**Supplementary Materials for “Simulated Fishing to Untangle Catchability and Availability in Fish Abundance Monitoring”**

Author: Vanessa D. Tobias, [vanessa\_tobias@fws.gov](mailto:vanessa_tobias@fws.gov)

Note that the author plans to archive code in an open code repository at the time the manuscript is published. This supplement is provided for reviewers in the interim.

**APPENDIX A: CODE TO GENERATE DATA**

# Front Matter ####

# Author: Vanessa Tobias (vanessa\_tobias@fws.gov)

# Date: 2019-10-22

# Purpose: simulate data to address the relationship between turbidity and catchability of Delta Smelt

# Notes:

# - Code here is based on FMWT characteristics

# Load Libraries ####

#library(circular)

library(CircStats)

# https://rdrr.io/cran/CircStats/man/rwrpnorm.html

# rwrpnorm uses modulo 2\*pi as the default, which gets around the issue with circular::rwrappednormal (apparently) ignoring the modulo argument.

library(doBy)

set.seed(123)

# Define Functions ####

#vel() calculates tow velocity from volume data

# assumes a 12 minute tow for the FMWT

vel<-function(x) {x\*100/(10.704\*720)}

# Create a dataset of fish information

# Returns a data.frame with one row for each fish

# width.path = width of the gear/net in cm

mkfish <- function(n.pops, n.fish, width.path){

# create the data

fish <- data.frame(pop = rep(1:n.pops, each=n.fish), #pop = a population of fish for each tow of the net

distance = runif(n.fish\*n.pops, min = 0, max = width.path), #distance in cm from edge of net = random uniform

#angle = runif(n.fish\*n.pops, min=0, max=2\*pi), #uniform, full circle

angle = rwrpnorm(n = n.fish\*n.pops, #default is to use modulo 2\*pi

mu = 1.32296,

sd = 0.5555139), #angle in radians = WRAPPED NORMAL DISTRIBUTION; #mu = (165.8-90)\*(pi/180); adjust for rotation between Meager and the simulation#Meager et al 2006: Escape trajectory mean=165.8\*, SE=3.7\*, n=74; sigma2=1013.06

#angle = runif(n.fish\*n.pops, min=0, max=pi), #uniform, half circle

fish.vel = rnorm(n.fish\*n.pops, m=27.6, sd=5.1), #rnorm(n.fish\*n.pops, m=27.6, sd=5.1) average critical swimming speed for DS in cm/sec (Swanson) #can be varied

fish.id = rep(1:n.fish, n.pops))

# adjust the data

#calculate the angle inside the triangle made the fish's escape path, the line perpendicular to the net boundary, and the net boundary itself

#0:90\* = 0 : pi/2

fish$angle.prime <- fish$angle

#90\*:180\* = pi/2 : pi

fish$angle.prime[fish$angle>pi/2 & fish$angle<pi] <- pi - fish$angle[fish$angle>pi/2 & fish$angle<pi]

#180\*:270\* = pi : 3pi/2

fish$angle.prime[fish$angle>pi & fish$angle<3\*pi/2] <- fish$angle[fish$angle>pi & fish$angle<3\*pi/2] - pi

#270\*:360 = 3pi/2 : 2pi

fish$angle.prime[fish$angle>3\*pi/2] <- 2\*pi - fish$angle[fish$angle>3\*pi/2]

#CALCULATE ACTUAL DISTANCE TRAVELED BY FISH:

d.prime <- fish2$distance/as.numeric(cos(fish2$angle.prime))

#CALCULATE ESCAPE TIME:

#When does the fish cross the edge of the path of the net? t=0 is when the fish sees the net.

esc.time <- d.prime/fish2$fish.vel #time (sec) it takes for a fish to swim out of the path of the net

#CALCULATE ESCAPE BASED ON RELATIVE DISTANCE INSTEAD OF TIME

#How far does the fish travel in the X direction (ie in relation to the net)?

dx <- fish2$distance\*as.numeric(tan(fish2$angle.prime))

# fish that swim towards the net go in the negative direction:

dx[which(fish2$angle>pi)] <- 0 - dx[which(fish2$angle>pi)]

#Wrap everything together

fish <- cbind(fish, d.prime, dx, esc.time)

return(fish)

}

# Create a dataset of tow information

# Returns a data.frame with one row for each population or tow

mktow <- function(n.pops, net.vel, n.fish, secchi){

pop <- data.frame(pop = 1:n.pops,

secchi = secchi,

net.vel = net.vel,

n.fish = n.fish)

return(pop)

