,"\n",

 file=outfile.txt,append=T) }

 if(btype !="None" & length(bt[is.na(bt)==F])<nab) {

 cat(" Less than",nab,"years with abundance data available, shown on second axis\n",file=outfile.txt,append=T) }

 cat(" Comment:", comment,"\n","----------------------------------------------------------\n\n",

 file=outfile.txt,append=T)

 } # end of loop to write text to file

 if(close.plots==T) graphics.off() # close on-screen graphics windows after files are saved

} # end of stocks loop

#stop parallel processing clusters

stopCluster(cl)

stopImplicitCluster()