}

# Calculate which fish escape the net

# Returns the original fish data.frame with new columns for pop values, fish position, and caught status

escfish <- function(fish, pop){

#The position of the fish depends on when it reacted to the net:

fish2 <- merge(fish, pop,

by = "pop",

all.x = TRUE)

# 4. How long does it take the net to travel to where the fish escaped (dx+secchi)? ####

net.time <- (fish2$secchi + dx)/fish2$net.vel

# 5. Does the net get to the fish before the fish gets out? ####

time.diff <- net.time - esc.time

fish2$caught <- as.numeric(time.diff < 0) # TRUE = caught

#Adjust for swimming failures:

# 60% successfully swim, 40% fail to swim.

# binomial w/ p=0.4 --> 0 if fish swim, 1 if they don't.

# if they fail to swim, they get caught (caught)

# add catches from swimming failure to catches from slowness

# change that to a binomial: if caught = 1 or 2, it's 1; if caught = 0, it stays a 0.

fish2$caught <- fish2$caught + rbinom(n = length(fish2$caught), size = 1, prob = 0.4)

fish2$caught <- as.numeric(fish2$caught > 0)

fish2$escaped <- as.numeric(fish2$caught == 0)

fish2 <- cbind(fish2, net.time, time.diff)

return(fish2)

}

# Summarize the proportion of fish caught

# Returns a data.frame of pop with catch and catch.p appended to it

sumcatch <- function(fish, pop, n.fish){

catch <- summaryBy(caught ~ pop, data = fish, FUN = sum)[,2]

catch.p <- catch/n.fish

return(cbind(pop, catch, catch.p))

}

# Set Values ####

# Fish

n.fish <- 1000 # number of fish in the path of each net tow #was 500

n.pops <- 1000 # number of populations to simulate # was 100

width.path = 365.8

# Tows

secchi = round(runif(n.pops, 1, 450), 0) #round(runif(n.pops, 1, 1200), 0)

net.vel = runif(n.pops, 61.9, 84.5)

#c(rep(84.5, 333), rep(72.8, 333), rep(61.89, 334)) #, #72.8,#\*0.75, #rnorm(n.pops, 72.8, 19.6), #net.vel,

# Create Data ####

fish3 <- mkfish(n.pops, n.fish, width.path)

tow3 <- mktow(n.pops, net.vel, n.fish, secchi)

fish4 <- escfish(fish3, tow3)

tow4 <- sumcatch(fish4, tow3, n.fish)

net.vel.st <- (tow4$net.vel - mean(tow4$net.vel))/sd(tow4$net.vel) #same as scale()

**APPENDIX B: CODE TO RUN BAYESIAN REGRESSION**

library(R2OpenBUGS)

#getwd()

#memory.limit(size = 2500)

#http://www.openbugs.net/Manuals/Tutorial.html

#Kery book: U:\Articles\Articles\Stats Books\Whole Books

# see SS8.4.1, p. 105

#Interaction code: https://rpubs.com/mkearney/105689

#i = trawl counter

set.seed(123)

sink("secchi\_effect\_esc7\_2.txt")

cat("

model{

# Priors

alpha ~ dnorm(0.0, 0.01)

beta1 ~ dnorm(0.0, 0.01) #secchi slope prior

beta2 ~ dnorm(0.0, 0.01) #net.vel slope prior

beta12 ~ dnorm(0.0, 0.01) #interaction slope prior

sigma ~ dunif(0, 100)

# Likelihood

for(i in 1:N)

{

catch.p[i] ~ dnorm(mu[i], tau)

mu[i] <- alpha + beta1\*(secchi[i]) + beta2\*(net.vel.st[i]) + beta12\*(secchi[i]\*net.vel.st[i])

}

#Derived quantities

tau <- 1/(sigma \* sigma)

#p.decline <- 1-step(beta) #probability of decline

# Assess model fit using sums of squares type discrepancy

for (i in 1:N){

residual[i] <- catch.p[i] - mu[i] #residuals for observed data

predicted[i] <- mu[i] #predicted values

sq[i] <- pow(residual[i], 2) #squared residuals for observed data

# Generate replicate data and compute fit stats for them

y.new[i] ~ dnorm(mu[i], tau) #one new dataset at each MCMC iteration

sq.new[i] <- pow(y.new[i]-predicted[i], 2) # Squared residuals for new data

}

fit <- sum(sq[]) # Sum of squared residuals for actual data set

fit.new <- sum(sq.new[]) # Sum of squared residuals for new data set

test <- step(fit.new - fit) # Test whether new data set more extreme

bpvalue <- mean(test) # Bayesian p-value

}

", fill=TRUE)

sink()

#kery uses alpha[pop[i]] instead of alpha[i,j] -- not sure if that's different

pop <- tow4 #run TurbSim\_Functions.R

# make the number of escaped fish a poisson random variable"

pop$escape <- rpois(n.pops, lambda = 1000-pop[,"catch"])

# calculate catch from escape in win.data....

#package stuff for winbugs

#bundle data

#openbugs doesn't like data.frames

win.data <- list(secchi = as.matrix(pop)[1:n.pops,"secchi"],

#catch.p = (1000-as.matrix(pop)[1:n.pops,"escape"])/1000,

catch.p = rpois(n.pops, lambda = pop[,"catch"])/1000,

#catch.p = as.matrix(pop)[1:n.pops,"catch.p"],

#n.fish = as.matrix(tow4)[1:n.pops, "n.fish"],

net.vel.st = (pop$net.vel-mean(pop$net.vel))/sd(pop$net.vel),#as.matrix(tow4)[1:n.pops, "net.vel"], #use standardized net.vel to get real parameter estimates

N = n.pops)

#starting values

# inits <- function(){c(alpha=rnorm(1),

# beta=rnorm(1),

# #eta = list(rnorm(50)),

# tau=rgamma(n = 1, shape = 1))}

inits <- function(){ list(alpha = rnorm(1),

beta1 = rnorm(1),

beta2 = rnorm(1),

beta12 = rnorm(1),

sigma = rlnorm(1))}

#Parameters to estimate

# params <- c("alpha",

# "beta",

# "sigma",

# "tau")

params <- c("alpha",

"beta1",

"beta2", "beta12",

"sigma", "bpvalue",

"fit", "fit.new",

"residual", "predicted")

#MCMC settings

nc=3; nt=1

#ni=100; nb=2

ni=3000; nb = 1000

#Start Gibbs Sampler

out <- bugs(data=win.data, inits=inits, parameters.to.save=params,

model.file="secchi\_effect\_esc7\_2.txt", n.thin=nt,

n.chains=nc, n.burnin=nb,

n.iter=ni, debug=TRUE, DIC=TRUE, working.directory=getwd(),

codaPkg = FALSE) #TRUE lets you use the coda package, but breaks the prediction/mean code

print(out, digits = 3)

params.coda <- c("alpha",

"beta1",

"beta2", "beta12",

"sigma", "bpvalue")

out.coda <- bugs(data=win.data, inits=inits, parameters.to.save=params.coda,

model.file="secchi\_effect\_esc7\_2.txt", n.thin=nt,

n.chains=nc, n.burnin=nb,

n.iter=ni, debug=TRUE, DIC=TRUE, working.directory=getwd(),

codaPkg = TRUE)

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

#### End of BUGS Code ####

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

# Credible interval Kery p. 110-111

predictions <- array(dim = c(length(0:450), length(out$sims.list$alpha)))

# net.vel = 0 ==> mean net velocity here (72.8 cm/sec)

for(i in 1:length(0:450)){

predictions[i,] <- out$sims.list$alpha + out$sims.list$beta1\*seq(0, 450)[i]# + out$sims.list$beta2\*0 + out$sims.list$beta12\*0

}

LPB <- apply(predictions, 1, quantile, probs = 0.025) # Lower bound

UPB <- apply(predictions, 1, quantile, probs = 0.975) # Upper bound

plotVelLinesBUGS <- function(net.vel, legend = TRUE){

# center net.vel

net.vel <- (net.vel - mean(pop$net.vel))/sd(pop$net.vel)

for (j in 1:length(net.vel)){

# make predictions

predictions <- array(dim = c(length(0:450), length(out$sims.list$alpha)))

for(i in 1:length(0:450)){

predictions[i,] <- out$sims.list$alpha + out$sims.list$beta1\*seq(0, 450)[i] + out$sims.list$beta2\*net.vel[j] + out$sims.list$beta12\*net.vel[j]\*seq(0, 450)[i]

}

LPB <- apply(predictions, 1, quantile, probs = 0.025) # Lower bound

UPB <- apply(predictions, 1, quantile, probs = 0.975) # Upper bound

# plot

polygon(c(0:450, 450:0),

c(LPB, rev(UPB)),

col = brewer.pal(n = 8, name = "Pastel2")[j], lty = 2, border = NA)

lines(0:450,

out$mean$alpha + out$mean$beta1 \* 0:450 + out$mean$beta2 \* net.vel[j] + out$mean$beta12 \* net.vel[j] \* 0:450,

lwd = 2,

col = brewer.pal(n = 8, name = "Dark2")[j])

}

if (legend == TRUE)

legend("bottomleft",

lty = 1, lwd = 2, col = brewer.pal(n = 8, name = "Dark2")[1:length(net.vel)],

legend = c("Fast", "Average", "Slow"),

title = "Tow Velocity (cm/sec)",

bty = "n")

}

# FIGURE 4 ####

windows()

par(cex = 1.25)

plot(c(0, 450), c(0, 1), type = "n",

xlim = c(0, 450), ylim = c(0.75, 1),

xlab = "Secchi Depth (cm)", las = 1,

ylab = "Proportion Caught",

cex = 1.25,

bty = "l")

points(pop$secchi, pop$catch.p, pch = ".")

plotVelLinesBUGS(net.vel = c(84.5, 72.8, 61.9),

legend = TRUE)

#heat map of posterior distribution of catch.p

pred.heat <- apply(predictions, 1, FUN = function(x) data.frame(table(cut(x, breaks = seq(0, 1, by =0.001))))$Freq)

image(t(pred.heat), ylim = c(0.8, 1), col = rev(brewer.pal(n = 8, name = "YlGnBu")))

contour(x = 0:450,

y = seq(0, 1, by =0.001),

z = pred.heat)

# TABLE 1 ####

#list =

summary(fmwt.sample$Secchi)\*100

# for mean net.vel:

cbind(

#2.5% = LOWER

out$summary[1, 3] + out$summary[2, 3]\*c(0, 39, 59, 68, 85, 457),

#mean

out$summary[1, 1] + out$summary[2, 1]\*c(0, 39, 59, 68, 85, 457),

#97.5% = UPPER

out$summary[1, 7] + out$summary[2, 7]\*c(0, 39, 59, 68, 85, 457))

# Bayesian p-value

mean(out$sims.list$fit.new > out$sims.list$fit)

# Gelman Plot

gelman.plot(out.coda)

**APPENDIX C: EXTRA FIGURES AND TABLES**

**Summary of Bayesian parameter estimates:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Parameter** | **mean** | **sd** | **2.50%** | **25%** | **50%** | **75%** | **97.50%** | **Rhat** | **n.eff** |
| alpha | 1.00060 | 0.00199 | 0.99680 | 0.99930 | 1.00100 | 1.00200 | 1.00400 | 1.004426 | 550 |
| beta1 | -0.00038 | 0.00001 | -0.00040 | -0.00039 | -0.00038 | -0.00038 | -0.00037 | 1 | 1 |
| beta2 | -0.00115 | 0.00200 | -0.00509 | -0.00248 | -0.00113 | 0.00021 | 0.00272 | 1.002215 | 1400 |
| beta12 | 0.00005 | 0.00001 | 0.00004 | 0.00005 | 0.00005 | 0.00006 | 0.00007 | 1.002084 | 1500 |
| sigma | 0.03092 | 0.00069 | 0.02961 | 0.03044 | 0.03091 | 0.03138 | 0.03231 | 1.001042 | 6000 |
| bpvalue | 0.50217 | 0.50004 | 0.00000 | 0.00000 | 1.00000 | 1.00000 | 1.00000 | 1.001283 | 3700 |

**Summary of OLS parameter estimates:**

> summary(lm(catch.p ~ secchi\*net.vel.st, data = pop))

Call:

lm(formula = catch.p ~ secchi \* net.vel.st, data = pop)

Residuals:

Min 1Q Median 3Q Max

-0.039234 -0.005079 0.000034 0.005235 0.034055

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 1.000e+00 5.398e-04 1852.839 <2e-16 \*\*\*

secchi -3.848e-04 2.079e-06 -185.037 <2e-16 \*\*\*

net.vel.st -8.285e-05 5.293e-04 -0.157 0.876

secchi:net.vel.st 5.139e-05 2.070e-06 24.827 <2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.00849 on 996 degrees of freedom

Multiple R-squared: 0.9732, Adjusted R-squared: 0.9731

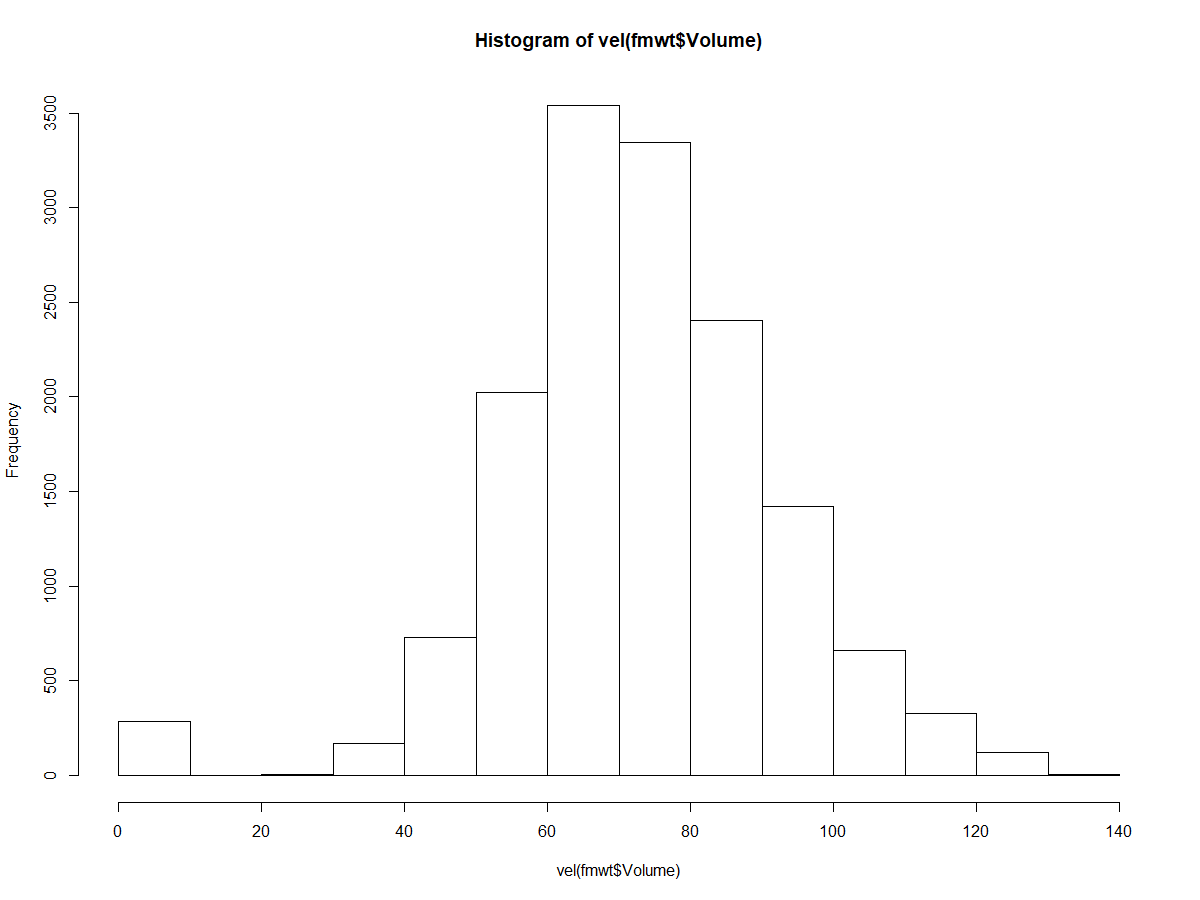
F-statistic: 1.206e+04 on 3 and 996 DF, p-value: < 2.2e-16

**Summary of tow velocities, calculated from tow volumes and a 12-minute target duration**

> summary(vel(fmwt$Volume))

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

0.00 61.89 72.15 72.78 84.52 130.56 8407



**Test for trends in Secchi depth over time in FMWT data:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Whole Time Series | |  |  |  | Since Clams | |  |  |
|  | linear Regression | | Mann-Kendall | |  | linear Regression | | Mann-Kendall | |
|  | slope | p | tau | p |  | slope | p | tau | p |
| Sept | 0.0050055 | < 0.001 | 0.392 | 8.87E-05 |  | 1.44E-02 | < 0.001 | 0.596 | 2.74E-06 |
| Oct | 5.42E-03 | < 0.001 | 0.345 | 0.00048113 |  | 1.35E-02 | < 0.001 | 0.544 | 1.85E-05 |
| Nov | 8.59E-03 | < 0.001 | 0.52 | 1.19E-07 |  | 1.53E-02 | < 0.001 | 0.639 | 4.77E-07 |
| Dec | 7.06E-03 | < 0.001 | 0.424 | 2.24E-05 |  | 9.37E-03 | < 0.001 | 0.385 | 0.002483 |

**Relationship between Secchi Depth and DSM catch in the FMWT:**

